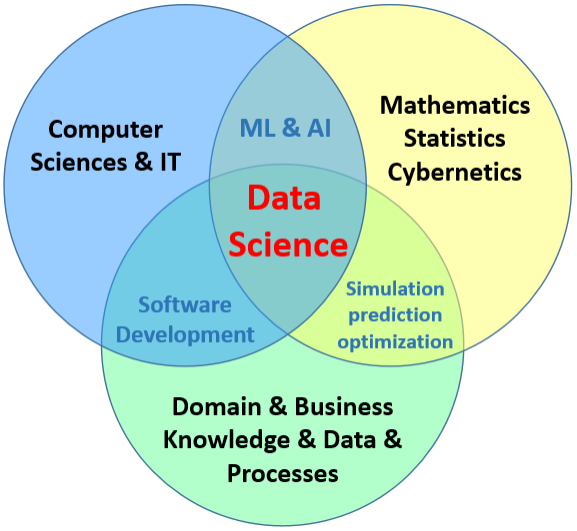
Мокін В.Б., Дратований М.В.



Наука про дані: машинне навчання та інтелектуальний аналіз даних

Міністерство освіти і науки України
Вінницький національний технічний університет

НАУКА ПРО ДАНІ: МАШИННЕ НАВЧАННЯ ТА ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

Навчальний посібник

Вінниця
ВНТУ
2024

УДК [004.65-047.44+004.8/.9](075.8)
М 74

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол №__ від __.__.2023 р.).

Рецензенти:
Іванчук Я. В., доктор технічних наук, професор
Квєтний Р. Н, доктор технічних наук, професор
Литвин В. В., доктор технічних наук, професор

Мокін, В.Б.

М 74 Наука про дані: машинне навчання та інтелектуальний аналіз даних – Електронний навчальний посібник / В. Б. Мокін, М. В. Дратований – Вінниця : ВНТУ, 2024. – 263 с.

У навчальному посібнику викладено теоретичні відомості про основні поняття, методи і засоби науки про дані (Data Science), машинне навчання і штучний інтелект (Machine Learning & Artificial Intelligence) та інтелектуальний аналіз даних (Intellectual Data Analyze), а також практичні рекомендації для застосування сучасних технологій при розв'язанні численних реальних задач та проблем системного аналізу. Наведено перелік контрольних запитань для перевірки набутих теоретичних знань та практичних навичок.

Посібник рекомендується для здобувачів вищої освіти, які навчаються за спеціальностями 124 – «Системний аналіз» та 126 – «Інформаційні системи та технології» I, II та III рівнів при вивченні наступних дисциплін: «Інформаційні технології моніторингу та аналізу даних», «Інформаційні технології моніторингу та аналізу стану складних систем», «Інформаційні інтелектуальні технології», «Інтернет речей та інтелектуальний аналіз даних», «Інтелектуальний аналіз даних», «Системний аналіз», «Інформаційні технології обробки зображень», «Технології машинного навчання», «Аналіз зображень», «Смарт-технології», «Аналіз даних», а також – для студентів, які проходять навчальну, виробничу і переддипломну практику, та для педагогічної практики аспірантів.

УДК [004.65-047.44+004.8/.9](075.8)

ЗМІСТ

ВСТУП

Наука про дані, технології штучного інтелекту, машинне навчання моделей та інтелектуальний аналіз даних з використанням цих моделей стають дедалі актуальнішими та поширенішими в усіх сферах нашого життя. Особливо бурхливо зараз розвиваються напрями, основані на використанні великих мовних моделей та сервісів на кшталт ChatGPT від OpenAI. Однак, ефективність розв'язання будь-якої задачі залежить від її правильної постановки, вибраних інформаційних технологій та архітектури моделей, вибраних методів аналізу даних та результатів передбачення з використанням цих моделей. ChatGPT та його аналоги розв'язують багато задач, але далеко не усі – все одно, потрібний датасайнтист, тобто вчений з науки про дані, який правильно поставить задачу, сформулює запит, верифікує відповідь (ChatGPT часто «галюціонує», тобто синтезує випадкову, а не – правильну відповідь), використає її для розв'язання задачі. Розв'язувати задачі стало набагато легше, робота стала більш інтелектуальною, а не – рутинною. Про це та інше буде у цьому посібнику.

Мета даного посібника – ознайомити студентів та аспірантів із основними поняттями науки про дані, основними знаннями і навичками технологій штучного інтелекту та машинного і глибокого навчання, методами і засобами інтелектуального аналізу даних, необхідними для розв'язання реальних задач різної складності, а також – для оптимального вибору інформаційних технологій та сервісів для автоматизації цього процесу.

Посібник має чітку практичну спрямованість, тобто весь матеріал викладений в контексті розв'язання прикладних задач. Автори не ставили за мету дублювати класичні видання з теорії штучного інтелекту, чи машинного навчання, чи наукі про дані. Чимало визначень наведено в авторській інтерпретації. Інформаційні технології описані максимально стисло, але достатньо, щоб розуміти відмінності між ними і щоб можна було вибрати оптимальні для тієї чи іншої задачі. В тексті є більше сотні посилань на різні Python-ноутбуки з програмними рішення щодо кожної теми. Крім того, є багато посилань на сучасні складні програмні рішення, які неможливо отримати, коористуючись одним ChatGPT чи документацією. У більшості розділів є опис рішень, які перемогли на конкурсах датасайнтистів, передусім 2022 і 2023 років, з використанням дійсно сучасних рішень та з прикладами їх ефективного застосування – вивчення такого досвіду дозволить суттєво удосконалити рівень знань читачів.

Посібник буде корисним здобувачам вищої освіти, які навчаються за спеціальностями 124 – «Системний аналіз» та 126 – «Інформаційні системи та технології» I, II та III рівнів при вивченні наступних дисциплін: «Інформаційні технології моніторингу та аналізу даних», «Інформаційні технології моніторингу та аналізу стану складних систем», «Інформаційні

інтелектуальні технології», «Інтернет речей та інтелектуальний аналіз даних», «Інтелектуальний аналіз даних», «Системний аналіз», «Інформаційні технології обробки зображень», «Технології машинного навчання», «Аналіз зображень», «Смарт-технології», «Аналіз даних», а також – для студентів, які проходять навчальну, виробничу і переддипломну практику, та для педагогічної практики аспірантів. Матеріал посібника може бути корисним і для слухачів другої вищої освіти з системного аналізу, з інформаційних систем та технологій та студентам інших спеціальностей, які недостатньо володіють основними знаннями та навичками роботи у цій сфері.

Посібник містить такі розділи:

1. Загальні аспекти постановки та розв’язання задач науки про дані та поняття штучного інтелекту.
2. Передоброблення та розвідувальний аналіз даних (EDA).
3. Інженерія ознак (Feature Engineering).
4. Побудова моделей машинного навчання.
5. Нейронні мережі та глибоке навчання.
6. Інтелектуальний аналіз зображень та відео.
7. Інтелектуальний аналіз та оброблення природномовного тексту.
8. Інтелектуальний аналіз і прогнозування часових рядів.

У Додатку Б наведено рекомендації для створення власного датасету на платформі Kaggle, а у Додатку В – приклади постановки задачі у конкурсах Kaggle (приклади їх розв’язання цих задач наведені в кінці відповідних розділів посібника).

Перший розділ – це основні поняття, постановка задачі та як вибрати чи побудувати власний датасет. Другий і третій розділи – як дослідити датасет та удосконалити його. Четвертий і п’ятий розділи – це машинне навчання моделей (п’ятий – нейромережевих) на основі сформованого у розділах 1-3 датасету. Розділи 6-8 присвячений інтелектуальному аналізу даних. Тобто, важливо зазначити, що самі моделі машинного навчання, побудові яких присвячені розділи 1-5, не є самоціллю. Як правило, їх будують для розв’язання прикладних задач з використанням технологій інтелектуального аналізу даних, тобто з використанням спеціальних технологій, які враховують специфіку даних, та з використанням передтренованих моделей машинного навчання. Тому, хоча наведений у розділах 1-5 матеріал й дозволяє розв’язувати задачі довільної складності, але більш ефективним є їх використання разом із матеріалом розділів 6-8: зображення і відео (розділ 6), природномовні тексти (розділ 7), ряди даних (розділ 8). З іншого боку, ефективне розв’язання задач інтелектуального аналізу даних у розд. 6-8 з використанням передтренованих моделей машинного навчання, потребує розуміння як подібні моделі розроблюються. А у більшості випадків, треба вміти самому робити такі моделі, тому перед ознайомленням з розділами 6-8 варто ознайомитись з розділами 1-5.

Усі програми у цьому посібнику – на Python. У Додатку А наведено основні команди Python, типи даних та базові бібліотеки на Python, які слід мінімально знати для ознайомлення з подальшими розділами.

Під час викладення матеріалу буде враховано те, що зараз легко дізнатись в ChatGPT зміст та параметри будь-якої команди, оператора, функції, бібліотеки, тому такий матеріал буду викладатись мінімально – викладатись будуть тільки ключові для кожної команди чи методу або моделі параметри чи аспекти, на які обов'язково слід звернути увагу.

У кожному розділі будуть наводитись огляди прикладів розв'язання реальних задач, з урахуванням багаторічного досвіду авторів, та задач тренувального характеру з платформи датасайнтистів Kaggle, яка станом на листопад 2023 року містить вже біля 16 млн. акаунтів, 48 млн. ноутбуків та 4 млн. датасетів датасайнтистів з усього світу (див. метадані Kaggle). Крім того, будуть наводитись посилання на готовий програмний код на Python з ілюстрацією викладеного матеріалу на прикладі розв'язання таких реальних чи тренувальних завдань. Кожен розділ буде завершуватись списком важливих команд і функцій, які слід знати чи опанувати, та – списком контрольних питань для самоперевірки опанування матеріалу.

Автори посібника мають великий досвід розв'язання реальних задач у сфері медицини та біології, екології та метеорології, економіки та торгівлі, електроніки та Інтернету речей, енергетики та електромеханіки, транспорту та керування дронами, дешифруванні даних дистанційного зондування Землі, у т.ч. аерофотозйомки, тощо з використанням технологій штучного інтелекту, машинного навчання та технологій інтелектуального аналізу даних. Окрім реальних задач, автори мають значні здобутки у рейтингах платформи Kaggle. Професор Віталій Мокін має звання Kaggle Notebooks Grandmaster (першим в Україні отримав це звання, тоді їх у світі було лише біля 60 осіб), а у світовому рейтингу Kaggle з розробки ноутбуків досяг 10-го місця! Має більше 600 ноутбуків на Python, з яких майже половина є публічними. і саме вони будуть використовуватись у даному посібнику як приклади. Михайло Дратований має значний досвід застосування Python-рішень для різних прикладних завдань, був відповідальним виконавцем роботи з оптимізації нейромережевих моделей оптимізації енергозбереження заренових елеваторів за фінансування Програми «Горизонт» ЄС і бере участь у Kaggle-змаганнях з 2019 р.

Основний текст посібника написано професором Віталієм Мокіним (9 авторських аркушів), але практичні аспекти застосування усіх технологій та особливості машинного навчання з підкріпленням описував Михайло Дратований (підрозділи 2.4, 3.4, 4.4, 5.4, 6.4, 7.4, 8.4, а також пункти 1.4.1 та 6.1.3, усі контрольні питання та опис частини практичних завдань, загалом: 1,68 авторських аркушів).

Авторські Kaggle-ноутбуки присвячені розв'язанню усіх видів задач, які розглянуті у цьому посібнику. Саме з них брались багато рисунків та пояснень. Читачі можуть їх скопіювати і адаптувати до своїх задач,

користуючись цим посібником. По багатьох з цих ноутбуків ще й є україномовні лекції Мокіна В.Б. в його YouTube-каналі «Курс AI-ML-DS Training на Python».

Колектив авторів висловлює подяку доктору технічних наук, професору кафедри системного аналізу та інформаційних технологій ВНТУ, проф. Мокіну О.Б. за цінні поради з різних аспектів.

Навчальний посібник за такою прикладною концепцією публікується вперше. Просимо надсилати коментарі, зауваження та рекомендації щодо його удосконалення на пошту sait@vntu.edu.ua кафедри системного аналізу та інформаційних технологій ВНТУ, де працюю автори посібника. Незабаром, буде видана друга версія, в якій будуть усунені усі виявлені неділоки і суттєво додано приклади ефективного застосування наведених у посібнику методів, моделей та технологій.

1 ЗАГАЛЬНІ АСПЕКТИ ПОСТАНОВКИ ТА РОЗВ'ЯЗАННЯ ЗАДАЧ НАУКИ ПРО ДАНІ ТА ПОНЯТТЯ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Основні поняття науки про дані, машинне навчання, штучний інтелект та інтелектуальний аналіз даних

В усіх сферах нашого життя треба вміти оптимально та коректно опрацьовувати інформацію: збирати її, аналізувати, прогнозувати, використовувати для прийняття обґрунтованих рішень тощо. Узагальнено весь комплекс підходів, прийомів, методів і засобів для вирішення цих задач називається «*Наука про дані*» (англ.: «Data Science», скорочено «DS»). Основними складовими DS є такі:

- *Інженерія даних* (англ. «Data Engineering» – DE), яка охоплює методи і засоби збирання та управління даними, їх попереднього аналізу та передоброблення для використання в задачах машинного навчання;

- *Машинне навчання* (англ. «Machine Learning» – ML) – комплекс прийомів, методів та технологій з розв'язання прикладних задач, якому передує «навчання» моделей на комп'ютерах (машинах) на певних даних;

- *Штучний інтелект* (або технології штучного інтелекту) (англ. «Artificial Intelligence» – AI) – комплекс прийомів, методів та технологій машинного навчання з використанням імітації різних аспектів людського пізнання, таких як навчання, вирішення проблем, міркування, сприйняття та прийняття рішень;

- «*Інтелектуальний аналіз даних*» (Intelligent Data Analysis – IDA) – розв'язання прикладних задач з аналізу різноманітних даних з використанням комплексу прийомів, методів та інтелектуальних технологій.

Ще варто зазначити що є інтелектуальними моделями. Існує багато визначень. Ми пропонуємо таке: *інтелектуальна модель* – це модель, побудована для ефективного розв'язання аналітичної задачі, яка здатна навчатися на основі досвіду та узагальнювати знання для опрацювання нових даних та сценаріїв. Це визначення відображає головну відмінність такого роду моделей, яка полягає в тому, що вони дозволяють передбачати з високою точністю дані, події, генерувати нові знання та інформацію. Якщо інформаційна система чи технологія просто дуже добре і швидко порівнює вхідні дані з базою зразків (відбитки пальців, ДНК, обличчя, кадастрова інформація тощо), то вона не є інтелектуальною, а ось якщо вона може передбачити як зміниться задане обличчя через 10 років чи на 10 років раніше, то це вже ознака її інтелектуальності у сенсі цього визначення.

«Штучний інтелект» ще часто поділяють на 2 підвиди:

- *Слабкий або вузький штучний інтелект* (narrow AI, weak AI) – це штучний інтелект, який здатний вирішувати конкретні завдання, але не здатний до загального інтелекту. Наприклад, слабкий штучний інтелект може використовуватися для розпізнавання мови, машинного перекладу, гри в шахи тощо.

- *Сильний або загальний штучний інтелект* (general AI, strong AI) – це штучний інтелект, який здатний до загального інтелекту, тобто здатний до мислення, навчання, вирішення проблем, тощо, як людина. Сильний штучний інтелект також називають штучним розумом (artificial intelligence, AI). Поки він існує тільки в теорії.

На жаль, у різній літературі зазначених вище термінів сильно перекривається і часто вони вживаються як взаємозамінні. Однак, варто відзначити такі важливі особливості:

1. Машинне навчання (ML) зосереджується, передусім, на навчанні машинних моделей за інформацією з датасетів, а інженерія даних (DE) – на створенні та управлінні цими датасетами. Саме тому вони так і називаються. Але деякі автори публікацій та роботодавці під час оголошення вакансій на дата-інженера чи ML-розробника часто включають ML в DE або DE в ML.

2. Штучний інтелект зосереджується, передусім, на використанні нейромережевих моделей або інших моделей, які імітують особливості людського пізнання (нечітка логіка, генетичні алгоритми, баєсівське чи когнітивне моделювання тощо), а машинне навчання охоплює не тільки ці, а й усі можливі інші підходи та методи створення машинних моделей, тому ML є більш загальним, аніж AI. Водночас, AI вивчає й будову людського мозку, особливості людського пізнання та різні напрямки, які, в загальному випадку, не відносяться до ML.

3. Інтелектуальний аналіз даних (IDA), як правило, полягає у використанні спеціально розроблених інформаційних технологій та передтренованих інтелектуальних моделей для розв’язання прикладних аналітичних задач. Хоча, деякі автори часто розглядають ML/AI як один з початкових етапів IDA або IDA – як один з фінальних етапів ML або AI.

В даному посібнику пропонується такий спосіб структурування цих термінів, концепцій, методів та технологій (рис. 1.1):

1. Спочатку здійснюється збирання та передоброблення даних (Data), у т.ч. великих даних (Big Data), і формування датасетів для аналізу даних – інженерія даних (DE), яку можна розглядати як розділ науки про дані (DS) або як самостійний розділ.

2. За результатами DE здійснюється розвідувальний аналіз даних (англ. «Exploratory Data Analysis» – EDA), який передбачає прикладне застосування окремих розділів математики, у т.ч. статистики, до даних. Також, цей аналіз може передбачати пробну побудову моделей для аналізу певних закономірностей щодо ознак. Причому, такими моделями можуть бути й нейронні мережі (DL).

3. За результатами EDA проводиться машинне навчання (ML) моделей та моделей глибокого навчання (DL) й штучного інтелекту (AI).

4. Передтреновані (переднавчені) моделі ML/DL/AI використовуються для інтелектуального аналізу даних (числа, текст, зображення, відео, мова, звуки) (IDA), спрямованого на розв’язання різних прикладних аналітичних задач.

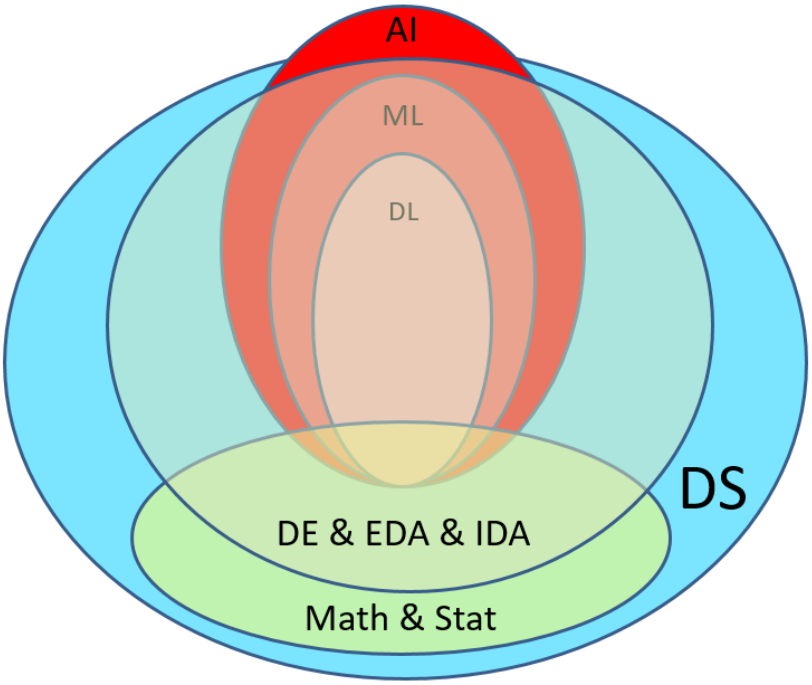


Рисунок 1.1 – Основні поняття та їх поєднання у сфері ML/DL/AI/DE/DS

Усіх їх об'єднує наука про дані (DS), у т.ч. DE, хоча усі вони мають деякі аспекти програмно-апаратного плану, які не відносяться до DS, а тому вони лише перекриваються.

DS є більш загальним поняттям, аніж математика, у т.ч. статистика, та DE, EDA, IDA, а тому вона їх охоплює повністю.

Для автоматизації усіх цих операцій використовується прикладне програмне забезпечення та інтегровані середовища розробки (англ. «Integrated Development Environment» – IDE).

Такий спосіб структурування відповідає поділу задач в IT-компаніях на датаінженерів та датасайнтистів. У свою чергу, датасайнтисти або створюють нові моделі ML/DL/AI, або, частіше, використовують вже готові моделі, але розвивають технології їх застосування для розв'язання нових задач. Найбільш цінним в наш час є вміння застосовувати саме IDA, але це неможливо якісно без розуміння основ побудови моделей ML/DL/AI, тому важливо опанувати усі ці аспекти DS.

Слід зазначити, що сфера поєднання DS/ML/DL/AI стосується, передусім такого класу проблем, як слабоструктуровані.

Як відомо, в системному аналізі розрізняються 3 види проблем:

- *добре структуровані* (англ. «well-structured»), або кількісно сформульовані проблеми, в яких істотні залежності з'ясовані дуже добре;

- *неструктуровані* (англ. «unstructured»), або якісно виражені проблеми, що містять лише опис найважливіших ресурсів, ознак і характеристик, кількісні залежності між якими абсолютно невідомі;

- *слабо-структуровані* (англ. «ill-structured»), або змішані проблеми, які містять як якісні елементи, так і маловідомі, невизначені сторони, які мають тенденцію домінувати.

Комплекс технологій DS/ML/DL/AI є найбільш ефективним саме щодо слабо-структурованих проблем або – щодо неструктурованих, які він дозволяє перевести у клас слабо-структурованих. Етапи EDA та ML/DL/AI стосуються більше неструктурованих проблем, а етап IDA застосовується вже до слабо-структурованих. Саме цьому виду проблем і присвячений цей посібник.

З іншого боку, якщо методи і технології DS/ML/DL/AI застосовуються до даних, які характеризують певну складну систему (як правило це – система з невизначеністю) і вони допомагають зробити певні висновки щодо цієї системи в цілому (виявити закономірності, побудувати моделі, знайти оптимальні рішення задач, обгрунтувати рекомендації тощо), тоді їх можна називати ще й методами і технологіями системного аналізу. За цих умов, варто під час аналізу застосовувати й специфічні методи системного аналізу та метолодогію системного підходу.

Щодо DS/ML/DL/AI в літературі є поширеним дещо відмінне поєднання понять з рис. 1.1. З деякими удосконаленнями відповідна схема наведена на рис. 1.2.

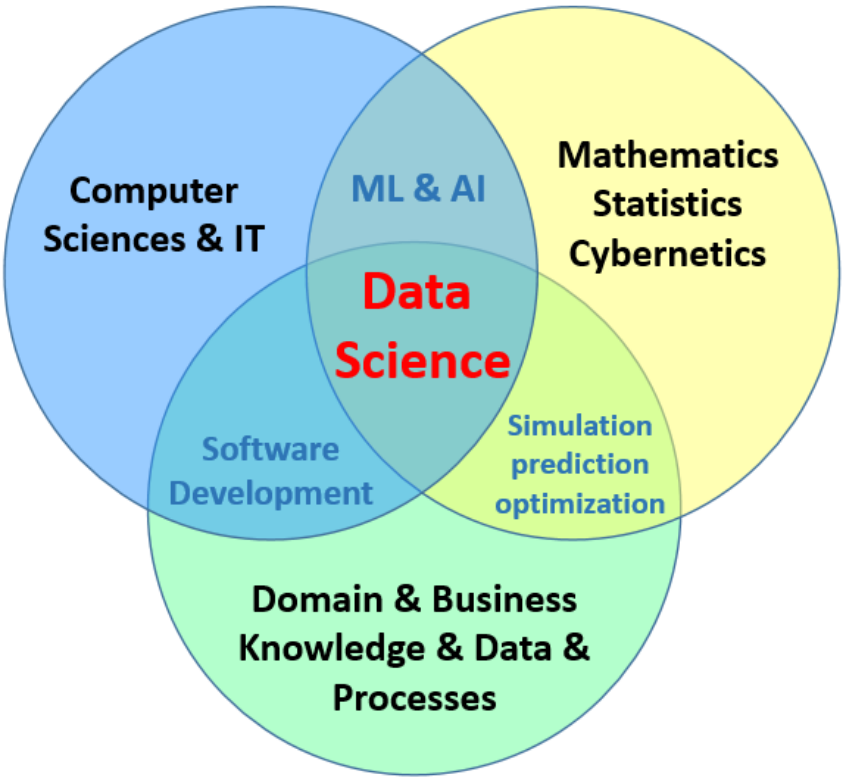


Рисунок 1.2 – Основні поняття та їх поєднання у сфері Data Science

На рис. 1.2 показано, що інформатика (IT та комп'ютерні науки – англ. «Computer Sciences») у поєднанні зі знаннями і даними про предметну об-

ласть (англ. «Domain Knowledge & Data») та знаннями і даними про показники ефективності, вимоги користувачів чи клієнтів, бізнес-стратегії та обмеження тощо (англ. «Business Knowledge & Data») утворюють «Software Development», тобто – розроблення програмного забезпеченняА поєднання математики, у т.ч. статистики, та кібернетики, у т.ч. теорії керування та оптимізації систем, з цими знаннями це – традиційні дослідження у сфері математичного моделювання, прогнозування та оптимізації процесів і систем. Поєднання ІТ та комп'ютерних наук з математикою, у т.ч. статистикою, дало створити машинне навчання та штучний інтелект (ML & AI). А на стику їх усіх утворилась наука про дані (DS). Отже, для опанування DS для конкретної області чи задачі треба і знати математику, передусім – статистику, і вміти розробляти програмне забезпечення, передусім – на Python, і володіти знаннями про цю предметну область та про постановку задачі, передусім – щодо вимог, обмежень та критеріїв ефективності методів і технологій.

Як відомо у сфері науки про дані розрізняють 3 типи машинного навчання за видом навчання:

– Навчання з учителем (supervised learning) – у навчанні з учителем модель отримує набір даних, які включають вхідні функції та відповідні цільові значення. Метою моделі є навчитися відображати зв'язок між вхідними функціями та цільовими значеннями, щоб вона можала зробити прогнози для нових даних. Наприклад, у задачі передбачення ціни будинку, модель навчається на основі раніше відомих пар "характеристики будинку - його ціна".

– Навчання без учителя (unsupervised learning) – у навчанні без учителя модель отримує набір даних, але без відповідних цільових міток або позначень. Модель намагається знайти приховані закономірності або структуру в даних, групуючи їх на основі схожості або інших характеристик. Наприклад, у задачі кластеризації текстових даних, модель намагається згрупувати схожі документи без будь-яких заздалегідь визначених категорій.

– Навчання з підкріпленням (reinforcement learning) – у навчанні з підкріпленням модель взаємодіє з динамічним середовищем, приймаючи рішення і отримуючи нагороду або покарання за кожну дію. Метою моделі є максимізація загальної нагороди або мінімізація загального покарання. Наприклад, у задачі навчання робота керувати літаком, модель навчається приймати рішення про керування літаком на основі навколишнього середовища та максимізувати час польоту або мінімізувати кількість аварій.

У подальшому, якщо це не зазначено окремо, буде йтись про машинне навчання "з учителем". Про машинне навчання без учителя див. підрозділ 2.2 "Кластеризація та зменшення розмірності даних" та пункт 6.2.3 "Автоенкодери у задачах без учителя". Про машинне навчання з підкріпленням -

див. підрозділ 5.4 "Машинне навчання з підкріпленням (Reinforcement Learning)".

Важливо усвідомлювати, що в прикладному плані наука про дані, машинне навчання, штучний інтелект, інтелектуальні технології є, передусім, інформаційними технологіями, які здійснюють перероблення інформації з вхідної на вихідну за певними алгоритмами, спрямованими на збільшення кількості інформації *I*, передусім – на кращу систематизацію і формалізацію, виявлення і передбачення нових знань і закономірностей. Ці алгоритми, залежно від невизначеності вхідних даних, структури моделей чи розгалуженості і багатоваріантності алгоритму, мають різну складність *S*. Отже, для кращого розуміння матеріалу, у посібнику буде використана відповідна інфографіка, яка буде демонструвати яким чином кожен блок інформаційних технологій можна розташувати в такій системі координат *S(I)* (рис. 1.3).

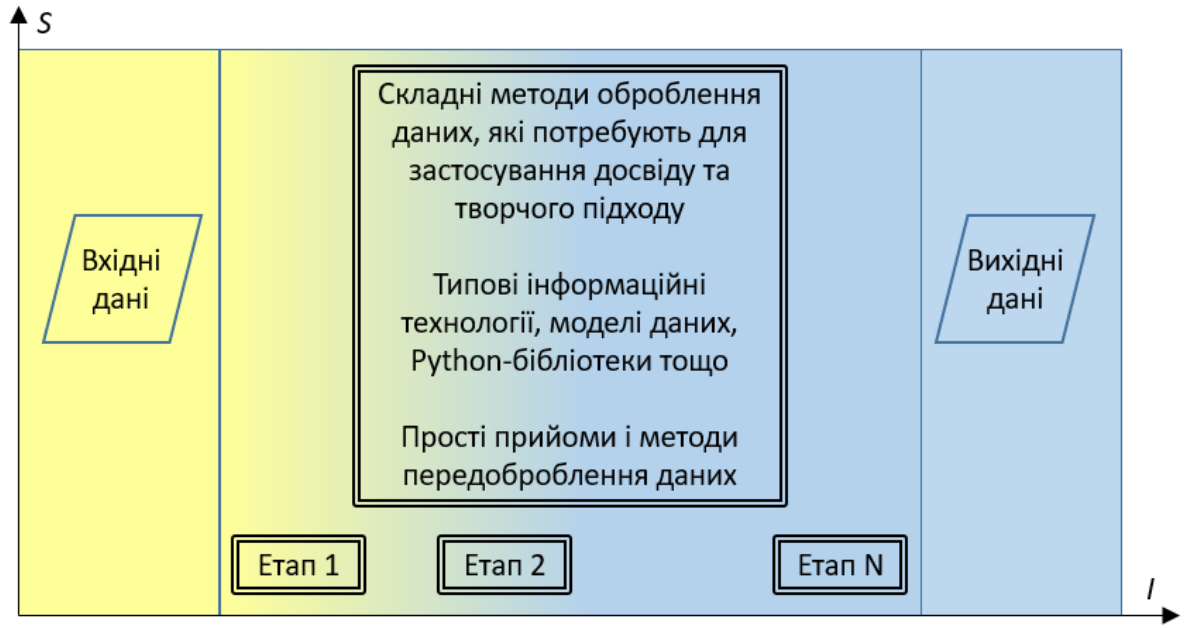


Рисунок 1.3 – Інфографіка для візуалізації інформаційних технологій для перетворення вхідних даних на вихідні, в залежності від складності *S* алгоритму та кількості інформації *I*, яку вони додають

У разі, якщо на діаграмі буде багато блоків, то для кращого розуміння їх послідовності по осі абсцис ще можуть додаватись проєкція цих блоків на неї у вигляді зелених вертикальних ліній від центру блоків до цієї осі.

У кожному розділі будуть наводитись приклади постановки та напрямки розв’язання і реальних задач, і тренувальних задач у платформі Kaggle з відповідної тематики. Посібник орієнтований на використання мови програмування Python (див. Додаток А з деякою інфографікою та посиланнями на документацію і збірники задач, що дозволить прискорити її освоєння, за наявності знань вже хоча б з якоїсь іншої мови програмування та з основ алгоритмізації).

1.2 Постановка задачі аналізу даних. Пошук по ній інформації. Побудова датасету та його поділ на тренувальний, валідаційний і тестовий

Переважна більшість задач аналізу даних зводиться до 2 класів:

1. *Задачі розвідувального аналізу даних*, де достатньо саміх даних, по яких можна провести аналіз закономірностей, залежностей, первинний статистичний аналіз тощо без використання складних моделей.
2. *Задачі інтелектуального аналізу та передбачення даних*, які потребують використання складних моделей для проведення передбачення високої точності, після чого або проводиться аналіз зроблених передбачень, або аналізується сама модель, чия гарна прогностична функція довела її адекватність.

Задачі другого класу, у свою чергу, поділяються на задачі, які можна ефективно вирішити з використанням передтренованих моделей, і – на задачі, які потребують побудови нової моделі.

Даний посібник присвячений найбільш складному випадку, коли треба провести і розвідувальний аналіз даних, і інтелектуальний аналіз даних, для якого ще треба побудувати моделі за датасетами, які ще треба створити за даними, які ще треба знайти.

Для першого класу задача ставиться, як правило, як вимога *провести розвідувальний аналіз даних* для виявлення наявності та ідентифікації закономірностей, у т.ч. статистичних; виявлення помилкових, аномальних та проблемних даних; для виявлення зв'язків між даними, їх групування і кластеризації; для візуалізації отриманих висновків у зручній для сприйняття формі. Можуть бути й ширші постановки задач.

Більшість задач другого класу – це задачі оптимізації. Варто розрізняти задачі машинного навчання інтелектуальних моделей та задачі ІАД. Навіть, якщо це не сформульовано явно, оптимізація може здійснюватись у функціях бібліотек, які використовуються для поліпшення точності навчання моделей.

Класичні оптимізаційні задачі в теорії систем, як правило, формулюються таким чином: для заданих вхідних даних, змінних керування, за умов впливу контрольованих і неконтрольованих збурень забезпечити оптимум критерію оптимізації за певних обмежень.

У задачах машинного навчання виділяються тільки вхідні дані і так званий «таргет» (калька з англ. «target» – «цільова ознака»), хоча таргетів може бути багато або усі дані по черзі можуть бути таргетом, наприклад, аналіз взаємозв'язку між алергенами пацієнта [1]. Часто усі ці дані розташовуються в одній таблиці. Критерієм оптимізації є числовий показник (метрика чи похибка). Обмеження можуть бути відсутніми. По суті, їх роль виконує обмеження використовувати тільки значення із датасету і заборона використовувати інші. Але можуть бути й певні фізичні обмеження, наприклад, заборона дробових значень: у разі використання регресійних моделей

для передбачення чи прогнозування кількості об’єктів слід округляти всі значення до цілого і тільки потім визначати метрику (див. приклад визначення кількості тих, хто вижив на «Титанику» у ноутбуці). Або: кількість хворих на коронавірус не може бути від’ємною та ін.

Отже, класична *задача машинного навчання інтелектуальної моделі*, як правило, формулюється таким чином: для заданого датасету (таблиці даних, текстові файлі, аудіо-, відеофайли чи зображення) побудувати і навчити інтелектуальну модель, яка забезпечить оптимум (максимум чи мінімум) заданої метрики (метрик) для заданої цільової ознаки (ознак). Можуть бути додаткові обмеження, але не обов’язково.

Задачами інтелектуального аналізу може бути оптимізація використання різних моделей, методів та технологій для виявлення складних зв'язків та закономірностей у різноманітних наборах даних, у т.ч. з числовою, текстовою, графічною статичною та/чи динамічною інформацією, з метою формування передбачень чи отримання інсайтів для ефективного прийняття рішень та розв'язання завдань у різних галузях. Можуть бути й інші (більш загальні чи більш вузькі) постановки задач ІДА.

Єдиного алгоритму постановки і розв’язання усіх подібних задач не існує, хоча нижче у цьому розділі наведено ряд узагальнених алгоритмів і рекомендацій для найбільш складних варіантів постановки задачі, але вони, на жаль, не охоплюють усі можливі варіанти.

Відмінність в алгоритмах може бути й у тому, з якого етапу починається розв’язання задачі. У змаганнях та навчальних прикладах вже є датасети і постановка задачі, яку треба розв’язувати. У реальних задачах:

Варіант 1. Може бути відомим датасет, наприклад, дані медичних аналізів, і лише бажання покращити ефективність лікування чи більш точно оцінювати поширення хвороби у країні, а як саме це зробити і за якими показниками – невідомо, тобто точні постановки задач ще слід сформулювати самостійно.

Варіант 2. Може бути відомою постановка задачі, наприклад, підвищити точність щодобового прогнозування офіційно оприлюденної кількості хворих на коронавірус в країні, але які фактори враховувати – невідомо, тобто датасет ще треба створити, а створивши, уточнити постановку задачі, оскільки не всі дані, зазвичай, вдається знайти, відповідно до вимог.

Варіант 2 має місце частіше і він є найбільш узагальненим, а варіант 1 є його спрощеним випадком. Тому більшу увагу будемо приділяти варіанту 2.

У разі, коли задача вирішується «з нуля», тобто є лише постановка задачі (варіант 2) про те, що саме слід навчитись вирішувати, а даних немає ніяких, тоді рекомендуємо скористатись таким алгоритмом:

1. Уточнити ключові слова для опису предметної області з використанням експертів, спеціальної літератури чи, хоча б, чат-ботів. Зробити серію англомовних запитів із заданої тематики в ChatGPT (чи аналогічних сер-

вісах), саме – англомовних, тоді результати будуть більш повними і релевантними. Основні цілі такого вивчення: аналіз сучасних підходів і технологій до розв'язання задачі; пошук публічних датасетів з цієї задачі; визначення переліку англомовних ключових слів з тематики задачі для подальшого пошуку.

2. Вивчити топові (перші сторінки списку) англомовні статті із відібраними у п.1 ключовими словами у сервісі Google Scholar. Для більшої ефективності слід у вікні зліва вибрати останній чи передостанній рік, а нижче – зняти пташечку «включаючи цитування» (рис. 1.4). Основні цілі такого вивчення: аналіз сучасних підходів і технологій до розв'язання задачі та конкретних їх результатів, у т.ч. у графічному вигляді; перевірка достовірності відомостей, які навів ChatGPT у п.1; пошук публічних датасетів, які використовували автори статей та які порадив використовувати ChatGPT (сучасні топові журнали вимагають публікувати і датасети, і програми для їх оброблення з результатами, наведеними у статтях – найчастіше їх можна знайти у GitHub або Kaggle); уточнення переліку англомовних ключових слів з тематики задачі для подальшого пошуку.

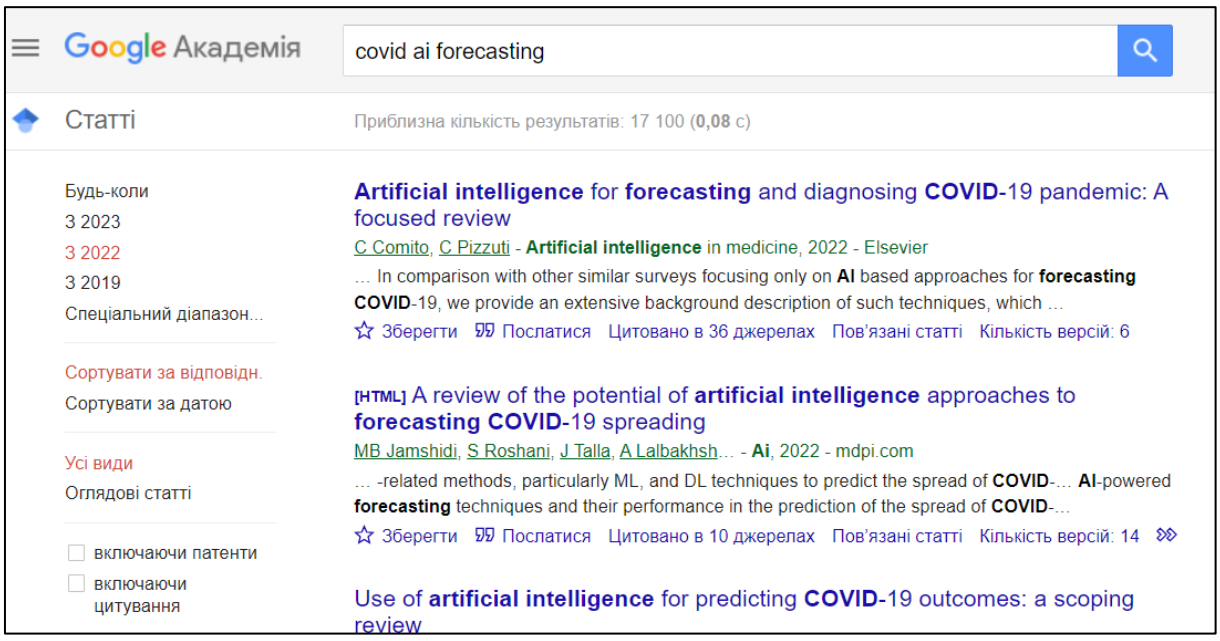


Рисунок 1.4 – Приклад результатів пошуку статей у Google Scholar з тематики прогнозування у сфері коронавірусу з використанням технологій штучного інтелекту, за останні 2 роки

3. Здійснити пошук релевантних ноутбуків Kaggle та вивчити використані у них датасети (розділ «Input»). Однак, пошукова система Kaggle усі ключові слова поєднує за операцією «OR» і це поки, на жаль, не можна змінити, тому шукати доцільно тільки за одним словом, або шукати засобами Google-пошуку, додаючи в пошук слово «kaggle» – саме в лапках, вимагаючи його обов'язкову наявність у результатах пошуку, тоді, як правило, результатом пошуку буде саме датасети і програми в Kaggle. При цьому, варто

щоразу уточняти ключові слова для опису предметної області з використанням загально прийнятої у машинному навчанні термінології.

4. Здійснити пошук датасетів у GitHub з прикладами розв'язання поставленої задачі за ключовими словами, відібраними у пунктах 1-3.

Ці етапи варто повторити пару разів, щоразу уточняючи або урізноманітнюючи пошукові запити.

У разі, якщо не вдалось знайти потрібний датасет або його знайдено, але він має не такий вигляд, як треба, варто створити власний датасет. Часто є сенс робити свій датасет, навіть, коли є різні існуючі, але до них ще треба щоразу застосовувати багато операцій передоброблення. А тоді застосовуєте їх, відбираєте усе, що треба, зберігаєте один раз в оптимальному вигляді і тоді вже тільки його й використовуєте в подальшому.

Під час збирання даних дуже важливо враховувати ті ознаки, які впливають чи потенційно можуть вплинути на цільову ознаку, і не враховувати ті, які точно не впливають. Наприклад, існує веб-сторінка, де колекціонують так звані «Spurious Correlations» (з англ. – «Хибні кореляції»), тобто – закономірності, між якими немає фізичного чи семантичного зв'язку, але статистичний аналіз показує, що кореляція є (рис. 1.5).

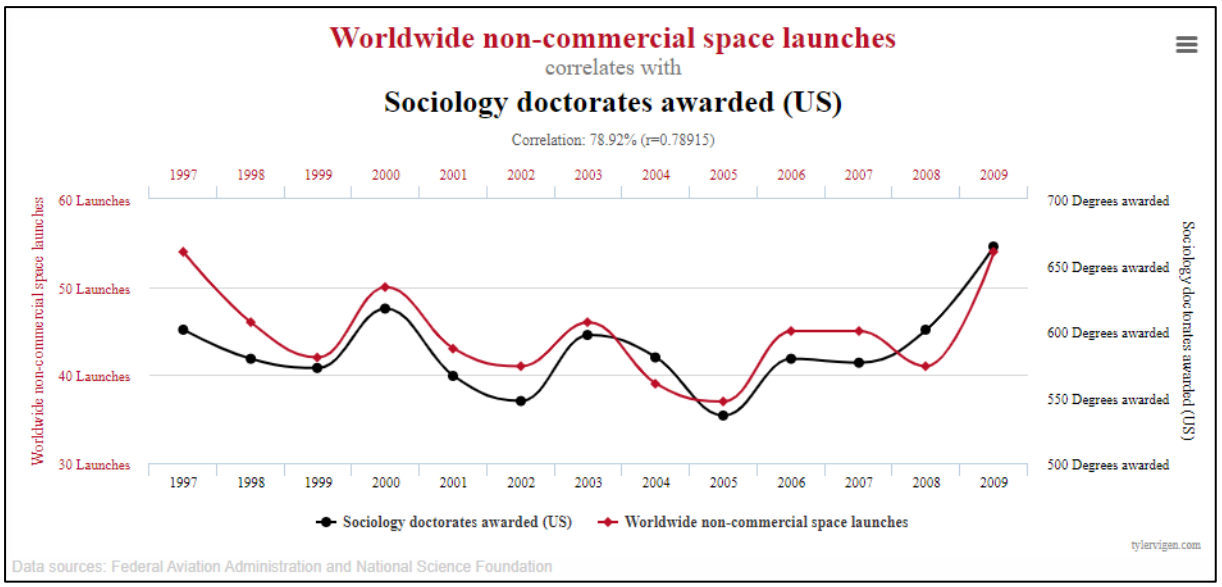


Рисунок 1.5 – Кореляція на 0,79 кількості некомерційних пусків космічних ракет та захищених докторських ступенів у сфері соціології на веб-сторінці «Spurious Correlations»

У Додатку Б наведено алгоритм та рекомендації побудови власного датасету у середовищі Kaggle. Також, можна створювати датасети у вигляді файлів на локальному комп'ютері (для використання у Python-програмах на цьому ж комп'ютері), у GitHub, на своєму Google-диску, у платному сервісі Amazon S3 тощо.

Інформацію для датасетів можна брати з різних джерел:

- збирати самостійно чи за участі помічників;

- з реальних інформаційно-вимірювальних та інформаційних систем, у т.ч. на основі Інтернету речей;
- використовувати відкриті дані, у т.ч. з веб-порталів і сервісів, соцмереж, датасетів Kaggle, GitHub та ін., найбільш зручно – користуватись даними через API.

Слід зазначити, що, щоразу, перед використанням інформації з певного датасету чи джерела, спочатку слід уважно вивчити ліцензійні умови цього датасету та законодавчі вимоги щодо використання такого роду інформації і дотримуватись усіх вимог (не брати ці дані, або анонімізувати їх, цитувати першоджерело у той спосіб, як вимагається у ньому). Важливим є ще спосіб, в який ви плануєте поширювати свій датасет. Іноді автори дозволяють використовувати дані індивідуально, для власних потреб. Іноді, ви можете їх завантажити безкоштовно, як здобувач вищої освіти (слід лише зареєструватись з пошти з акаунту університету), але тоді ви не маєте права поширювати цей контент для інших користувачів.

Kaggle просить вказувати (хоча, це – не обов'язковий параметр) вид ліцензії датасету (це особливо актуально, якщо він створюється як публічний) – під час вибору типу, варто уточнити вид ліцензії першоджерел, щоб не було неузгодження. Як правило, для публічних датасетів використовується CC-ліцензія (англ. «Creative Commons»). Ще в Kaggle є популярною ліцензія Массачусетського технологічного інституту («MIT») та ін. Зазвичай, ChatGPT може допомогти відповісти на питання чи можна дані, отримані за такою-то ліцензією, поширювати за такою-то, і ще підкаже як саме слід правильно виконати усі вимоги цієї ліцензії.

Важливим аспектом в задачах машинного навчання є формування навчального, валідаційного та тестового датасетів. На навчальному датасеті навчаються, тренуються моделі. Його ще називають «трейновий» – від англ. «training». На валідаційному (від англ. – «validation») вибирається яка з моделей – оптимальна. А тестовий (англ. «testing») використовується вже для застосування тільки оптимальної моделі. Тестові датасети популярні у різних змаганнях. В реальних же задачах, усі дані, як правило, поділяються тільки на навчальні та валідаційні або – на навчальні і тестові, але тестовий відіграє роль валідаційного. Зазвичай, навчальний датасет містить від 70% до 90% загальних даних, відповідно, валідаційний: від 10% до 30%. Найбільш популярні варіанти: 75/25% чи 80/20%. Варіант 90/10% використовується, якщо загальна кількість даних – замала і модель не може навчитись на 70-85%, а варіант 70/30% (або, навіть, 60/40%), коли дані – дуже одноманітні і модель дає надвисоку точність у 100% збігів, навіть за 70/30%.

Зазвичай, валідаційні дані відбирають із загальних довільно випадково або «рандомно»), але для часових рядів, коли треба прогнозувати майбутнє, тоді валідаційними є останні значення ряду, щоб оптимальна модель найбільш точно відтворювала саме дані, розташовані в часі безпосередньо перед тестовими.

Для класичних задач машинного навчання, як правило, використовується крос-валідація, коли задається тільки відсоток даних, який слід відбирати у навчальні та валідаційні, та кількість крос-валідацій (англ. скорочено: «cv») та інші параметри (відсоток перекриття та ін.) (рис. 1.6).

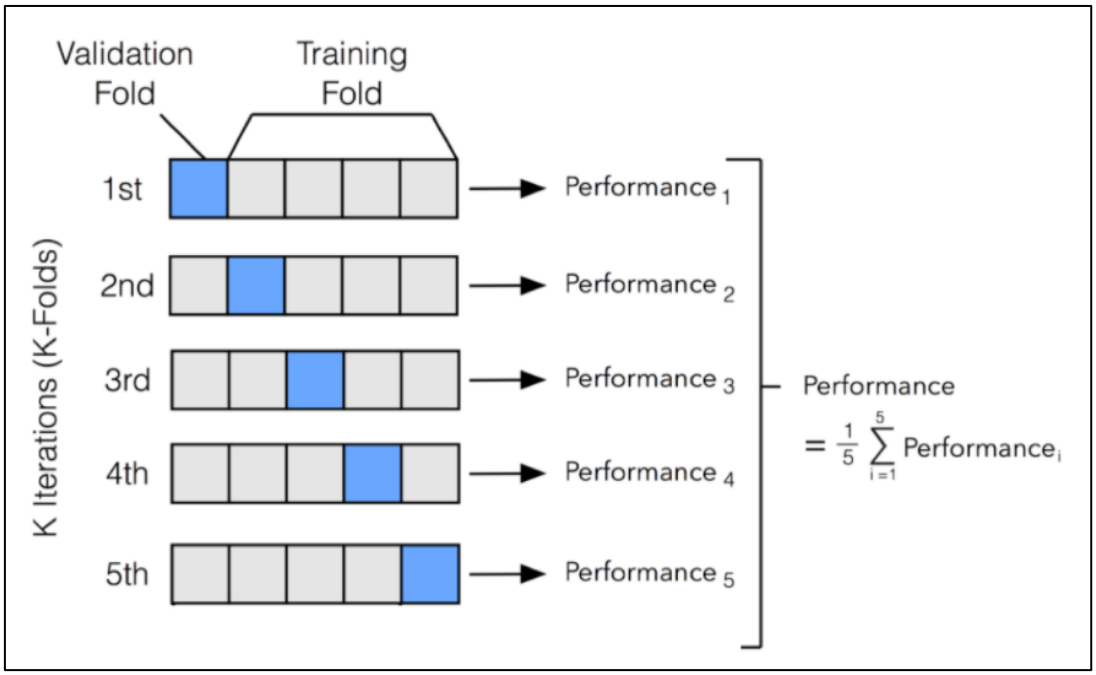


Рисунок 1.6 – Крос-валідація навчальних та валідаційних даних з cv=5, без перекриття даних (з опису у GitHub)

Тобто усі дані розбиваються на блоки, які по-черзі стають валідаційним датасетом, а решта є навчальними даними. Якщо задано перекриття даних, тоді кожне значення по декілька разів може бути у валідаційному датасеті. Такий спосіб дозволяє, в деяких випадках, суттєво підвищити точність моделі.

Ще важливо слідкувати, щоб у валідаційних та тестових даних були ті самі області значень ознак, що й – у навчальному, інакше модель може недонавчитись. Наприклад, якщо у навчальному датасеті є вживані авто тільки вартістю від 10 до 20 тис. дол., а у валідаційному та тестовому – від 300 тис. дол. до 2 млн. дол., то така модель навряд чи буде ефективною. Для цього важливо аналізувати та співставляти закони розподілу ознак цих датасетів, про що більш буде йтись у наступному розд. 2.

Датасет, як окремий комплекс файлів на диску чи у хмарному середовищі, не завжди є необхідним, якщо дані можна отримати через API (прикладний інтерфейс) чи просто – з веб-ресурсів. Для цього достатньо зробити запит у програмі, завантажити дані і почати обробляти. Наприклад, можна:

- завантажити веб-сторінку, вилучити з неї текст і далі його аналізувати (про це буде йтись у розділі 7);

- курс криптовалюти завантажити по API з криптовалютної біржи (про це буде у розділі 8);

- у Kaggle можна у своєму ноутбуці (Python-програмі) інший ноутбук використати як датасет і підтягнути його результати напряму, без окремого збереження у хмару як датасет.

У YouTube-каналі одного з авторів наведено практичні поради на прикладі як можна самому сформувати запит через API на основі аналізу вебсторінки заданого ресурсу, якщо параметри доступу не описано в документації, щоправда, повторити цей запит вже неможливо, оскільки портал РНБО України перестав надавати доступ до статистичних даних по коронавірусу, але загальна методологія є універсальною.

Однак, використання датасету є зручнішим, оскільки можна зберігати в одному місці різні версії даних, загальні, навчальні, валідаційні та тестові частини датасету, версії різними мовами та з різним кодуванням, їх опис, коментарі до них, програми для їх оброблення тощо. У Kaggle ще й зручнішим є те, що усі ноутбуки збираються у колекції саме в контексті датасетів. Тобто вибираєш датасет і там бачиш усі ноутбуки, які створені на його основі, свої чи чужі.

1.3 Визначення цільової ознаки, види задач і метрик машинного навчання. Уточнення постановки задачі

Після формування датасету слід уточнити постановку задачі чи задач, які можна розв'язати на його основі.

Для уточнення постановки задачі важливо, щоб замовник (ним може бути й сам дослідник) спочатку описав задачу, як він її бачить, у 3-5 речень, додав датасет чи датасети з даними, стисло описав усі бажані очікувані результати. З урахуванням цього, слід чітко визначити тип цільової ознаки чи ознак, вибрати метрику чи метрики з тих, які можна використовувати для цього типу, визначити чи є якісь обмеження та які. Отже, слід отримати відповіді на такі питання:

1. Що є цільовою ознакою і якого вона типу?

Цільову ознаку ще часто називають її англійським варіантом «таргет» («target»). Неправильне розуміння того, що є таргетом, чи неправильне розуміння його типу чи природи призводить до неможливості коректно та якісно розв'язати задачу. Це питання, у свою чергу, поділяється на ряд підпитань.

1.1. Якими є допустимі значення цільової ознаки?

Слід знайти чи отримати приклади значень цільової ознаки, проаналізувати її розмірність, в ідеалі – з'ясувати повну множину можливих значень (якщо їх – мало) або – теоретично досяжні статистичні показники (мінімальне, максимальне, середнє значення, середньоквадратичне відхилення, квартилі тощо), якщо значень може бути багато.

1.2. Чи цільова ознака – єдина, чи її можна звести до такої, чи це – задача з багатьма цільовими ознаками одночасно?

Найбільш поширеним і простим є варіант з однією цільовою ознакою або – коли задачу можна звести до ряду окремих задач, де щоразу буде інша єдина цільова ознака і кожну таку вирішувати окремо. Більш складним варіантом є задачі, де слід визначати одразу декілька взаємопов'язаних цільових ознак і їх не можна розділити на окремі – для таких задач слід застосовувати спеціальні моделі, наприклад, нейронні мережі з багатьма виходами.

2. Задача є класифікаційною чи регресійною?

Класифікаційними є задачі машинного навчання, де здійснюється передбачення обмеженої кількості класів, на кожен з яких є кількість даних, достатня для вивчення закономірностей їх формування. Відповідно, *регресійними* є задачі, де здійснюється передбачення довільних значень цільової ознаки, для вивчення закономірностей кожного діапазону даних яких може й не бути достатньо вхідних даних, а тоді вони просто передбачаються за моделлю, тобто – виявлені закономірності поширюються («регресують») на інші значення. Зауважимо, що не слід плутати регресійні моделі з регресійними задачами – це різна класифікація! Наприклад, для розв'язання класифікаційних задач часто застосовується модель логістичної регресії.

На практиці, задачі, де цільова ознака є дробовим числом, одразу відносять до регресійних, а де – цілим і це число є 2-10 або менше – до класифікаційних. В інших варіантах потрібне більш ретельне вивчення. Наприклад, задача прогнозування приросту кількості хворих на коронавірус за добу є регресійною, хоча цільова ознака містить тільки цілі числа, але, коли йдеться про десятки тисяч осіб за рядами з менше 1000 даних, то більшість значень будуть мати місце вперше і по них немає статистики, а отже, їх слід визначати регресійно. А задачі визначення чи схоже довільне фото більше на собаку чи кішку, або яка саме це арабська цифра? – це все класифікаційні задачі. Якщо задача формулюється для декількох цільових ознак одночасно, тоді вона є класифікаційною, тільки за умови, коли вона є класифікаційною за кожною з окремих цільових ознак (наприклад, кожна така ознака має лише 2-3 варіанти значень), у протилежному випадку, задача є регресійною.

Задачу з 100 класами і мільярдом даних можна розглядати як класифікаційну, а якщо даних – лише 200 штук, то її краще розв'язувати як регресійну, інакше модель недонавчиться.

Найбільш ефективно розв'язуються задачі з так званим «бінарним таргетом», тобто, коли цільова ознака приймає тільки 2 значення (0 або 1, True або False, хворий або здоровий, вижив пасажир на «Титаніку» чи загинув тощо). Тому, коли даних – замало, або коли не вдається добитися успіху з регресійними моделями, можна спробувати трансформувати задачу у класифікаційну і розв'язати хоча б її. Класичним прикладом є прогнозування не самих значень, наприклад, обсягів продаж, курсу валют, метеопараметрів чи ін., а чи буде в наступний крок приріст (значення 1), чи – ні (0)?

Важливість класифікації задачі за цільовою ознакою стане зрозумілою у розділі 4, де зазначається, що більшість назв моделей машинного навчання складаються з двох частин, перша з яких – власне назва моделі, а друга –

тип задачі («Classifier» або «Regressor»), наприклад: RandomForestClassifier та RandomForestRegressor.

3. Це – задача аналізу, передбачення (англ. «prediction») чи – прогнозування (англ. «forecasting»)?

Є задачі, де треба тільки проаналізувати наявні дані і результатом будуть різні цінні висновки та рекомендації. А є задачі, де треба здійснити передбачення (або прогнозування) результатів. Насправді, вони є пов'язаними:

- краще аналізувати не самі дані, а – модель, яка їм відповідає, а критерієм адекватності моделі є можливість здійснювати за нею передбачення частини даних по іншій частині;

- для навчання моделі, за якою слід здійснювати передбачення даних, потрібно спочатку провести гарний аналіз даних.

Однак, існують задачі, де ніяке передбачення не здійснюється, а тільки – аналіз (у підрозд. 2.4 охарактеризовано ряд таких прикладів у вигляді призових конкурсів змагань Kaggle, наприклад, аналіз результатів опитування датасайнтистів), тому є сенс виділити тип задач «аналіз» як окремий.

У загальному випадку, задачі прогнозування є підвидом задач передбачення, але у задачах прогнозування важливо враховувати залежність від часу та використовувати спеціальні моделі часових рядів. Задачі передбачення – більш поширені, але у них, навпаки, ознаку кожного моменту часу, якщо вона є, обов'язково слід видаляти, інакше модель, як-то кажуть, перенавчиться (або «заоверфітиться» – від англ. «overfitting»), тобто буде старанно передбачати факти, які мали місце строго в моменти часу у тренувальному датасеті, не буде узагальнюватись та аналізувати природу самого явища і не буде здатна коректно працювати з тестовими даними у раніше недосліджені моменти часу. Наприклад, ми випадково отримуємо фото і маємо визначити це кішка чи собака, тоді це – задача передбачення і в ній не варто враховувати час отримання фото. А якщо ми прогнозуємо курс валюти на ринку, показник якості довкілля, кількість машин на світлофорі, обсяг виробленої електроенергії сонячною панеллю з щохвилинним усередненням, очікуваний урожай на полі тощо, тоді це – задача прогнозування і час обов'язково слід враховувати. Більш докладно про аналіз та прогноз часових рядів буде викладено у розділі 8, а у розділах 4-7 більше буде йтись, переважно, про задачі передбачення.

4. У розвиток попереднього питання: яким є тип формалізації даних та яким є вид поставленої задачі? Від відповіді на ці питання суттєво залежатиме алгоритм розв'язання поставленої задачі, тобто вибір серед наступних варіантів:

- *таблиці даних*, які не є послідовностями даних у часі (не є часовими рядами), та класичні класифікаційні чи регресійні задачі багатофакторного передбачення, коли за певних вхідних ознак слід визначити одну чи декілька

цільових ознак (стовпців таблиці) – про універсальні моделі цього типу більше детально викладено у розділах 4, 5;

- *зображення чи відео*: задачі аналізу, розпізнавання та/або генерування зображень – задачі, в яких слід визначити до якого класу відносяться надані зображення (або відео як послідовність зображень), або знайти на них задані об'єкти чи проаналізувати за іншими критеріями – про це більш детально викладено у розд. 6;

- *текст*: задачі аналізу, оброблення та/або генерування природномовного тексту – задачі, в яких слід проаналізувати, класифікувати чи в інший спосіб обробити природномовний текст – про це більш детально викладено у розд. 7;

- *часові ряди даних*: задачі (як правило, регресійні), в яких слід прогнозувати задану цільову ознаку, як результат зміни іншої чи багатьох інших, або – цієї ж, але – у попередні моменти часу – про це більш детально викладено у розд. 8.

Звичайно, ця класифікація не охоплює усього розмаїття даних і постановок задач. Ще є *мовлення* як звукові сигнали; звуки, які не є мовленневими сигналами; ігрові алгоритми (шахи, шашки, гра го тощо) та інші, але даний посібник зосереджений саме на наведених вище видах задач, оскільки саме вони зараз найбільш широко вирішуються з використанням технологій машинного навчання та інтелектуального аналізу даних і мають не дуже високий поріг щодо спеціальних знань. Для інших видів задач необхідним є значний обсяг спеціальних знань у предметній області, у сфері інженерії даних, програмування, теорії сигналів тощо, це має бути матеріалом окремого посібника по кожному з них. Хоча, деяка інформація щодо цих задач теж буде наведена: рекурентні нейронні мережі, які часто використовуються для розв'язання задач, пов'язаних з природною мовою; розв'язання задач машинного навчання з підкріпленням, які використовуються для ігрових алгоритмів (Alpha Zero та ін.), тощо.

Більш коротко це питання звучить так: таблиці, зображення, відео, текст, ряди чи інше?

Тепер розглянемо види метрик для перевірки оптимуму цільової ознаки.

Метрикою в машинному навчанні називають критерій оптимізації, якому слід доставити оптимальне значення на різних етапах оброблення даних (кластеризації, побудови моделей, класифікації даних та ін.).

Для задач передбачення найбільш популярними є такі (більше детально та з формулами див. про це у підрозд. 4.3):

- для класифікаційних задач: «accuracy_score», ROC-AUC, якщо датасет незбалансований, тобто якийсь клас суттєво переважає, тоді – «F1_score» або «F2_score»;

- для регресійних задач: «r2_score», «MAE», «MAPE», «SMAPE»;

- для задач кластеризації: «siluette_score» та ін.

У бібліотеці sklearn є й багато інших – див. у документації та в її описі «User Guide».

В наступних розділах будуть згадуватись й інші метрики в контексті задач, які будуть там описуватись.

Після визначення цільової ознаки, виду задачі (типу даних) та метрики варто здійснити уточнення постановки задачі. У Додатку В наведено приклади таких постановок задач як реальних, так і з призових конкурсів Kaggle.

На рис. 1.7 наведена інфографіка алгоритму постановки задачі у системі координат *S(I)*.

Рисунок 1.7 – Інфографіка алгоритму постановки задачі машинного навчання

1.4 Узагальнені алгоритми розв'язання задач машинного навчання та ІАД та ІТ-інфраструктура для їх реалізації

1.4.1 Узагальнений алгоритм розв'язання задачі машинного навчання інтелектуальних моделей

Для розв'язання задачі машинного навчання інтелектуальної моделі, в загальному випадку, використовується такий алгоритм:

1. Зібрати дані. Побудувати датасет.
2. Здійснити передоброблення та очищення даних.
3. Провести розвідувальний аналіз даних та визначити які моделі слід будувати далі. Якщо результат – задовільний і є зрозумілим які моделі слід будувати, тоді перейти до п.5, інакше – до п.4. Якщо у п.4 не вдається поліпшити дані, тоді – перейти до п.1 і знайти додаткові дані.

4. Здійснити інженерію ознак (FE – з англ. «Feature Engineering»). Після цього – повторити п.3, якщо нові ознаки містять необроблені дані, тоді – до п. 2.

5. Вибрати архітектуру перспективних моделей. Здійснити їх навчання (тюнінг – з англ. «tuning») на навчальному датасеті.

6. Провести діагностування моделей на валідаційному датасеті та проаналізувати викиди (аномалії). Якщо за результатами діагностування виявилось, що усі моделі пере- або недонавчились, тоді повторити п. 5 з іншими параметрами тюнінгу. Якщо це не допомогло після N спроб, тоді – перейти до п. 4, де – побудувати різні діаграми важливості ознак за побудованими моделями та проаналізувати ці ознаки. Якщо одна чи декілька моделей задовольняють вимоги щодо метрики, тоді перейти до п. 7.

7. Вибрати оптимальну модель (можливо, ансамбль моделей), яка задовольняє вимогам задачі щодо метрики та обмежень.

8. Застосувати оптимальну модель та здійснити передбачення (прогнозування), проаналізувати викиди та виявлені закономірності, забезпечити візуалізацію та відтворюваність отриманих результатів.

На рис. 1.8 наведена блок-схема цього алгоритму.

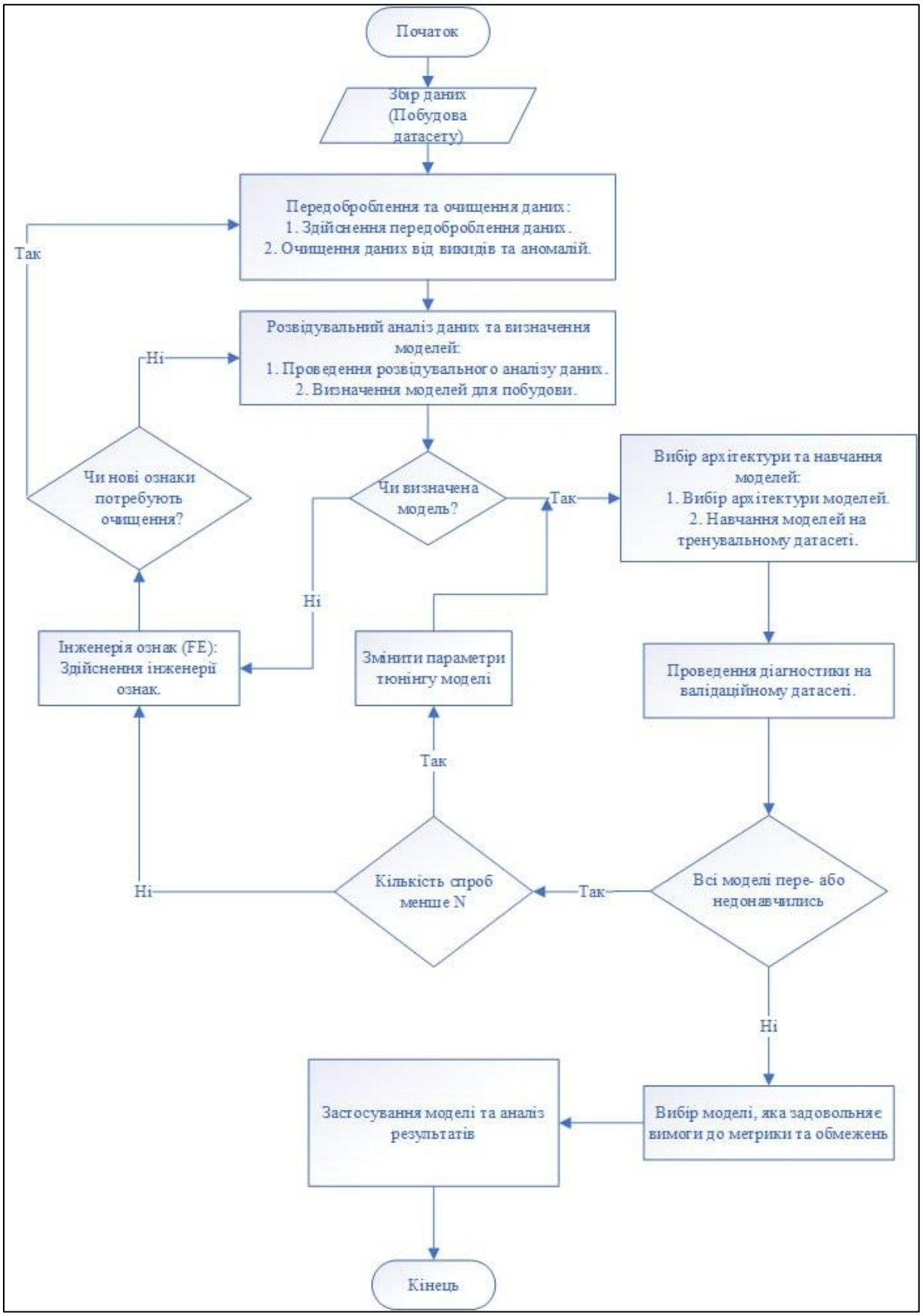


Рисунок 1.8 – Блок-схема узагальненого алгоритму з розв’язання задачі машинного навчання інтелектуальної моделі

На рис. 1.9 наведена інфографіка цього алгоритму у системі координат *S(I)*.

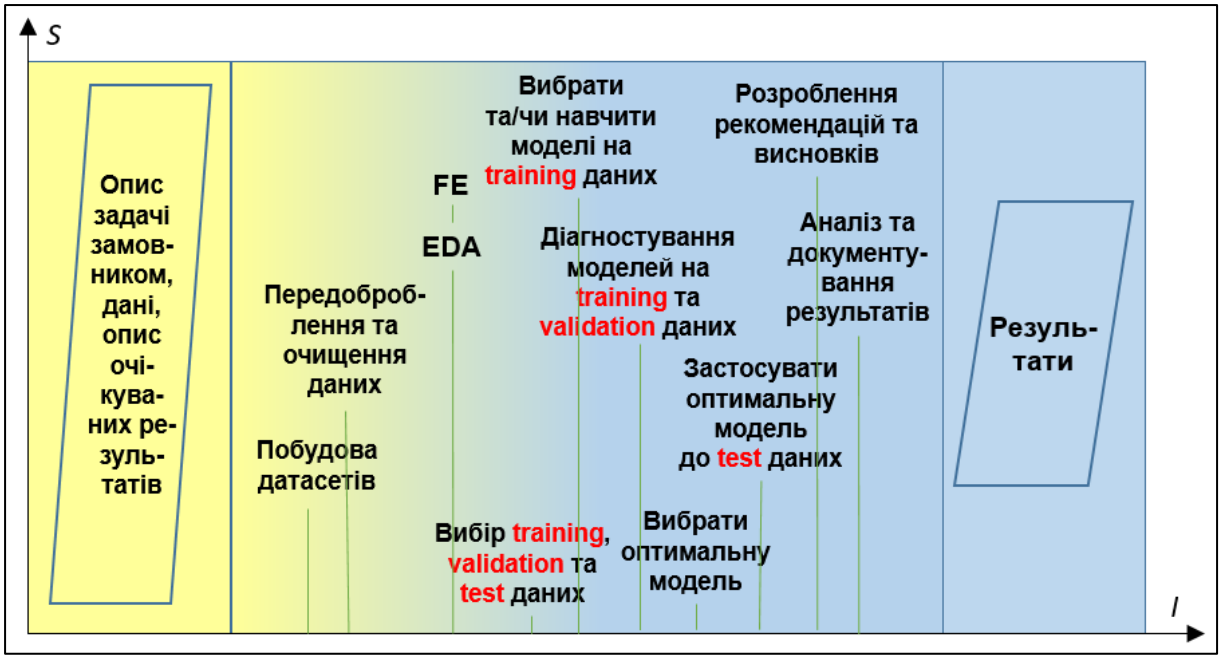


Рисунок 1.9 – Інфографіка узагальненого алгоритму з розв'язання задачі машинного навчання інтелектуальної моделі

Наступні розділи будуть присвячені етапам цього алгоритму.

1.4.2 Узагальнений алгоритм розв'язання задачі інтелектуального аналізу даних

Як було зазначено вище, на етапі інтелектуального аналізу даних (ІАД) використовуються вже передтреновані моделі машинного навчання. Крім того, як правило, на вході є не необроблені дані, а – готові датасети, до яких слід застосувати ІАД. Отже, варто навести алгоритм, який враховують ці відмінності.

У більшості випадків, є сенс здійснити передоброблення та очищення даних, оскільки моделі, як правило, працюють з числовими даними, а в ІАД даними можуть бути зображення, відео, текст та ін. Треба очистити і підготувати дані до вигляду, який є загально прийнятим в моделях відповідного класу задач. Потім провести їх розвідувальний аналіз даних та інженерію ознак. Далі вибрати релевантні моделі і трансформувати вхідні дані у формат, прийнятний у цих моделях. Як правило, етапи передоброблення, EDA, FE та трансформації даних проводяться неодноразово, щоб моделі запустились. Потім – застосувати моделі. Як правило, в задачах ІАД слід здійснювати післяоброблення результатів для отримання вищої якості аналізу. Останні етапи – такі самі: аналіз, документування, розроблення рекомендацій та висновків. На рис. 1.10 наведена інфографіка узагальненого

алгоритму розв’язання задачі інтелектуального аналізу з використанням передтренованих моделей машинного навчання, де синім кольором позначено нові блоки, яких немає на рис. 1.9.

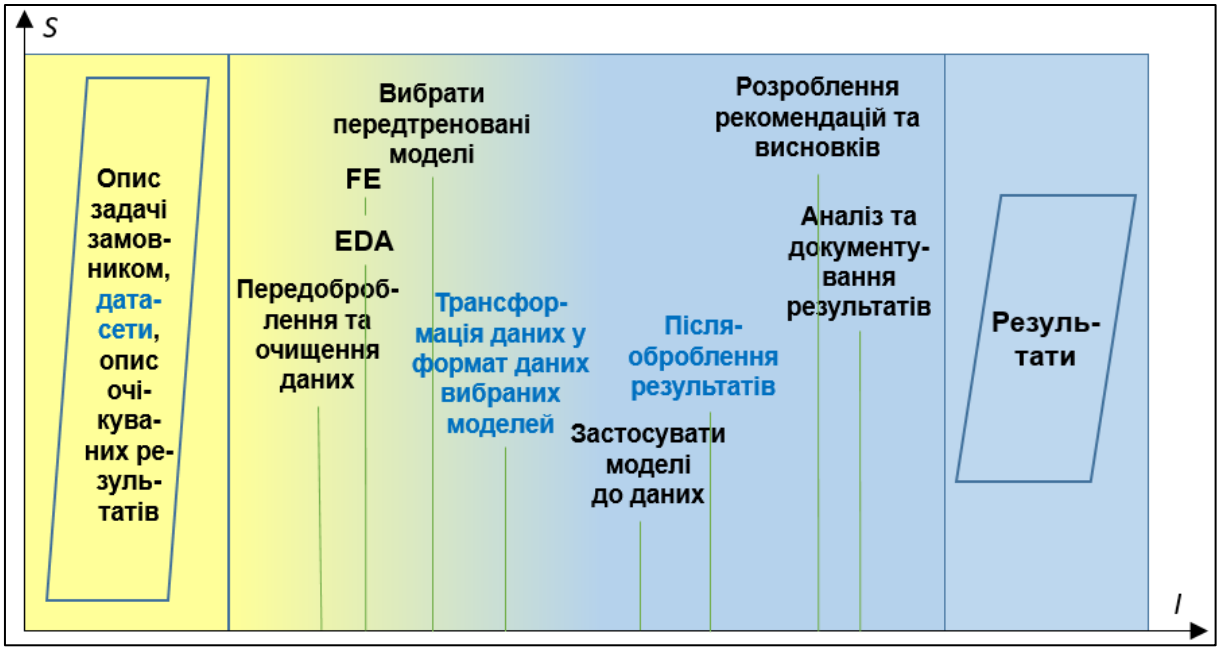


Рисунок 1.10 – Інфографіка узагальненого алгоритму розв’язання задачі інтелектуального аналізу з використанням передтренованих моделей машинного навчання (інтелектуальних моделей)

В загальному випадку рідко для якісного ІДА достатньо використати тільки передтреновані моделі. Частіше, вони використовуються тільки на якихось етапах, а потім для узагальнення результати їх застосування будуються нові моделі машинного навчання. Отже, для розв’язання реальної задачі слід поєднувати алгоритми рис. 1.9 і 1.10.

1.4.3 IT-інфраструктура для розв’язання задач машинного навчання та інтелектуального аналізу даних

Для застосування наведених вище алгоритмів розв’язання задач машинного навчання використовується різна IT-інфраструктура, яка містить такі складові:

- мови програмування: від вибору мови програмування залежить доступність бібліотек та фреймворків, які можна використовувати для розв'язання задач машинного навчання;

- середовища, веб-платформи (чи хмарні платформи) та сервіси: середовища для розроблення та розгортання моделей машинного навчання, які забезпечують доступ до обчислювальних ресурсів та даних;

- IDE (англ. «Integrated Development Environment») допомагає розробникам зручно писати і зберігати код та налагоджувати моделі машинного навчання (як правило, інтегрується з GitHub для збереження і контролю версій програм);

- бази даних та системи управління ними: використовуються для збереження та структурування даних та файлів, а також обробляють та кешують запити на отримання даних за заданими критеріями;

- фреймворки, пакети та бібліотеки – забезпечують доступ до функцій, класів, методів, які можна використовувати для розв'язання задач машинного навчання.

Вибір складових інфраструктури для розв'язання задач машинного навчання залежить від багатьох *критеріїв*:

- вимог замовника – іноді, забороняється використання хмарних ресурсів або, навпаки, забороняється зберігати копії даних на локальному комп'ютері;

- умов застосування – іноді, модель машинного навчання потрібна як елемент вже розгорнутої системи, наприклад системи Інтернету речей чи веб-сервісу, і повинна запускатись як окремий модуль чи в окремому контейнері;

- доступних фінансів – наприклад, для роботи з сервісами Amazon потрібні чималі кошти;

- знань і навичок програмістів, у т.ч. дата інженерів та датасайнтистів – для забезпечення якісної реалізації потрібні досвідчені кадри, хоча, під задачу зі значним фінансуванням може проводитись окремий найм відповідних працівників.

Можуть бути й інші критерії, у т.ч. вимоги чи обмеження.

Найбільш популярною *мовою програмування* для машинного навчання та інтелектуального аналізу даних, в наш час, є Python, оскільки вона є простою у вивченні та використанні, має велике та активне співтовариство розробників, і пропонує широкий вибір бібліотек та фреймворків для машинного навчання.

В минулому традиційно розв'яння задач машинного навчання, штучного інтелекту та інтелектуального аналізу даних здійснювали на мовах Prolog, R, у пакеті MATLAB та ін. [2, 3, 4, 5], але в наш час, особливо в IT-компаніях, які займаються створенням програм та готових для впровадження рішень, популярною є тільки мова Python. R раніше була популярною мовою для статистичного машинного навчання, оскільки вона пропонує потужні функції для аналізу та візуалізації даних. На мову Python вже переорієнтувалась більшість розробників рішень на R, Prolog, у MATLAB та ін. Якраз, автори даного посібника раніше програмували на R, а також – у MATLAB.

Існує ряд *хмарних платформ та сервісів*, які можна використовувати для розробки та розгортання моделей машинного навчання. Деякі популярні середовища включають:

- Amazon Web Services (AWS) – надає широкий спектр послуг для машинного навчання, включаючи Amazon SageMaker, Amazon Lex тощо;

- Microsoft Azure Machine Learning – це платформа для машинного навчання, яка пропонує такі послуги, як навчання моделей, розгортання моделей, і управління моделями;

- Google Cloud AI Platform – це платформа для машинного навчання, яка пропонує такі послуги, як навчання моделей, розгортання моделей, і управління моделями;

- Colaboratory (або Google Colab) – безкоштовне середовище для розробки та розгортання моделей машинного навчання на Python чи R, яке доступне в браузері, є й безкоштовні можливості щодо отримання доступу до обчислювальних можливостей з використанням GPU (хоча на платних умовах ці можливості значно більші);

- Kaggle – веб-платформа датасайнтистів, яка пропонує конкурси, безкоштовний редактор коду та обчислювальні потужності й форум для обговорення проблем.

Існують платформи для використання на локальному комп'ютері, наприклад платформа WEKA, більш детально про її застосування описано у посібнику [8], але в наш час, вони вже є мало поширеними.

Найбільш популярні *IDE* для машинного навчання на Python чи R:

- PyCharm – це IDE для Python, розроблена JetBrains, є безкоштовною для здобувачів вищої освіти (якщо вони реєструються з пошти навчального закладу);

- RStudio – це IDE для R, розроблена RStudio;

- Visual Studio є потужнім, але складним середовищем розробки інтегрованим (IDE) від Microsoft;

- Anaconda – це безкоштовний дистрибутив Python та R, який включає багато популярних пакетів для наукових обчислень і машинного навчання, таких як NumPy, pandas, Scikit-learn, TensorFlow та PyTorch;

- Visual Studio Code (VS Code) – це легкий редактор коду від Microsoft, який підтримує Python;

- Jupyter Notebook – це популярне середовище для виконання коду в браузері на Python.

Даний посібник орієнтований на використання саме Jupyter Notebook (JN), як найбільш універсального середовища. Автори, з власного досвіду, знають, що його код у форматі .ipynb можна редагувати і в Jupyter Notebook Anaconda, і в Kaggle Editor, і в Amazon SageMaker, і в Google Colab.

Для зберігання інформації в задачах машинного навчання можуть використовуватись як добре структуровані *бази даних* (наприклад, реляційні чи ієрархічні), так і неструктуровані, які просто містять csv-файли, зображення, відео- чи аудіо-файли. Другий тип зустрічається частіше.

В наш час більшу популярність отримали хмарні неструктуровані бази даних:

- Amazon S3 (скорочення від "Simple Storage Service");

- Microsoft Azure Blob Storage (Microsoft);

- Google Cloud Storage;

- IBM Cloud Object Storage та ін.

Важливо зазначити, що такі бази даних використовують надбудови, які структурують дані і роблять можливим запити SQL до них. Наприклад, в AWS для цього використовується Amazon Athena.

Для локальних рішень більшу популярність набули реляційні бази даних та мова SQL для роботи з ними.

Існує ряд базових *фреймворків та бібліотек*, які можна використовувати для машинного навчання:

- Scikit-learn – це бібліотека машинного навчання, який пропонує широкий вибір алгоритмів машинного навчання, включаючи класифікацію, регресію, кластеризацію, зменшення розмірності та багато інших, буде більш детально розглянутий в усіх наступних розділах посібника;

- NumPy – це бібліотека для наукових обчислень, яка забезпечує доступ до високопродуктивних обчислень, роботи з масивами, матрицями, часто використовується для оброблення та підготовки даних;

- Pandas – це бібліотека для аналізу даних, яка забезпечує високошвидкісні операції оброблення структурованих даних, таких як датафрейми (таблиця) і серії (окремі стовпці таблиці);

- Matplotlib – це бібліотека для візуалізації даних, яка забезпечує побудову графіків та діаграм;

- TensorFlow (TF) (інтегрований з Keras) – це фреймворк Google для машинного навчання, який спеціалізується на глибокому навчанні;

- PyTorch – це фреймворк Facebook для машинного навчання, який також спеціалізується на глибокому навчанні (конкурує з TF).

Інші бібліотеки, а також, пакети (збірки взаємопов’язаних модулів), як правило, використовують функції, методи і класи цих базових бібліотек та фреймворків, наприклад, популярними є такі:

- seaborn – бібліотека для візуалізації даних на основі Matplotlib з якісною інфографікою;

- plotly – бібліотек для інтерактивної візуалізації даних просто у браузері;

- xgboost, lightgbm – бібліотеки високопродуктивних бустингових моделей;

- spaCy, NLTK – бібліотека для оброблення природної мови (NLP);;

- opencv – бібліотека для комп'ютерного бачення, яка використовується для оброблення зображень та відео.

У ноутбуках, згаданих у довідковому авторському ноутбуці «Data Science for tabular data: Advanced Techniques», описано багато прикладів розв’язання задач машинного навчання з використанням згаданих вище бібліотек та фреймворків на Python. Більш детально ці та інші приклади будуть розглянуті у наступних розділах.

Можливі теми практичних і лабораторних завдань

<u>Тема № 1.</u> Постановка задачі машинного навчання на прикладі реальних задач та задач конкурсів міжнародної платформи Data Science та штучного інтелекту Kaggle (або **«Аналіз прикладів постановки задач у змаганнях Kaggle та в реальних дослідженнях у сфері системного аналізу»**, або **«Узагальнена постановка задачі та побудова чи вибір датасету для неї»**, або **«Постановка задач аналізу та передбачення стану складних систем з метою їх оптимізації на прикладі реальних задач аспірантів та задач конкурсів і датасетів платформи Kaggle. Створення чи вибір датасетів для цих задач»**).

Метою заняття є ознайомлення з постановкою задач та інформаційними технологіями, які використовувались під час розв'язання задач конкурсів міжнародної платформи штучного інтелекту Kaggle або реальних задач на прикладі одного із датасетів Kaggle.

План заняття:

1. Вибрати змагання чи датасет Kaggle, в якому є хоча б один ноутбук, автором якого є експерт, майстер чи гросмейстер Kaggle.

2. Навести назву (основну і додаткову), веб-посилання та автора чи організацію-власник датасету чи даних конкурсу Kaggle.

3. Описати українською мовою склад даних таблиць (чи однієї основної таблиці) датасету (назви стовпців, за які роки). Варто навести графік(и) з ноутбуків, які ілюструють які саме дані є в датасеті. Якщо дані мають географічну прив'язку, навести мапу, яка ілюструє цю інформацію.

4. Охарактеризувати задачі, які можна розв'язати на основі цього датасету (чи із завдання змагання, чи із розділу «Task» датасету, чи придумати самостійно).

5. Вказати та охарактеризувати які Python-бібліотеки та/або інформаційні технології використовувались у ноутбуках змагання чи датасет Kaggle з найкращим рейтингом (чи з найвищими місцями у змаганні, чи з найбільшою кількістю голосів за ноутбук). Наприклад, бібліотека Plotly для побудови інтерактивних графіків, бібліотека Xgboost для побудови моделі, бібліотека Folium для побудови інтерактивної мапи, IT аналізу розпізнавання зображень на основі PyTorch тощо.

Приклади датасетів:

- ноутбук з посиланнями та описом датасетів проф. Мокіна В.Б. у Kaggle у сфері моніторингу якості води;
- інші публічні датасети проф. Мокіна В.Б. у Kaggle:
- популярний конкурс «Titanic - Machine Learning from Disaster» для новачків Kaggle у (важливо зазначити, що ноутбуки цього змагання не можна буде використовувати в лабораторних роботах № 2-8, оскільки правила змагання забороняють їх розшарювати викладачеві – їх можна використовувати лише за умови, якщо автори будуть робити одразу публічні ноутбуки, тобто доступні усім користувачам Інтернету);

- популярний датасет «Heart Disease UCI» Kaggle з даними про серцеві захворювання в лікарнях Клівленду (США).

Тема № 2.

Метою заняття є опанувати знання та навички для створення власного датасету та його зчитування з використанням програми на Python.

План заняття:

1. Уточнити постановку задачі, щоб зрозуміти які самі потрібні дані для її розв'язання.
2. Проаналізувати наявні публічні датасети в Kaggle , GitHub.
3. Створити власний датасет (див. поради у Додатку Б). Див. приклади.

Тема № 3.

Формування інтегрованого датасету для аналізу даних про стан системи з різних джерел (API, CSV-файли та ін.) на Python в IDE чи Jupyter Notebook.

Метою заняття є вивчення систем збереження даних та опанування навичок для зчитування даних із цих систем з використанням програми на Python.

План заняття:

1. Вивчити джерело даних (назва, автор, зміст, обсяг даних та їх опис).
2. Вивчити приклади як імпортувати дані на Python з таких джерел:
   - інформаційна чи IoT-система;
   - датасет Kaggle (формати CSV, JSON та ін.);
   - датасет GitHub;
   - веб-система з API.
3. Вивчити можливості та опанувати основні навички роботи із заданою IDE (PyCharm, VSCode або Spyder в Anaconda) чи оболонкою типу «Jupyter Notebook» (NB) (SageMaker Amazon, Google Colab, редактор ноутбуків Kaggle чи в Jupyter Notebook в Anaconda) для створення програм на Python.
4. Створити програму на Python в IDE/NB з п. 3 для завантаження даних з п.1 з використанням прийомів з п.2. Усі дані об'єднати в один чи декілька датафреймів.

Приклади NB-ноутбуків із прийомами щодо завантаження даних у різний спосіб у Kaggle, у т.ч. через API:

- 50 Tips: Data Science (tabular data) for beginner
- 50 Advanced Tips: Data Science for tabular data

Контрольні питання

1) Що включає в себе поняття науки про дані? Дайте коротке визначення.

2) Які етапи включає процес збирання інформації та побудови датасету для аналізу?

3) Які цільові ознаки можуть використовуватися в задачах машинного навчання? Наведіть приклади.

4) Які основні види задач машинного навчання існують? Дайте короткий опис кожного виду.

5) Які метрики використовуються для оцінки якості моделей машинного навчання? Наведіть приклади метрик.

6) Що включає в себе узагальнений алгоритм розв'язання задачі машинного навчання? Наведіть основні кроки.

7) Які етапи включає узагальнений алгоритм розв'язання задачі інтелектуального аналізу даних? Опишіть кожен етап.

8) Яка інфраструктура використовується для розв'язання задач машинного навчання та інтелектуального аналізу даних? Наведіть приклади інфраструктури.

9) Які переваги має використання тренувального, валідаційного і тестового датасетів у задачах машинного навчання?

10) Як визначити оптимальну архітектуру моделі машинного навчання?

2 ПЕРЕДОБРОБЛЕННЯ ТА РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ (EDA)

2.1 Перевірка, очищення та передоброблення даних (Data Preprocessing)

Досвід показує, що дані датасетів, у більшості випадків, потребують додаткового передоброблення, до того, як починати їх серйозно аналізувати:

- хіміки в результатах лабораторного аналізу пишуть не тільки числові значення показників, а й «сліди», «відс.», «відсутні», «-», «те саме», «>0.2», «<10» та ін. і це ще треба перетворити на число або на значення «np.nan» («np» – скорочена назва бібліотеки NumPy, «nan» – «не число» з англ. «Not a Number»);

- медики описують словами те, що знайшли, і це ще треба перетворити у слова з певної фіксованої множини слів чи на числа;

- у разі отримання даних після парсингу веб-сторінки, туди можуть потрапляти html-теги, коди спецсимволів, веб-адреси, смайлики, які рідко несуть корисне навантаження і потребують видалення, хоча, у разі аналізу настроїв, може бути необхідним перекодуванням смайликів в якийсь більш зручний для аналізу текст (див. приклади у функціях «remove_emoji», «remove_punctuations», «convert_abbrev_in_text» ноутбука);

- у разі отримання даних з pdf-файлу, туди може потрапити текст з колонтитулів (номери сторінок, назва розділу чи інша малоцінна інформація), який потребує видалення.

Саме тому вхідні дані до передоброблення часто називають «raw» – з англ. «сирі». Налаштування фільтрів та алгоритмів передоброблення залежить від специфіки даних та задач, які розв'язуються. Наприклад, потужні мовні моделі, навпаки, погіршують точність аналізу, якщо «псувати» передобробленням сирий текст – вони чудово його розуміють і без цього. Хоча це, не завжди так. Наприклад, у Kaggle у 2020 р. було змагання «Tweet Sentiment Extraction» з аналізу твітів (https://www.kaggle.com/competitions/tweet-sentiment-extraction), яке показало, що навіть потужні на той момент мовні моделі типу BERT не могли без передоброблення класифікувати настрої тексту на кшталт «GGGGGGGGOOOOOOOOOLLLLLLLLLL!!!!!!!!!!».

Досить широкий список варіантів операцій передоброблення наведений у статті [27]. Але варто зупинитись на таких важливих аспектах, які часто мають місце:

1. Неоптимальний обсяг оперативної пам'яті під датасет («float64» замість «int8», «str» замість «bool» та ін.). За замовчуванням, датасети зчитуються командою read_csv з типами даних float64 та object, що, по-перше, займає в пам'яті забагато місця (вони тому так і зчитують, щоб точно усе

помістилось), а по-друге, ускладнює подальше оброблення даних. У разі використання операцій бібліотеки pandas, а як правило, саме вони застосовуються до табличних даних, таблиця займає в пам'яті у 2-3 рази більше місця, ніж її справжній розмір, для прискорення обчислень. А отже, неоптимальний формат призводить до значних втрат у швидкодії, а іноді, й унеможливлює роботу з даними, якщо їх не можна завантажувати меншими батчами (з англ. «batch» – «партія»). Для уникнення такої проблеми варто ще до початку роботи з датасетом оптимізувати формат усіх ознак таблиці. Для цього є 2 способи:

- одразу при зчитуванні вказувати потрібні типи даних (рис. 1.9) (варто пам'ятати, що, у разі, якщо там є np.nan, тоді замість «int» слід вказувати «Int64» з відповідною кількістю бітів) – див. приклад у пораді «Tip 2.4» в [10]);

- якщо наперед невідомо формати даних, тоді можна спочатку зчитати дані, потім логічні типи змінити на «bool», а потім скористатись порадою «Tip 5.1» в [11], яка автоматично визначить оптимальний формат числових ознак замість float64.

2. Трансформація форматів. Досить поширеною є практика подання даних як текстові: у вигляді слів чи окремих літер. Натомість, переважна більшість моделей машинного навчання вимагають тільки числові типи вхідних даних (за винятком деяких, на кшталт дерев рішень, оптимізованих під текстові дані, спеціалізованих мовних моделей та ін.), отже, важливо перетворити усі текстові дані з фіксованого списку значень на їхні номери у цьому списку. Для цього можна або скористатись простою заміною за допомогою словника, або використати спеціальну функцію sklearn.preprocessing.LabelEncoder (див. пораду «Tip 5.5» в [11]).

3. Усунення дублікатів. Наявність повних дублікатів рядків таблиці спотворює статистику, тому їх обов'язково слід виявляти і видаляти (див. приклад, усунення дублікату у медичному датасеті – «Tips 5.5» у ноутбуці [10]).

4. Очищення текстових даних. Для простих випадків використовується бібліотека re, а для більш складних – потужна бібліотека NLTK, більш докладно описана у роботах [28]). Також, цьому буде приділено більше уваги у розд. 6.

5. Заміна дуже малих чи дуже великих значень додатних np.inf та від'ємних (-np.inf) значень на np.nan, оскільки краще працювати лише з одним видом пропущених значень.

6. Імпутинг пропущених даних. Більшість моделей машинного навчання (окрім, наприклад моделі часових рядів Facebook Prophet та деяких інших винятків) вимагає відсутності пропущених (np.nan) значень. Якщо йдеться про числові ознаки, тоді способом їх усунення є «імпутинг» (з англ. «приписування»). Найбільш популярними є функції імпутинга (за зростанням складності): «SimpleImputer», «KNNImputer», «IterativeImputer» бібліотеки sklearn. Для таблиць ще є варіант заповнення середнього значення (або

медіани) по класу чи по кластеру (за методологією схожий на варіант функції ««KNNImputer»). Він буде більш детально розглянутий у наступному підрозділі.

Для табличних значень пропущені значення заміняють на якесь число, якого у таблиці точно немає, утворюючи новий клас. Наприклад, якщо усі числа є додатними, тоді пропущені заміняють на (-1) (див. пораду «Тір 4.5» в [10]).

7. Фільтрування аномальних значень. Ця задача більш детально буде розглянута в наступних підрозділах, оскільки для її вирішення варто скористатись результатами статистичного аналізу даних.

8. Перетворення дати. Часто дані містять дату, але після зчитування з csv-файлу чи в інший спосіб, дата, як правило має тип «str» (чи ще більш загальний: «object»). Однак, для застосування операцій з датою і часом, вони повинні бути у форматі datetime (є різні варіанти форматів). Для дати найбільш поширеним є формат «2023-10-30», який кодується в Python як «%Y-%m-%d» – див. приклад у пораді «Тір 5.6» в [10]).

2.2 Кластеризація та зменшення розмірності даних

Після виконання передоброблення даних часто проводять їх кластеризацію, або зменшення розмірності, або шукають асоціації, а вже потім аналізують більш ретельно результати цих операцій.

Кластеризація даних – це процес групування схожих об'єктів у класи або кластери на основі їхніх характеристик. Головна *мета кластеризації* – знайти приховані структури в даних та виділити групи об'єктів, які схожі одна на іншу, без заздалегідь відомого розподілу чи класифікації. Це – класична задача без вчителя.

Пошук *асоціацій* полягає в пошуку зв'язків та відносин між різними елементами у наборі даних. Основна *мета пошуку асоціації* – знайти правила асоціацій, які вказують на те, які елементи часто мають місце одночасно чи зі схожими характеристиками. Це може включати виявлення товарів, які часто купують разом; події, які мають місце у схожих умовах, тощо. Асоціація це - задача з учителем. Зазвичай, для цих задач використовують такі методи, як Apriori, Eclat, FP-growth та інші. Див., наприклад, Kaggle-ноутбук «Apriori Association Rules | Grocery Store».

Зменшення розмірності (англ. «dimensionality reduction») – це процес скорочення кількості даних або ознак (розмірності) в наборі даних. Мета цього процесу - зменшити кількість ознак, які слід враховувати під час аналізу даних, зберігаючи при цьому якнайбільше корисної інформації про структуру даних та зв'язки у них. Ця операція дозволяє підвищити ефективність та швидкість обчислень; зменшити ризик перенавчання, через зменшення зашумленості даних; покращити візуалізацію (наприклад, можна багатовимірний простір ознак звести до 2- чи 3-вимірного, який можна відо-

бразити та проаналізувати візуально). Популярними є метод головних компонент (англ. «Principal Component Analysis» – PCA), метод взаємного найближчого сусіда (англ. «t-Distributed Stochastic Neighbor Embedding» – «t-SNE»), UMAP, автоенкодери (нижче про це буде детальніше, оскільки t-SNE та UMAP є й методами кластеризації теж) тощо.

Найбільш часто в машинному навчанні застосовується саме операція кластеризації, тому розглянемо її більш детально.

Основні *функції кластеризації*, як етапу розвідувального аналізу даних, наступні:

– Виявлення структури даних для пошуку нових важливих закономірностей;

– Спрощення складних даних для зменшення розмірності даних чи для їх декомпозиції на менші датасети (наприклад, в ноутбуку (https://www.kaggle.com/code/vbmokin/fungi300-research) кластеризація 300 видів алергенів у пацієнтів дозволила здійснити декомпозицію на менші датасети в декілька ітерацій, після чого кожен кластер аналізувався окремо);

– Заповнення пропущених даних середньостатистичними по кластеру чи класу (приклад наведено у пораді «Tip 5.3» в [11]).

Усі методи кластеризації умовно можна поділити на такі види [6]:

- ітеративні (англ. «Partitioning methods»);
- ієрархічні (англ. «Hierarchical methods»);
- щільнісні (англ. «Density-based»);
- графові (англ. «Graph based methods»);
- кластеризація на основі моделі (англ. «Model based clustering»).

У посібнику [6] наведемо їх опис, дати створення та цікаві особливості. Але на практиці, у більшості випадків, застосовуються тільки деякі з них. Найбільш популярними *методами кластеризації* є такі:

1. *Метод k-середніх* (англ. «*Kmeans*»). Розділяє дані на k кластерів, де k – заздалегідь визначена кількість.

Алгоритм роботи: k вхідних даних випадковим чином вибираються як центроїди майбутніх кластерів. Далі визначається відстань від кожної точки до кожного із кластерів і тоді точка відноситься до того кластеру, до якого цей центроїд є найближчим. На наступній ітерації серед точок кожного кластеру вибирається інша точка, яка краще підходить на роль центроїда, і всі операції повторюються. І так до тих пір, поки відстань між старим і новим центроїдом не стане меншою певного порогу. Іншими критеріями ще може бути максимальна кількість ітерацій або значення інерції (англ. «inertia») – критерію, який дорівнює сумі квадратів відстаней між об'єктами та центроїдом їхнього кластера. Приклад реалізації цього методу наведено на рис. 2.2.

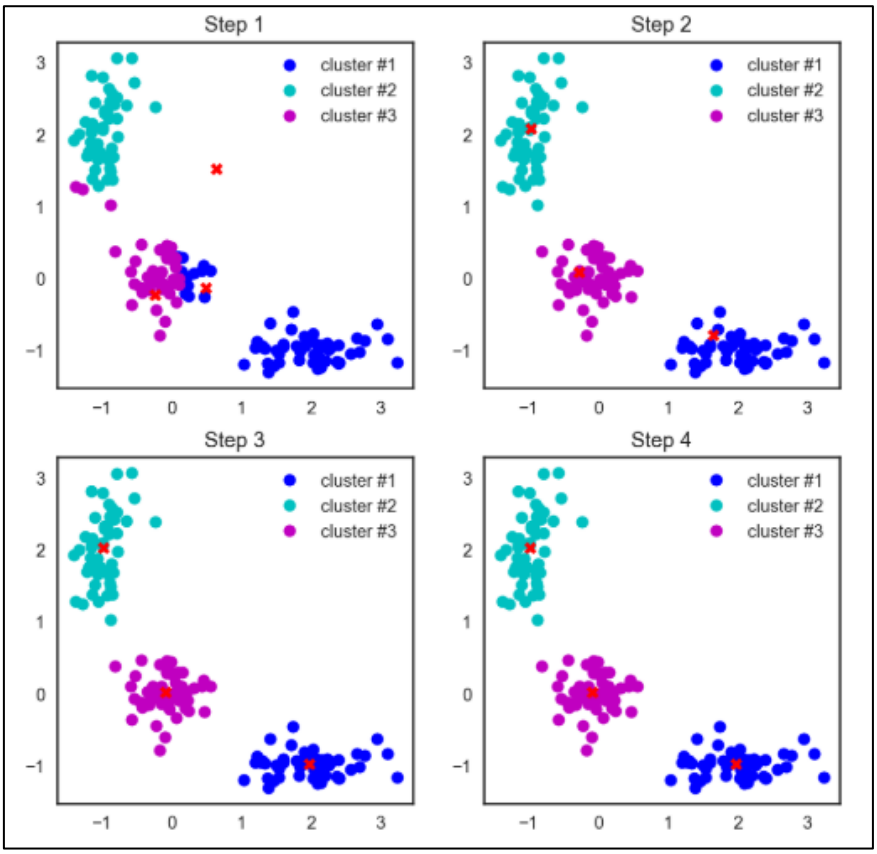


Рис. 2.2 Етапи методу кластеризації kmeans

Для використання методу слід задавати кількість кластерів. Але як її узнати? Для цього використовуються ряд підходів і метрик (в бібліотеці Sklearn їх біля 20). Найбільш популярним є використання критерію сілуету (англ. «Silhouette score» – sklearn.metrics.silhouette_score). Він є мірою того, наскільки точки в межах одного кластера є схожими між собою, у порівнянні з точками інших кластерів. Значення цього критерію s визначається для кожної точки окремо і має дві складові:

$$s = \frac{b-a}{\max(a,b)}, \tag{2.1}$$

де a – середня відстань між зразком і всіма іншими точками в тому ж класі; b – середня відстань між зразком і всіма іншими точками в наступному найближчому кластері.

Максимальне значення цього критерію відповідає оптимальній кількості кластерів.

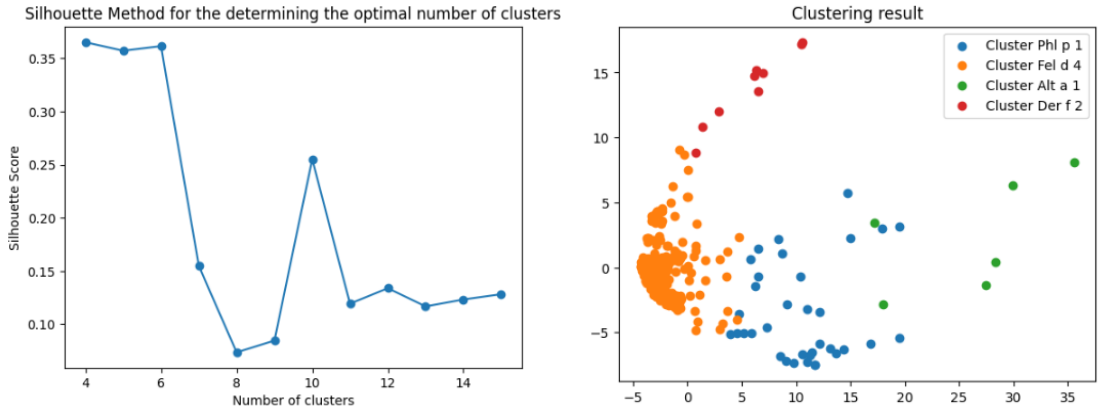
Якщо базові мітки істинності – невідомі, тоді можна використовувати, наприклад, один із двох індексів (ці та інші індекси більше детально описані у документації бібліотеки Sklearn щодо оцінки ефективності кластеризації):

- Індекс Калінскі-Харабаса (sklearn.metrics.calinski_harabasz_score) або його ще називають критерієм співвідношення дисперсій. Вищий бал Калінскі-Харабаса означає краще виокремлені кластери. Цей індекс — це відношення суми дисперсії між кластерами та дисперсії всередині кластерів для всіх кластерів;

- Індекс Девіса-Болдіна (sklearn.metrics.davies_bouldin_score). Нижчий індекс Девіса-Болдіна означає краще розділення між кластерами. Цей індекс означає середню «подібність» між кластерами, де подібність є мірою, яка порівнює відстань між кластерами з розміром самих кластерів. Нуль – це найменший можливий бал. Значення, ближчі до нуля, означають кращий розподіл.

- Також, популярними є метрики «Adjusted Rand Index» (ARI), «Adjusted Mutual Information» (AMI) тощо [7].

На рис. 2.3 наведено приклад аналізу чутливості пацієнтів до різних алергенів за реальними даними з авторського ноутбука. Як видно на рис. 2.3а, оптимальною кількістю кластерів є 4. І на рис. 2.3б видно, що кластеризація є задовільною.



а) б)

Рисунок 2.3 – Ілюстрація вибору оптимальної кількості кластерів у методі Kmeans з використанням Silhouette-критерію для аналізу чутливості пацієнтів до різних алергенів за реальними даними з авторського ноутбука: а) крива Silhouette-критерію, в залежності від кількості кластерів; б) результат застосування кластеризації методом Kmeans з 4-ма кластерами

Сучасна версія методу Kmeans, реалізованого у бібліотеці Sklearn, використовує удосконалений метод, який має назву «Kmeans++». Його основні відмінності наступні: 1) замість відстані між точкою і центроїдом визначається квадрат цієї відстані; 2) спершу вибирається тільки перший центроїд, а кожний наступний – з урахуванням ймовірності бути обраним як центроїд, пропорційно квадрату відстані від точки до найближчого вже обраного центроїда; 3) критерієм методу є мінімізація згаданої вище інерції.

KMeans є дуже витратним в обчислювальному плані і вимагає дуже багато пам'яті. Для великих даних або за обмежених обчислювальних ресурсів використовують інший варіант цього методу – MiniBatch. Основна ідея цього методу в тому, щоб застосовувати метод KMeans не до усіх даних, а тільки – для їх певної випадкової вибірки. Центроїди оновлюються після ко-

жної такої міні-партії. Цікаво, що цей метод може давати, іноді не гірші результати, ніж метод KMeans на усіх даних, але за значно менший час і може бути ефективним і для відносно малої кількості даних.

Для підвищення швидкодії методу рекомендовано здійснювати передоброблення даних з використанням методу аналізу головних компонентів (англ. «Principle Components Analysis» – PCA).

Є варіант tslearn.clustering.TimeSeriesKMeans – варіація методу Kmeans для часових рядів із бібліотеки tslearn. У ноутбуку наведено приклад кластеризації цим методом часових рядів курсу біля 80 криптовалют з більш як мільярдною капіталізацією (у дол. США) станом на квітень 2022 р.

2. Метод *DBSCAN (англ. «Density-Based Spatial Clustering of Applications with Noise»)* – кластеризація на основі густини даних за умов зашумленості. Кожна точка може бути центроїдом кластера («основною» точкою), якщо в певному радіусі від неї міститься задана мінімальна кількість точок (рис. 2.4). Алгоритм не завжди дає результат – треба підбирати параметри. Цінним є те, що метод дозволяє фільтрувати шум (аномалії) та самостійно визначає оптимальну кількість кластерів.

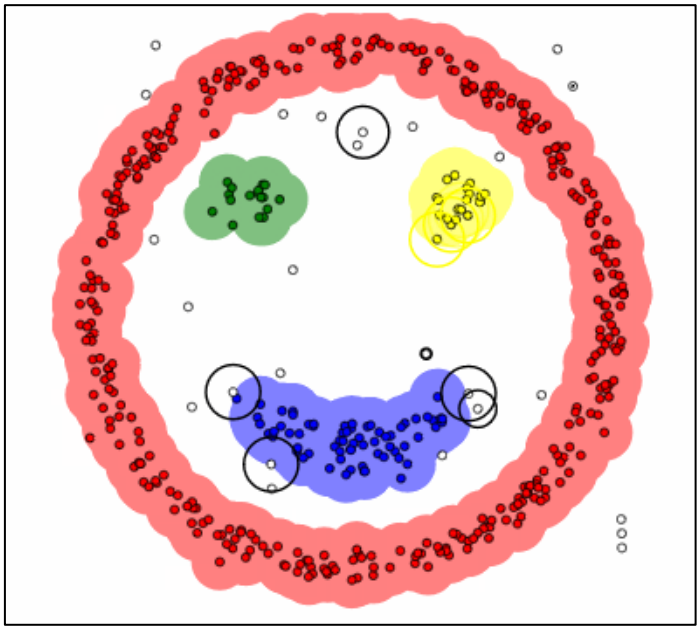


Рисунок 2.4 – Робота методу DBSCAN (gif-файл з ілюстрацією усіх ітерацій процесу кластеризації в динаміці доступний на сайті)

3. *Методи ієрархічної кластеризації.* Найбільш популярним із цих методів є *метод агломеративної кластеризації* (англ. «Agglomerative clustering»): розпочинає з окремих об'єктів і послідовно ітераційно попарно об'єднує їх у кластери, залежно від способу агрегування (за найменшою, за найбільшою, за середньою відстанню між ними чи ін.). Не потребує кількості кластерів. Після формування ієрархічного дерева пар точок слід задати рівень зрізу і метод одразу поверне кластери, які будуть відповідати цьому рівню (рис. 2.5).

```
from scipy.cluster import hierarchy
from scipy.spatial.distance import pdist
distance_mat = pdist(pca_df)
Z = hierarchy.linkage(distance_mat, 'single')
```

```
plt.figure(figsize=(10, 5))
dn = hierarchy.dendrogram(Z, color_threshold=3)
```

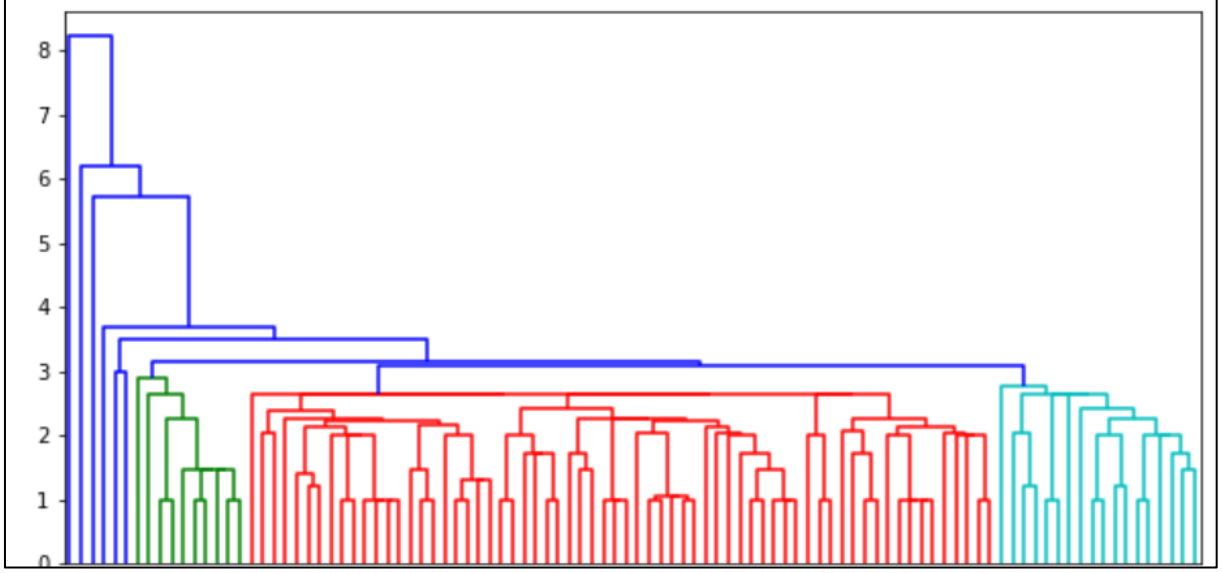


Рисунок 2.5 Метод агломеративної кластеризації

4. *Методи кластеризації і зменшення розмірності* «UMAP» (англ. «Uniform Manifold Approximation and Projection») і t-SNE («t-Distributed Stochastic Neighbor Embedding»). Метою обох методів є зменшення розмірності даних, зберігаючи при цьому важливі локальні структури та залежності між даними. UMAP використовує відстані на низькорозмірному просторі для знаходження подібності між точками, тоді як t-SNE використовує ймовірності наявності цієї подібності.

Яскравою демонстрацією можливостей методів кластеризації і зменшення розмірності «UMAP», «T-SNE» та «PCA» є сервіс «Embedding Projector» для інтерактивної візуалізації роботи цих методів на типових датасетах та на датасетах користувача.

Але це можна робити й програмно. Приклади наведено в авторських ноутбуках «MNIST Digits Original : 2D t-SNE with Rapids», «MNIST Original : 2D tSNE, 3D UMAP with RAPIDS», наприклад, там є 3D-подання результатів застосування методу UMAP (рис. 2.6).

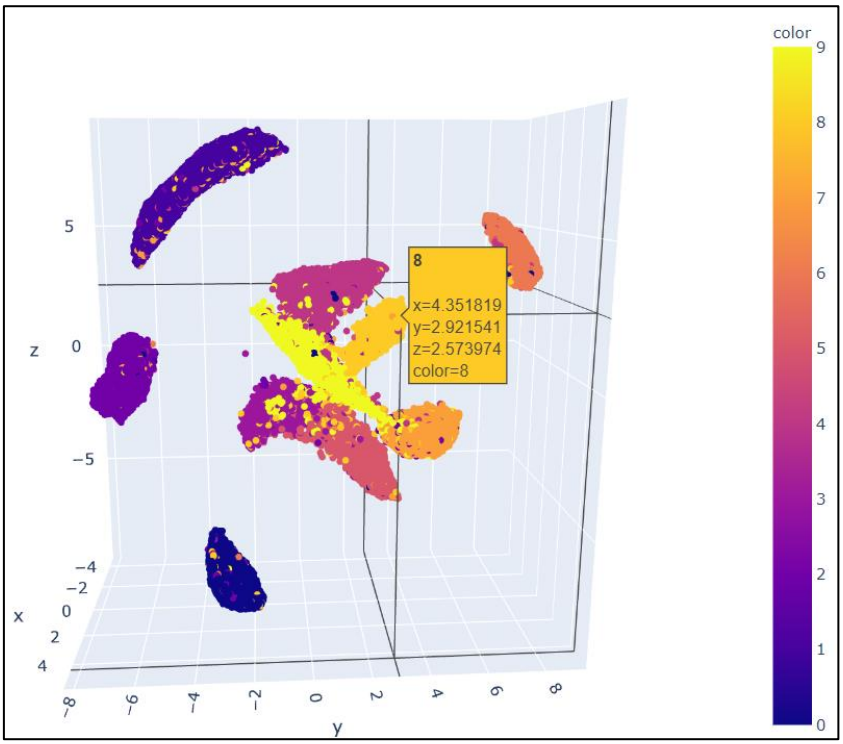


Рисунок 2.6 Результат застосування методу кластеризації UMAP до датасету рукописних арабських цифр MNIST з авторського ноутбуку

Бібліотека Sklearn містить гарну порівняльну таблицю різних методів кластеризації. У табл. 2.1 наведено основну інформацію з неї.

Таблиця 2.1 – Характеристики деяких методів кластеризації з документації

Назва методу	**Параметри**	**Розмір датасету**	**Кількість кластерів**	**Геометрія (метрика)**	**Особливості**
K-Means, MiniBatch	Кількість кластерів	Дуже великий (перевага MiniBatch)	Середня	Відстані між точками	рівномірний розмір кластера, плоска геометрія, індуктивний метод
Mean-shift	Пропускна здатність, не потребує кількості кластерів	Не дуже великий	Велика	Відстані між точками	нерівномірний розмір кластера, неплоска геометрія, індуктивний метод
Spectral clustering	Кількість кластерів	Середній	Мала	Відстань графіка (наприклад, графік найближчого сусіда)	Однакові розміри кластерів, неплоска геометрія, трансдуктивність

Ward hierarchical clustering	Кількість кластерів або поріг відстані	Великий	Велика	Відстані між точками	Можливе обмеження зв'язності, трансдуктивні
Agglomerative clustering	Кількість кластерів або порогове значення відстані, тип зв'язку, відстань	Великий	Велика	Будь-яка попарна відстань	Можливе обмеження зв'язності, неевклідові відстані, трансдуктивні
DBSCAN	Кількість сусідніх точок	Дуже великий	Середня	Відстані між найближчими точками	Неплоска геометрія, нерівномірні розміри кластерів, видалення викидів, трансдуктивність
BIRCH	Коефіцієнт розгалуження, поріг, необов'язковий глобальний кластеризатор	Великий	Велика	Евклідова відстань між точками	Видалення викидів, зменшення даних, індуктивний

У ноутбуку «Titanic Top 3% : cluster analysis» одним із авторів здійснена кластеризація пасажирів «Титаніка» 11-ма методами та здійснено їх порівняння зі значеннями цільової ознаки (вижив пасажир чи ні) (рис. 2.7).

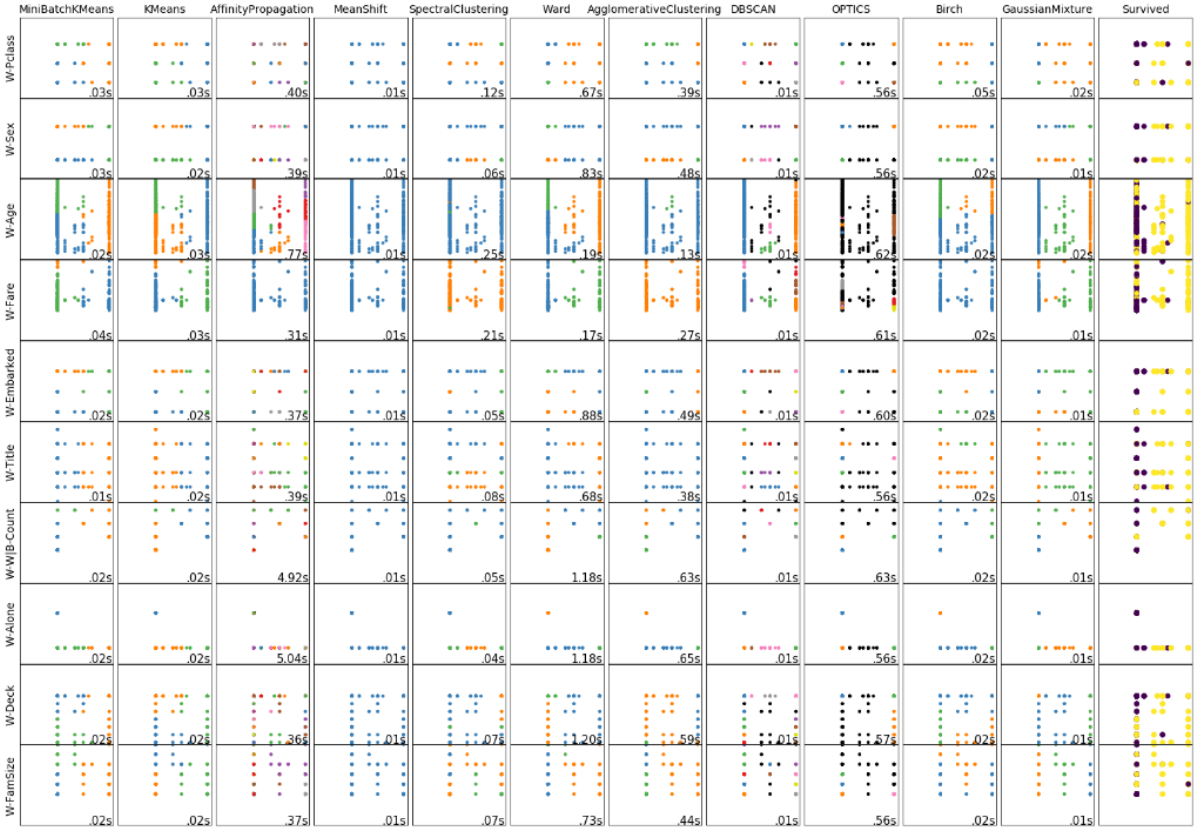


Рисунок 2.7 – Кластеризація пасажирів «Титаніка» 11-ма методами та здійснено їх порівняння зі значеннями цільової ознаки

Співставлення виявлених кластерів зі значенням таргета дозволяє виявити нові цікаві закономірності. Ноутбук «Titanic Top 3% : cluster analysis» містить універсальну функцію, яка одразу здійснює кластеризацію заданим методом із заданими параметрами (нові методи можна легко додати в clustering_algorithms).

В електронному посібнику [26] є деякі цікаві деталі щодо методів кластеризації, зменшення розмірності та пошуку асоціацій, у т.ч. з gif-візуалізацією цих операцій в динаміці.

На рис. 2.8 наведена інфографіка інструментарію, згаданого у підрозділах 2.1 та 2.2 у системі координат *S(I)*.

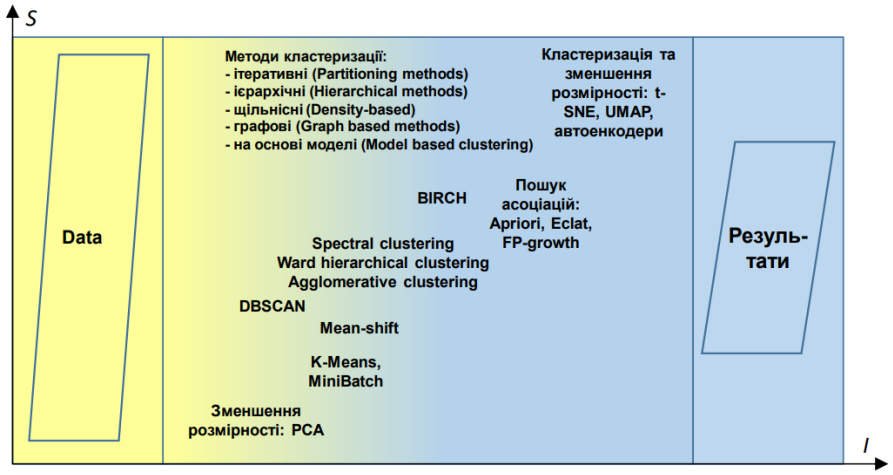


Рисунок 2.8 – Інфографіка передоброблення та кластеризації даних

2.3 Розвідувальний аналіз даних (Exploratory Data Analysis)

Розвідувальний аналіз даних (EDA) – це аналіз основних властивостей даних, знаходження в них загальних закономірностей, розподілів і аномалій з використанням відносно простих моделей.

Метою EDA є наступне:

- максимальне вивчення і «розуміння» даних;
- виявлення основних структур та систематизація даних;
- виявлення відхилень та аномалій (викидів);
- перевірка основних гіпотез;
- побудова та дослідження даних з використанням відносно простих моделей (регресій, дерев рішень).

Методи EDA застосовуються як до усіх даних, так і до їх кластерів та окремо до тренувальних, валідаційних і тестових даних:

- аналіз імовірнісних розподілів змінних;
- побудова й аналіз кореляційних матриць;
- факторний аналіз;
- дискримінантний аналіз;
- багатовимірне шкалювання та ін.

Залежно від особливостей задачі та результатів дослідження, EDA може містити наступні етапи (якщо не зазначене інше, то мається на увазі таблиця даних):

1. Підрахунок кількісних показників у датасеті та, за наявності, – в його тренувальній, валідаційній та тестовій складових (щодо виявлення та усунення таких проблем див. підрозд. 2.1):

- кількість рядків і стовпців усього та кількості пропущених значень у кожному стовпці (див. пораду «Tip 4.3» в [10]);
- виявлення рядків, де є значний відсоток пропущених значень у різних ознаках та, можливо, видалення таких рядків чи заповнення цих пропущених значень;
- формат і приклади значень у кожному стовпці – див. пораду «Tip 5.2» в [11];

2. Побудова різних графіків для аналізу закономірностей щодо значень кожної ознаки (feature) та їх комбінацій та ін. (бібліотеки matplotlib, seaborn та ін. – наприклад, див. ноутбуки [Plotting with pandas, matplotlib, and seaborn, Data-Visualization-Using-MATPLOTLIB-SEABORN-PLOTLY, Visualization Matplotlib vs Seaborn]).

3. Побудова описової статистики: по кожній ознаці (feature) аналізуються характеристики min, max, mean, std, кількість значень усього (counts); квантилі (квартилі), передусім P25(Q1), P50(Q2), P75(Q3), іноді – ще й P05, P10, P90, P95; скільки з них пропущених (missing) та скільки унікальних (uniques).

Також, можуть аналізуватись й квантилі (квартилі) значень по кожній ознаці.

4. Розширений первинний статистичний аналіз кожної ознаки та їх комбінацій. По кожній ознаці (features) слід побудувати закон розподілу та перевірити гіпотезу щодо його виду, в першу чергу, чи є він нормальним (гауссовим) [12, 13]. Іноді, закон розподілу можна нормалізувати, наприклад шляхом застосування методу Бокса-Кокса на основі логарифмування, див. про нього більше у підрозд. 8.2).

Як правило, будують закони розподілу для кожного класу окремо, наприклад дивись приклад на рис. 2.9 з ноутбуку) (рис. 2.9) та аналізують чи немає необхідності їх збалансувати. Балансування ознак (етап FE) описано у розд. 3.

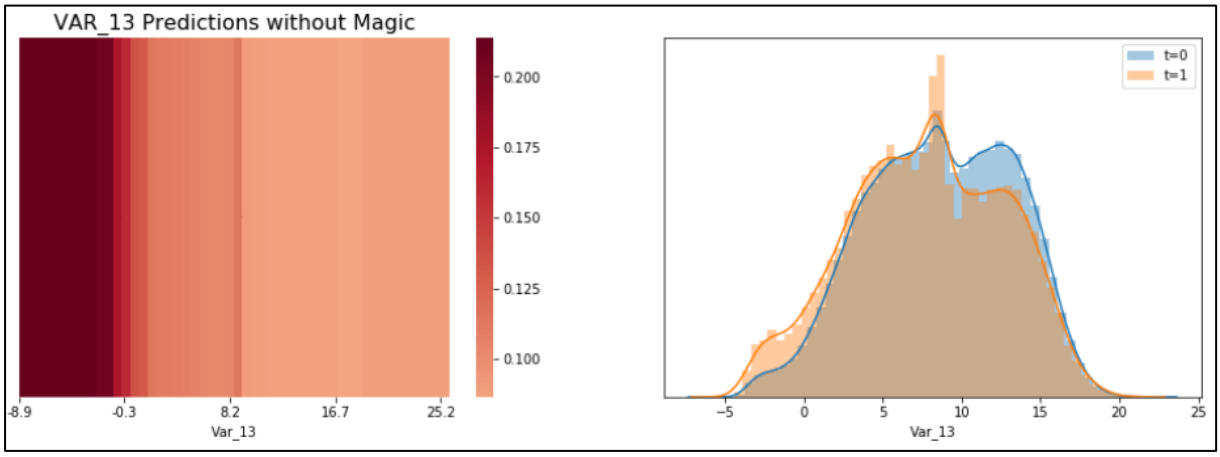


Рисунок 2.9 – Закони розподілу для різних значень таргета t=0 та t=1 з ноутбуку

5. Якщо датасет містить тренувальні, валідаційні і тестові дані чи хоча б 2 з цих 3-х варіантів, тоді порівнюються їх характеристики, передусім, закони розподілу – це дуже важливий етап, який рекомендується робити щоразу. Наприклад, на рис 2.10 наведено приклад порівняння законів розподілу, побудованих в одній системі координат, для тренувальних та тестових даних кожного фічера.

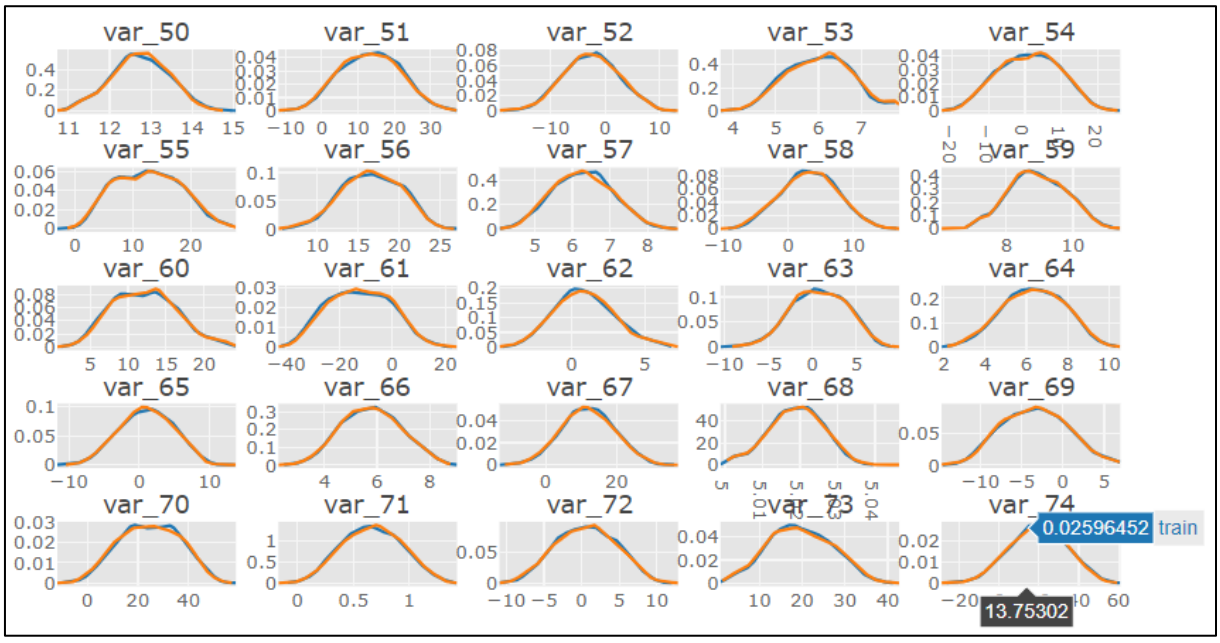


Рисунок 2.10 – Порівняння законів розподілу навчального (train) і тестового (test) датасетів [14]

Аналіз графіків на рис. 2.10 показує, що закони розподілу є нормальними і дуже схожими, отже датасети – гарні для подальшого оброблення. Для побудови законів розподілу використана функція **plotly.create_distplot** [14]. Також, можна використовувати інші функції, наприклад **seaborn.distplot**, як у ноутбуку.

6. Кореляційний аналіз (визначення наявності залежності та сили впливи між ознаками). Наприклад, аналіз взаємокореляції ознак та виявлення тих, що є найбільш залежними одні від інших. Результатом можуть бути як матриця чисел, так і графік типу «теплова карта» або гібридний варіант (див., наприклад [11]), коли в кожній комірці теплової карти вписується число. Із кожних двох сильно корельованих ознак, одну варто видалити, якщо ознак достатньо. Іноді, як наприклад, з аналізом ознак «Open», «High», «Low», «Close» курсу криптовалюти, то їх, як правило не видаляють, хоча вони й сильно корельовані, оскільки вони містять надто цінну інформацію, а таких ознак в цих датасетах – небагато [15]).

7. Регресійний аналіз – побудова та аналіз простих моделей (лінійної чи логістичної регресії, дерев рішень тощо) для дослідження певних закономірностей між ознаками для підтвердження наявності та визначення характеру і форми впливу одних показників на інші, особливо для тих, між якими на попередньому етапі було виявлено взаємозалежність – цей етап, скоріше, відноситься до інженерії ознак, але його часто відносять і до EDA.

8. Аналіз викидів та аномалій даних. Це можна робити трьома способами:

1) по квантилях, коли фільтруються значення ознак, де максимальне чи мінімальне значення в рази є більшими за P90 (або P95) чи меншими за P10 (або P05), відповідно, тоді відкидаються усі значення, більші P90 чи менші P10, відповідно – див. [15];

2) візуально – будуються графіки, як правило, з використанням інтерактивних графіків бібліотеки plotly і досліджуються аномалії по значенню чи по першій та/або другій зміні значень, вивчаються новини в Інтернеті чи дійсно це є аномалією, яка має якесь пояснення (наприклад, коли велика компанії чи уряд країни щось придбав або, навпаки, продав, чи ін., або спалах хвороби, або природна катастрофа чи ін.) і тоді це значення відноситься до аномальних (див. приклади в ноутбуці з прогнозування курсу криптовалюти чи – в ноутбуці приросту щоденної кількості хворих на коронавірус);

3) з використанням спеціальних бібліотек для часових рядів, про що буде більш детально написано у розділі 8.

9. Аналіз закономірностей даних методами кластеризації, факторного аналізу та зменшення розмірності – див. попередній підрозд. 2.2.

10. Аналіз варіативності ознак, тобто чи є достатня кількість різних варіантів значень кожної ознаки. Адже, ознаки, які є незмінними, слід видаляти, оскільки вони будуть заважати моделі навчатись.

11. Групування даних по певних ознаках та аналіз того, як кластеризуються інші ознаки відносно цієї. Більш детально про це дивись у попередньому підрозд. 2.2.

12. Для часових рядів: виявлення періодичності значень та ідентифікації періоду(ів) цих коливань, перевірка рядів на стаціонарність та на гетероскедастичність – про це буде більш детально у підрозд. 8.1.

13. Перевірка того чи є навчальний, валідаційний і тестовий датасети вибірками одного випадкового процесу, наприклад з використанням критерію Кохрена чи його аналогів [16].

14. Контроль розмірності кожної ознаки, у разі формування даних з декількох джерел. Наприклад, досвід показує, що може мати місце поєднання даних спостережень зі станцій, де атмосферний тиск вимірювався в кілопаскалях, з даними інших станцій у міліметрах ртутного стовпця (така проблема виникає, якщо брати дані з порталу SaveEcoBot). Якщо просто поєднати ці дані, то буде значна помилка. Їх треба аналізувати окремо, або пробувати одні дані перерахувати в інші, якщо є точна детермінована формула (для тиску якраз є).

Кінцевою метою EDA є відповіді на такі питання:

1. Чи готові дані для побудови моделей, чи вони потребують додаткового оброблення на іншому етапі (див. узагальнений алгоритм у підрозд. 1.4: інженерії ознак, передоброблення чи пошуку нових даних)? Якщо – ще потребують, тоді перехід на відповідний етап узагальненого алгоритму, інакше – див. наступне питання.

2. Які моделі варто будувати для розв’язання поставленої задачі, за якими метриками та з якими початковими значеннями параметрів і гіперпараметрів? Якщо є чіткі гіпотези (гарантій тут бути не може) щодо відповідей на ці питання, тоді – перехід на наступний етап узагальненого алгоритму – до побудови моделей.

Як було зазначено у підрозд. 1.4, як правило, EDA проводиться багато разів після кожних змін вхідних даних, чи – тоді, коли жодна гіпотеза щодо можливих оптимальних моделей не підтвердилась.

Для автоматизації EDA існує ряд спеціалізованих Python-бібліотек, які дозволяють суттєво прискорити цей аналіз і здійснювати не стільки програмування, скільки – дійсно аналіз автоматично побудованих графіків:

1. Найпростішим варіантом є функція «describe» бібліотеки pandas для побудови описової статистики (рис. 2.11). За замовчуванням, статистика будується для квантилів і квартилів P25(Q1), P50(Q2), P75(Q3), але, якщо в параметрах написати df.describe([.1, .9]), тоді в таблицю будуть додані ще й квантилі P10, P90 [17].

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.202211	0.523008	0.381594	32.204208
std	0.486592	0.836071	13.549459	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	21.000000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	26.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	36.750000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Рисунок 2.11 – Статистика функції describe бібліотеки pandas для даних конкурсу щодо пасажирів «Титаніка» з авторського ноутбуку

2. PandasProfiling (PP) [11] – це бібліотека для автоматизованого створення звітів та візуалізації для аналізу даних за допомогою бібліотеки Pandas. Все робиться однією командою ProfileReport, якій передається датафрейм з даними. Результат можна зберегти у html-формат і потім переглядати у зручний спосіб у браузері:

```
import pandas as pd
from pandas_profiling import ProfileReport
report = ProfileReport(data)
report.to_file('report.html')
```

Головний принцип PandasProfiling – це виведення статистики та різної інформації для кожної змінної окремо. Також є можливість на інтерактивному графіку аналізувати взаємозв'язок між довільними парами фічерів. Цінним є те, що на початку звіту наводяться загальні висновки щодо змінних і у різних кольорах наводиться різна текстова інформація: які фічери сильно корелюють, які містять дуже мало або дуже багато унікальних значень, які містять дуже багато пропущених значень та ін. (рис. 2.12).

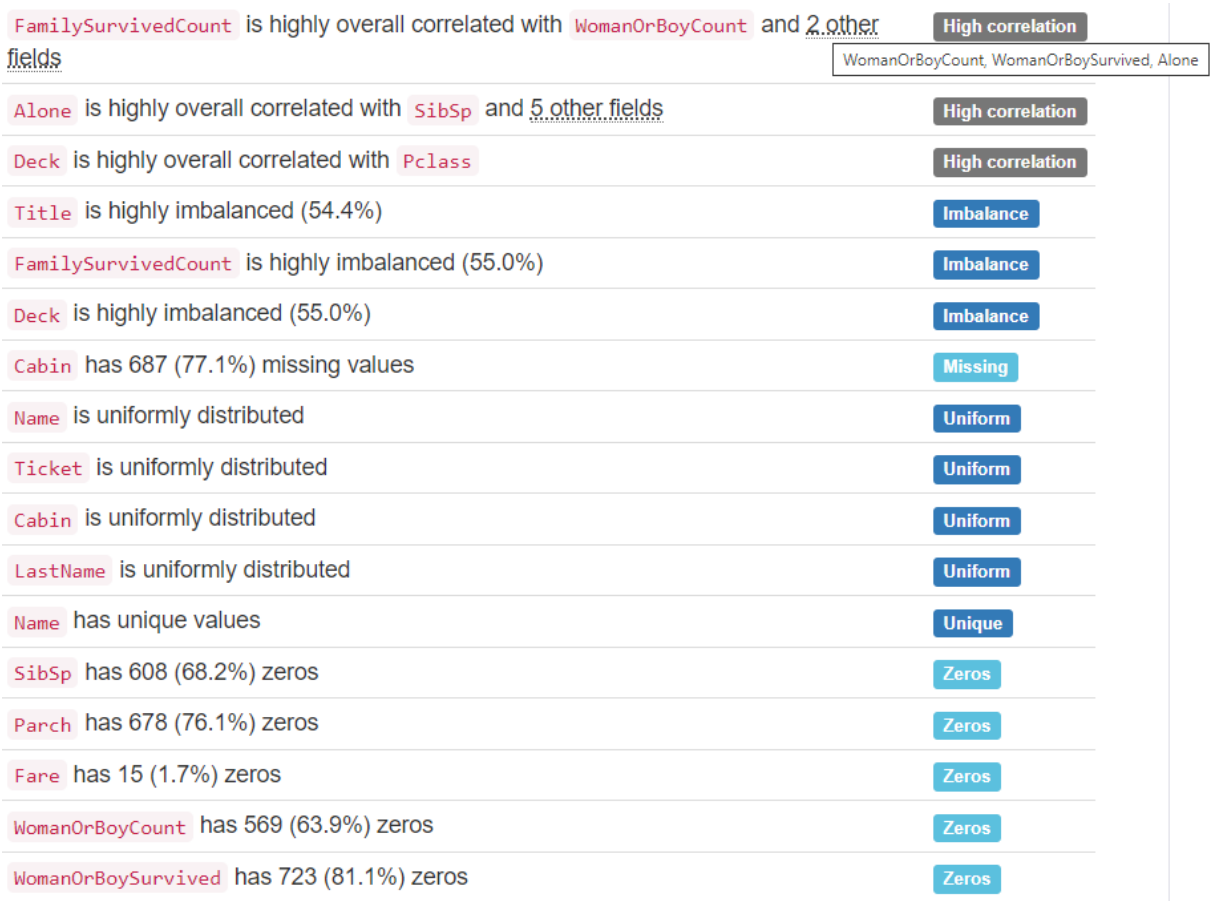


Рисунок 2.12 – Статистичні висновки у звіті PandasProfiling щодо фічерів тренувального датасету конкурсу щодо пасажирів «Титаніка» з авторського ноутбуку

3. AutoViz [11] – це бібліотека для автоматичної візуалізації даних у Python. AutoViz автоматично визначає тип графіка для кожної змінної в залежності від її характеристик. Наприклад, числові змінні можуть бути відображені у вигляді гістограм, діаграм розсіювання або лінійних графіків, а категоріальні – у вигляді кругових діаграм чи стовпчикових графіків – намагається надати корисні візуалізації для кожного типу даних. Може робити згруповані аналізи, наприклад, враховуючи залежності між змінними або розподіл значень змінних за певною категорією. Може генерувати інтерактивні графіки. На рис. 2.13 наведено приклад.

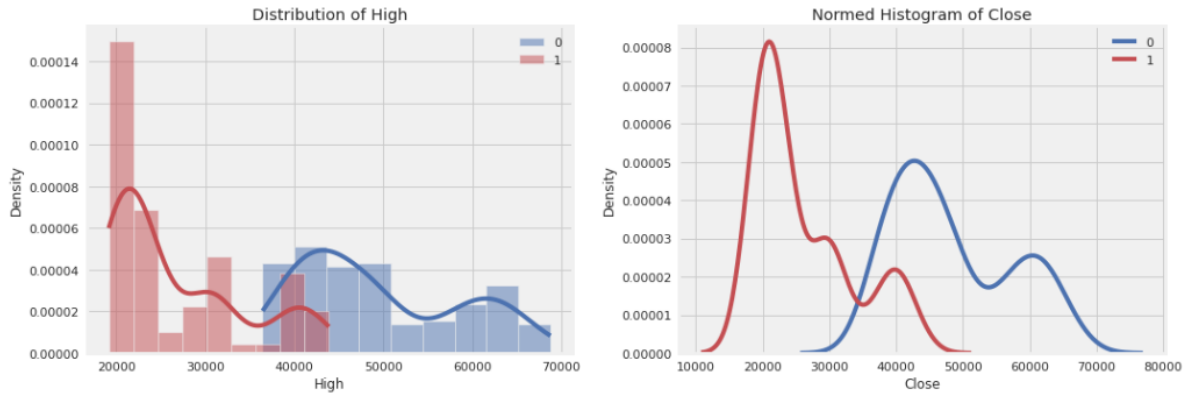


Рисунок 2.13 – Порівняння датасетів з курсом криптовалюти за різні дати, яким відповідають 2 класи: «0» – від 10.10.2021 р. до 06.04.2022 р., «1» – від 07.04.2022 р. до 03.10.2022 р. [18, підрозд. 6.3]

4. SweetViz [11] –це бібліотека для візуалізації та аналізу даних в середовищі Python. Вона автоматично будує гістограми для всіх числових та категоріальних змінних. є можливість аналізу зв'язку між числовою та категоріальною ознаками, будуючи графіки відносності. Це дозволяє визначити, як числові характеристики змінюються в залежності від категоріальних ознак. SweetViz надає зручні інтерфейси для порівняння двох різних датасетів. Це може бути корисно для аналізу різниці між наборами даних, наприклад, навчальним та тестовим, або даними, зібраними у різний час (рис. 2.14).

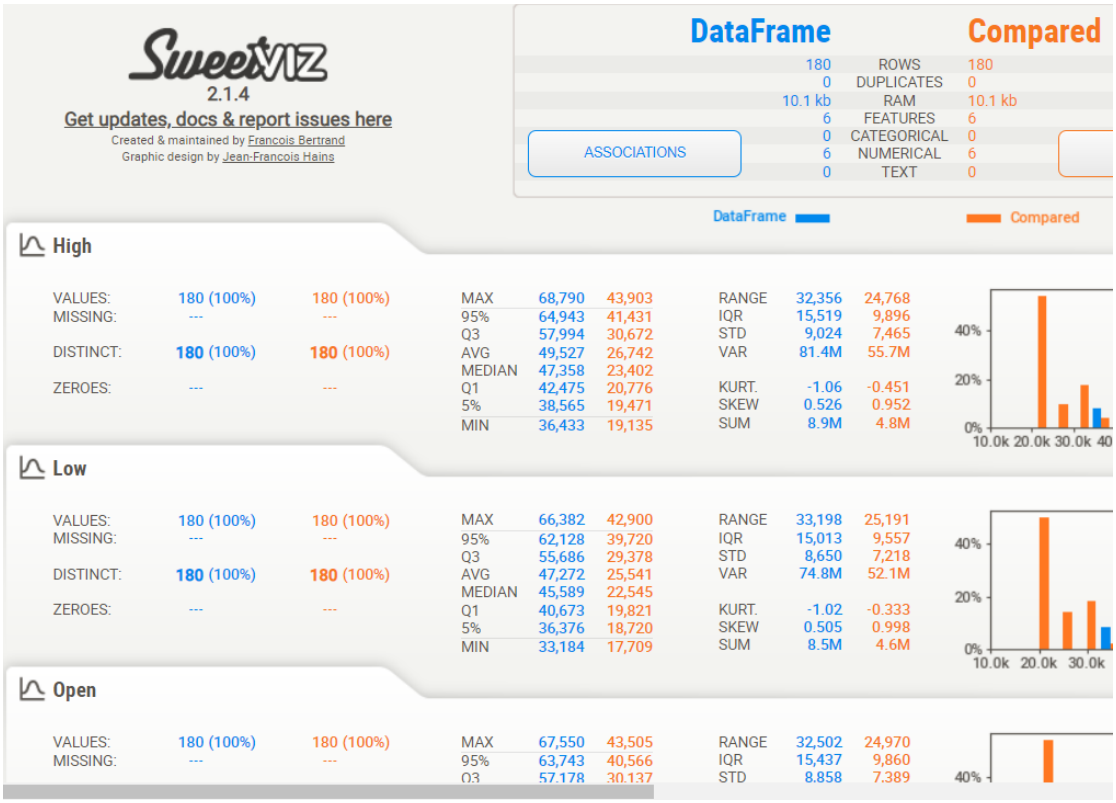


Рисунок 2.14 – Порівняння датасетів з курсом криптовалюти за різні дати, яким відповідають 2 класи: «Dataframe» – від 10.10.2021 р. до 06.04.2022 р., «Compared» – від 07.04.2022 р. до 03.10.2022 р. [18, підрозд. 6.2]

Узагальнені можливості останніх 3-х бібліотек див. у табл. 2.2.

Таблиці 2.2 – Порівняльна характеристика аналітичних можливостей EDA-бібліотек PandasProfiling, AutoViz, SweetViz

Бібліотека	Види графіків	Закономірності
PandasProfiling	Розподіл змінних (гістограма, ящиковий графік), матриця кореляцій, змінні-календарі, теплова карта кореляцій, пропущені значення, залежності між парами змінних, таблиці змінних.	Розподіл значень змінних, взаємозв'язок між змінними, кількість та відсоток пропущених значень, кореляція між змінними, статистика описового аналізу даних.
AutoViz	Гістограми, діаграми розсіювання, лінійні графіки, ящикові графіки, кругові діаграми, залежності між змінними.	Швидка автоматична візуалізація даних без великої кількості коду, автоматичний вибір типу графіка в залежності від характеристик даних.
SweetViz	Розподіл змінних, графіки розсіювання, порівняльні графіки, графіки відносності, таблиці з описом даних.	Взаємозв'язок між змінними, аналіз внутрішнього вигляду змінних, порівняння розподілів даних між різними наборами.

Крім цього, для побудови графіків за власним сценарієм можна використати універсальні бібліотеки, які мають вбудовані функції для EDA: Matplotlib, Seaborn, Plotly, Pandas. Більше детально їх основні поняття та команди описано у ноутбуках, згаданих в довідковому авторському ноутбуці «EDA for tabular data: Advanced Techniques». Деякі корисні команди описано у посібнику [9, розділи 1, 2].

2.4 Інтелектуальний розвідувальний аналіз даних, проведений призерами змагань Kaggle

У Kaggle є тип конкурсів «Analytics», де влаштовується призове змагання серед аналітиків і призерами оголошуються найбільш цікаві та оригінальні ноутбуки. Призовий фонд неодноразово дорівнював $100 тис. Конкурси можуть відрізнятись критеріями, але, зазвичай, вони формулюються як такі:

- Точність чи релевантність поставленій задачі. Обгрунтованість висновків. Якість візуалізації. Компонування і структурування ноутбука чи ноутбуків (іноді – це їх комплекс);

- Якість документації. Чи добре задокументовано ноутбук і додаткові джерела даних, щоб можна було зрозуміти що саме і як зроблено? Чи чітко цитуються джерела (як правило, в кінці ноутбуків подається список джерел, у т.ч. наукових статей)? Високоякісний аналіз повинен бути лаконічним і

зрозумілим на кожному кроці, щоб обґрунтування було легко зрозуміти, а процес був відтворюваним.

- Рівень висновків та рекомендацій. Обгрунтованість. Інноваційність. Чи дізнається читач щось нове, завдяки цій публікації? Або читач змушений думати про щось по-новому? Ноутбук має бути інформативним, спонукатиме до роздумів і водночас свіжим.

Ці критерії є цікавими в тому плані, що на них варто звертати увагу аспірантам під час виконання будь-яких своїх досліджень! Точніше, їх дотримання робить привабливим будь-яке дослідження. В наш час є популярним написання статей у Scopus / WoS і там вимагаються саме такі критерії (автори даного посібника мають такий досвід і такі статті у провідних журналах світу на основі публічних ноутбуків та власних датасетів у Kaggle).

Щороку Kaggle проводить опитування користувачів (в усіх країнах), а потім – конкурс типу «Kaggle ML & DS Survey», де стають доступними анонімізовані результати цього опитування (автори підбирають репрезентативні вибірки по кожній країні, якщо опитуваних – мало, тоді країну відносять до «Other»). Треба провести гарне грунтовне дослідження з гарною візуалізацією. Переможці отримують гарні грошові призи. Призовий фонд у 2019-2022 складав $30 тис.

Наприклад, у 2019 р. («2019 Kaggle ML & DS Survey») треба було розповісти історію даних про групи датасайнтистів, які пройшли опитування. Наприклад: студентки з обробки даних, які вивчають машинне навчання на магістерських програмах.

У 2019 р. 1-ше місце посів дослідник, який вигадав власний метод аналізу та візуаліції результатів з використанням теплової карти (рис. 2.15). За допомогою кожного набору таких карт він пояснював дії та робив цікаві висновки щодо різних груп дослідників. Перше місце йому явно дали за цю оригінальність та інноваційність.

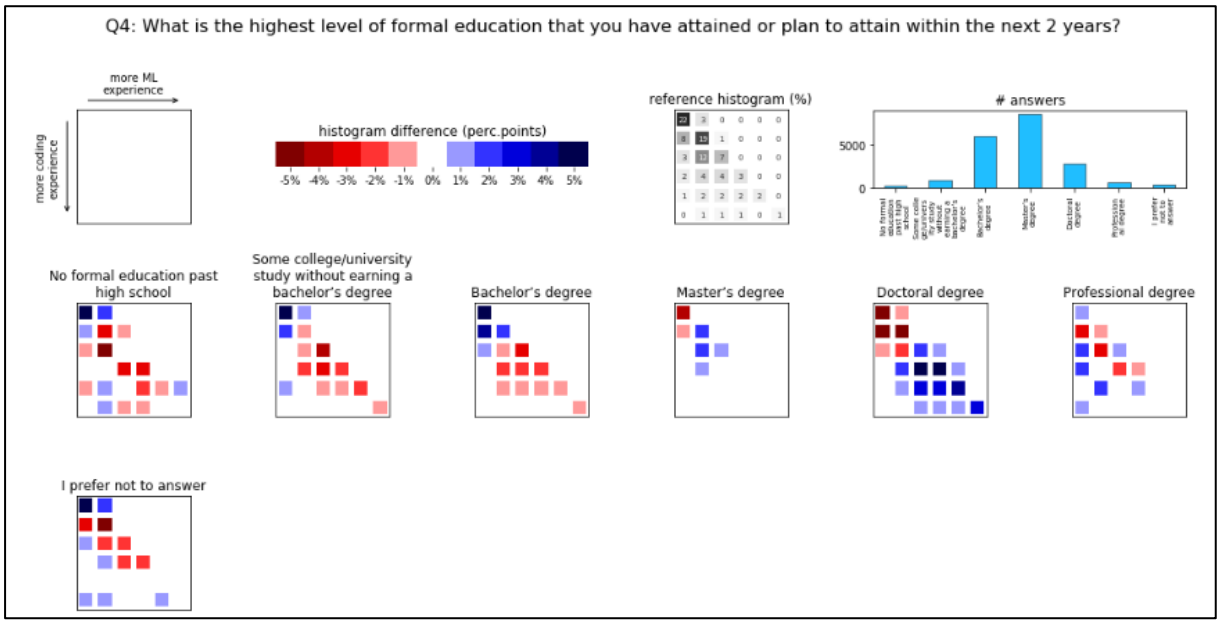


Рис. 2.15 – Інноваційна візуалізація статистичних даних з використанням теплових карт у ноутбуці «A story told through a heatmap», яка посіла 1 місце на конкурсі «2019 Kaggle ML & DS Survey»

У цьому конкурсі 2-ге місце посів ноутбук «Exploring PhD Community with Network Analysis», який вирішував дуже цікаву, як для читачів цього посібника, задачу – групу учасників з PhD у сфері Data Science. Його цікавило це питання, передусім, як для аспірантів, які хотіли б мати цей ступінь. Він досліджував різні аспекти. Кластеризував за різними критеріями. А найбільш цікавим є мережевий аналіз. На рис. 2.16 наведено мережевий аналіз навичок роботи з різними мовами, середовищами, пакетами та ін. Використаний метод графової кластеризації зі статті [19] (про це прямо написано у ноутбуці). Усі величини ранжуються за популярністю і це відображається щільністю зв'язків, розміром вершим, близькістю до ядра (так звана – «k-ядерна декомпозиція» – з англ. «k-core decomposition») та кольором: більш популярні навчики «центрального ядра» подано більш синім кольором, менш популярні – більш світлим блакитним, «периферійні» – у гамі червоного. Аналіз показує (звертаємо увагу, що це – лише 2019 р.!): найбільш популярними були методи DL та відповідні бібліотеки програмного забезпечення (конволюційні нейронні мережі, генеративні змагальні нейромережі (англ. «Generative Adversarial Networks»), байєсівські підходи, рекурентні нейронні мережі, TensorFlow, Keras тощо). MATLAB вже тоді ставав менш популярним.

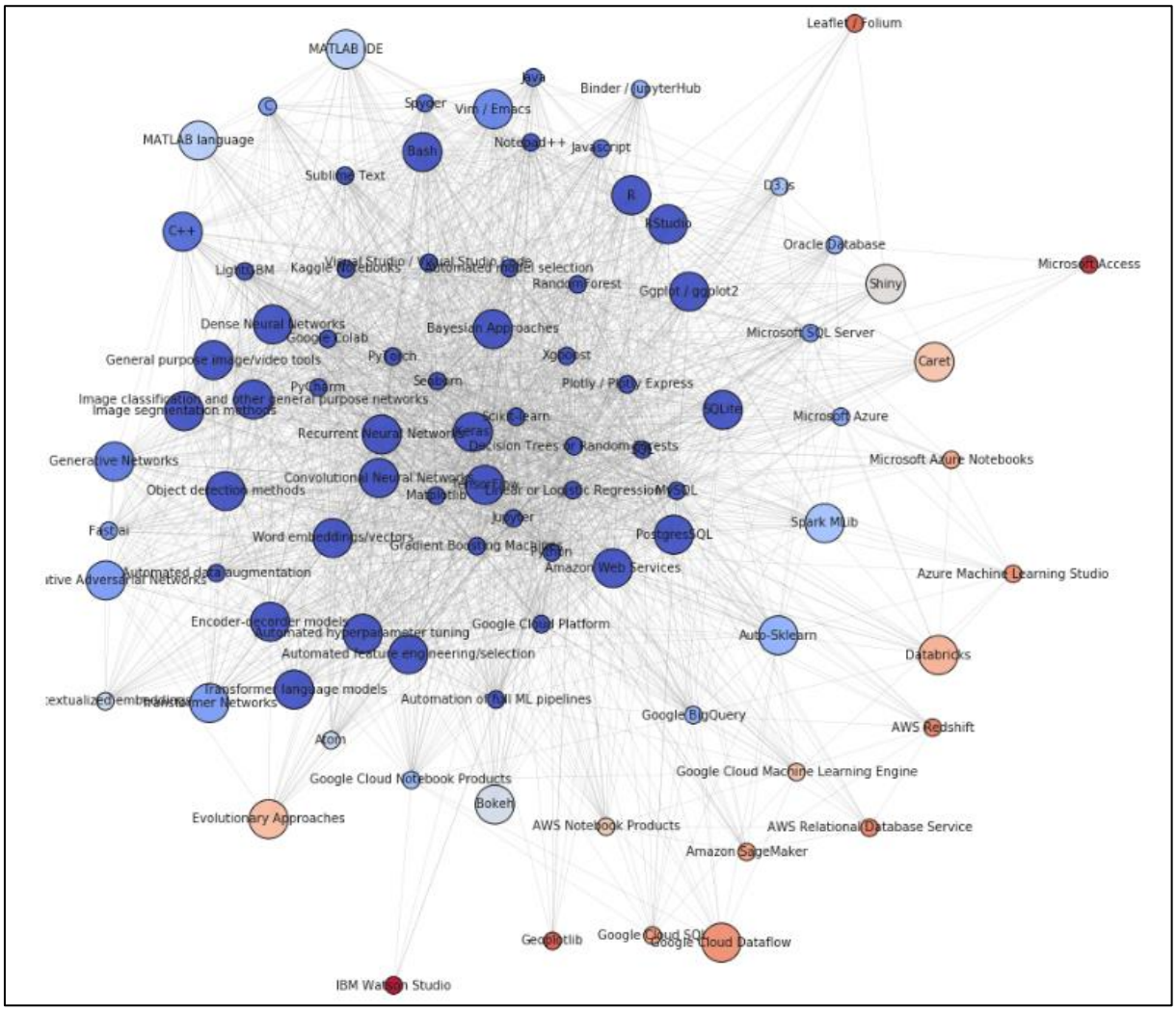


Рис. 2.16 – Інноваційна візуалізація статистичних даних щодо навичок датасайнтистів зі ступенем PhD серед користувачів Kaggle з використанням мережевого (графого) аналізу у ноутбуці «Exploring PhD Community with Network Analysis», яка посіла 2 місце на конкурсі «2019 Kaggle ML & DS Survey»

У 2019 р. був проведений конкурс «Data Science for Good: City of Los Angeles», де третина працівників служб адміністрації м. Лос-Анжелес виходила на пенсію і оголошувався конкурс на заміщення їх посад. Мета конкурсу полягала в тому, щоб перетворити папку, наповнену простими текстовими оголошеннями про роботу, в єдиний структурований файл CSV, а потім використовувати ці дані для: 1) визначення формулювань, які можуть негативно впливати на претендентів; 2) покращити різноманітність і якість заявників; 3) спростити визначення того, які підвищення доступні для працівників у кожній категорії посад.

Перше місце посів комплекс з 5 добре структурованих ноутбуків з цікавою візуалізацією «Bulletin Structuring Engine – CoLA» (це – посилання на перший із ноутбуків, але на початку є гіпертекстове меню з перемиканням на усі 5). Ці ноутбуки присвячені, передусім, автоматизації структурування даних, їх обробленню, аналізу та візуалізації їх особливостей. В наш час,

таке оброблення гарно виконують рішення на основі великих мовних моделей, у т.ч. з використанням API чат-ботів (ChatGPT та ін.), але гарна візуалізація такого оброблення поки містить значний елемент творчості – саме цим цікаві ці ноутбуки. На рис. 2.17 наведено приклад деякої візуалізації даних у різних ноутбуках цього комплексу.

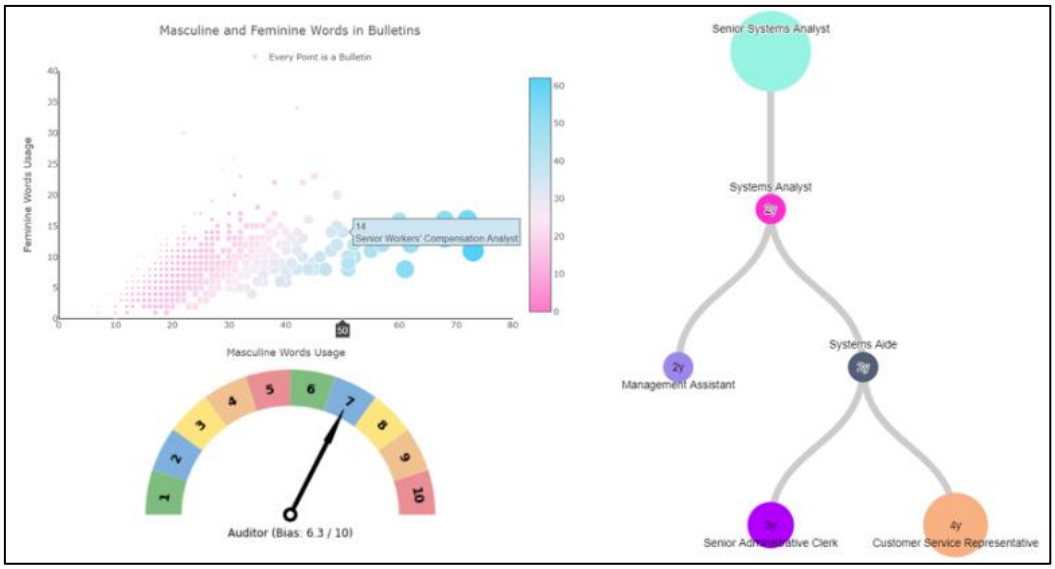


Рис. 2.17 – Інноваційна візуалізація змісту та структури оголошень про вакансії до міських служб м. Лос-Анжелес у комплексі з 5 ноутбуків «Bulletin Structuring Engine – CoLA», який посів 1 місце на конкурсі «Data Science for Good: City of Los Angeles»

Інший ноутбук «DSFG-CityOfLA-Analysis and Solution», який посів 2 місце, є прикладом гарного різноматнітного EDA комплексу текстової інформації. Окрім традиційних графіків, є аналіз з використанням бібліотеки SpaCy для порівняння відповідності кваліфікації людей вимогам до посад. Найбільш цікавим є інтерактивний графік зв'язків (рис. 2.18). У разі мишею на вершину графа у вигляді блакитного кола, автоматично розкривається піддерево, пов'язане з нею, у разі повторного натиснення – піддерево автоматично приховується. У такий спосіб можна здійснювати візуалізацію доволі складних дерев даних.

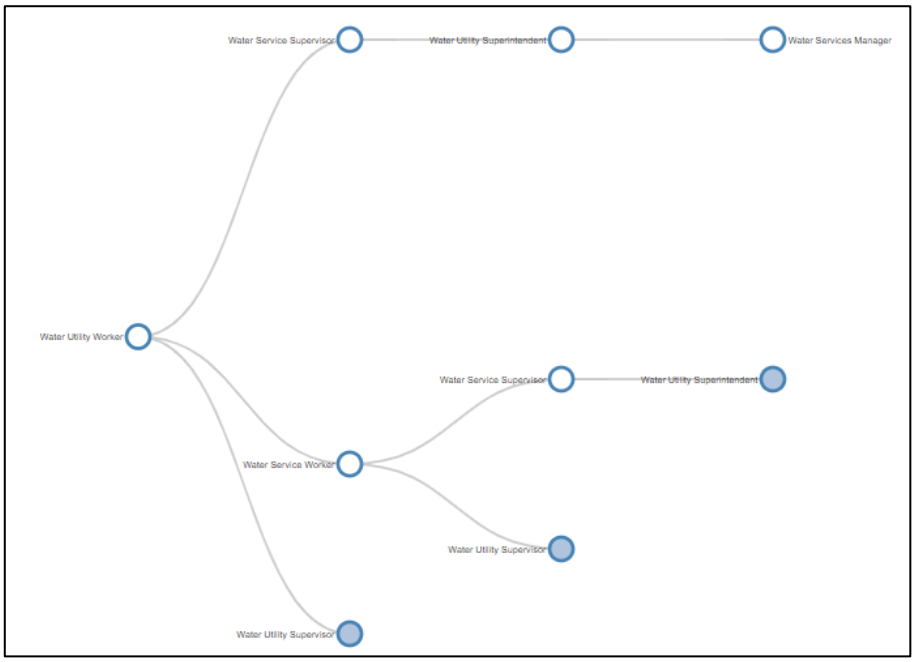


Рис. 2.18 – Інтерактивне дерево зв'язків вакансій до міських служб м. Лос-Анжелес у ноутбуці «DSFG-CityOfLA-Analysis and Solution», який посів 2 місце на конкурсі «Data Science for Good: City of Los Angeles»

Ноутбук EDA for tabular data: Advanced Techniques – це колекція найкращих ноутбуків Kaggle та дописи від переможців конкурсів з найкращим результатом, що містять високорівневі техніки розвідувального аналізу даних (EDA) для табличних даних.

У 2020 році відбувся щорічний конкурс 2020 Kaggle Machine Learning & Data Science Survey. Конкурс спрямований на створення інформативних та цікавих історій на основі набору даних опитування. Учасники мали розповісти історію про підгрупу спільноти даних, представлену в цьому опитуванні, через поєднання наративного тексту та дослідження даних. Оцінювання здійснюється з урахуванням таких критеріїв, як структура, оригінальність та документація.

Перше місце посів ноутбук "One chart, many answers: Kaggle Surveys in Slopes" – його автор використовує простий, але інноваційний тип графіку («Нахили»), щоб дослідити, як еволюціонували тенденції у сфері науки про дані у різні роки (з 2018 по 2020 рік). Ноутбук має чудову структуру та містить велику кількість документованих функцій та класів на Python, а інтерактивні графіки з нахилами дозволяють досліджувати цікаві тенденції та закономірності.

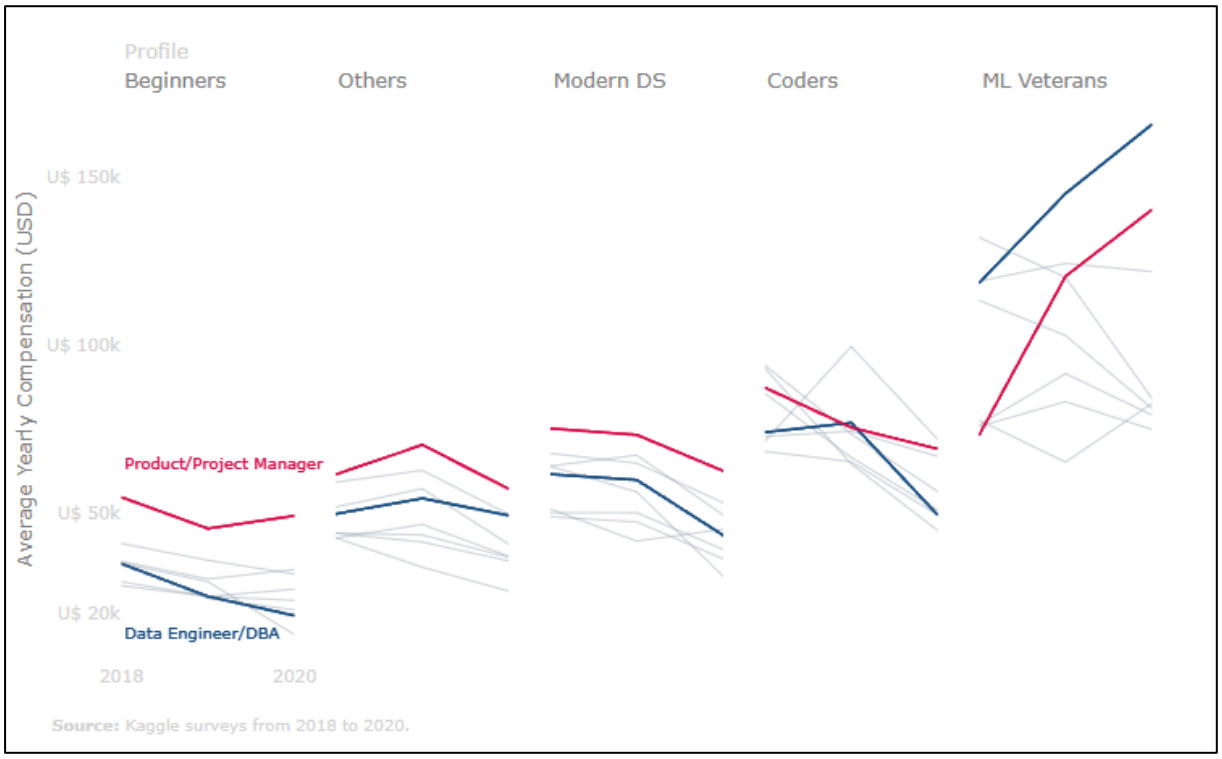


Рисунок 2.19 – Інноваційний тип графіку (нахили), щоб дослідити, як еволюціонували тенденції у сфері науки про дані протягом часу (з 2018 по 2020 рік)

Друге місце зайняв автор ноутбуку "Enthusiast to Data Professional - What changes?" – автор об'єднав Python, HTML та Javascript в єдиному ноутбуці, який містить детальні та інформативні таблиці та візуалізації. Цей ноутбук досліджує обов'язки різних професій у галузі обробки даних та містить багато додаткових інформативних обговорень, включаючи розмови про прогалини у навичках та інструментах торгівлі.

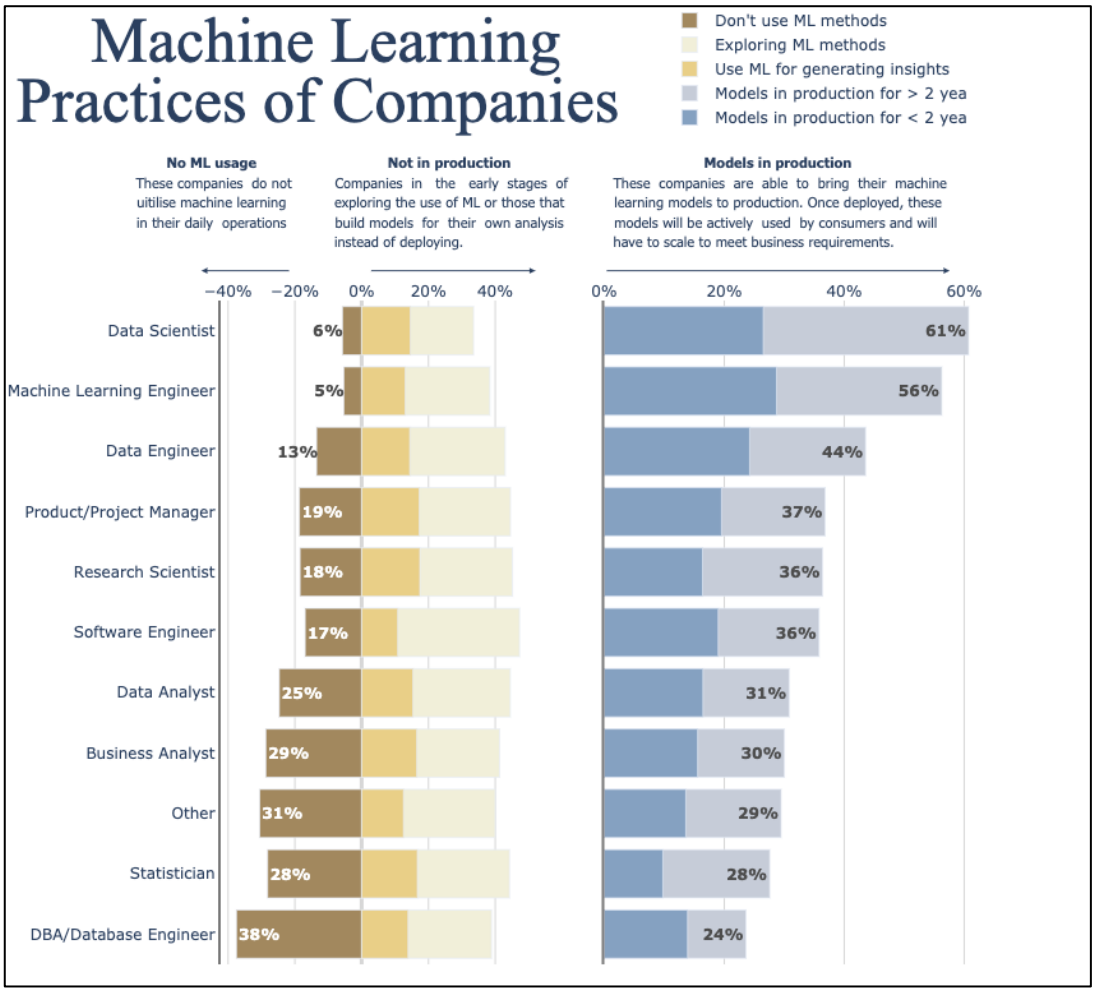


Рисунок 2.20 – Візуалізація поєднання технологій з статті про конкурс 2020 Kaggle Machine Learning & Data Science Survey

На рис. 2.21 наведена інфографіка інструментарію, згаданого у розділі 2 в цілому, у системі координат *S(I)*.

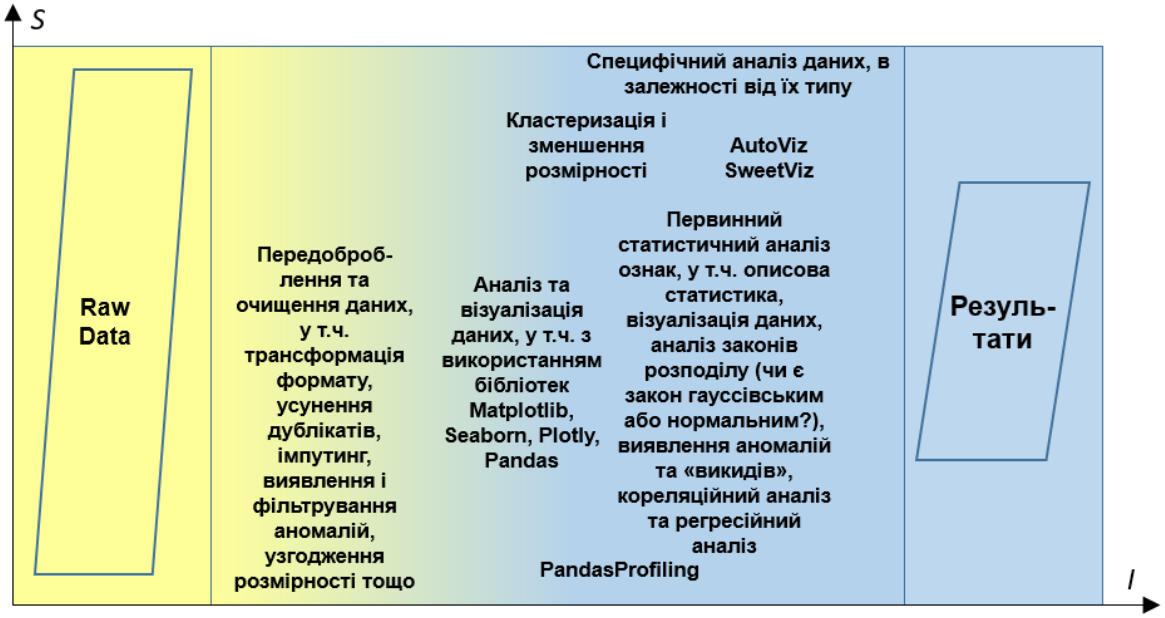


Рисунок 2.21 – Інфографіка розвідувального аналізу даних (EDA)

Можливі теми практичних і лабораторних завдань

Тема № 1. Вибір підходів та технологій для розвідувального аналізу та візуалізації даних на Python (або **«Розвідувальний аналіз даних про стан складної системи, аналіз та відбір ознак, візуалізація результатів аналізу на Python»**).

Метою заняття є вивчення інформаційних технологій і Python-бібліотек розвідувального аналізу й візуалізації даних та опанування практичних навичок їх застосування на прикладі одного із датасетів Kaggle чи за даними, завантаженими через API.

План заняття:

1. Знайти датасет з цікавими для аналізу реальними чи реалістичними даними з описом. Оптимально знайти датасет Kaggle, в якому є публічні ноутбуки з медалями (хоча б бронзовими). Описати його.
2. Вибрати Python-бібліотеки, які будуть використані для розвідувального аналізу та візуалізації даних (EDA): Matplotlib, Seaborn, Plotly, Pandas, Sklearn тощо, і зазначити для чого саме.
3. Зробити огляд ноутбуків чи статей щодо датасету з п.1 з використанням бібліотек з п.2 для його EDA. Навести не менше 5 графіків, з описом того, які саме закономірності вони ілюструють і що саме на них видно.
4. Розробити свій ноутбук, який проводить аналогічне чи інше дослідження (оптимальним є взяти наявний гарний (із золотою чи срібною медаллю) публічний Kaggle-ноутбук і адаптувати його до іншого датасету та самому зробити висновки про результати EDA.

Приклади ноутбуків з EDA:

- EDA for tabular data: Advanced Techniques
- Heart Disease - Multiple Clustering by 12 methods
- MNIST Original : 2D tSNE, 3D UMAP with RAPIDS
- MNIST Digits Original : 2D t-SNE with Rapids
- Automatic EDA with Pandas Profiling 2.9 (09.2020)
- Titanic Top 3% : cluster analysis
- Heart Disease - Automatic AdvEDA & FE & 20 models
- Autoselection from 20 classifier models & L_curves
- Biomechanical features - 20 popular models
- Suspended substances prediction in river
- AI-ML-DS Training. L1T: NH4 - linear regression

Також, можна використати усі ноутбуки з датасетів COVID-19 in Ukraine: daily data чи Forecasting Top Cryptocurrencies.

Контрольні питання

1) Що включає в себе процес передоброблення даних, і чому це важливо перед подальшим аналізом?
2) Назвіть кілька методів очищення даних та наведіть приклади ситуацій, коли вони можуть бути застосовані.
3) Які методи кластеризації використовуються для групування схожих об'єктів? Наведіть приклади їх використання.
4) Яким чином методи зменшення розмірності даних можуть допомогти у подальшому аналізі та моделюванні?
5) Що таке розвідувальний аналіз даних (EDA) і які завдання він вирішує в процесі аналізу даних?

6) Які візуалізаційні інструменти можна використовувати для проведення EDA? Наведіть приклади типів графіків.

7) Які основні кроки включає в себе розвідувальний аналіз даних, проведений призерами змагань Kaggle?

8) Які інтелектуальні методи використовуються для розвідувального аналізу даних у конкурсах Kaggle?

9) Як визначити аномалії або викиди в даних під час EDA, і як це впливає на подальший аналіз?

10) Чому важливо розуміти розподіл цільової ознаки під час розвідувального аналізу даних?

3 ІНЖЕНЕРІЯ ОЗНАК (FEATURE ENGINEERING)

3.1 Основні задачі та етапи інженерії ознак (FE). Синтез нових ознак

Інженерія ознак (FE) – це аналіз та оброблення ознак (фічерів – від англ. «feature») датасету, у т.ч. видалення малоінформативних та синтез нових.

Метою FE є наступне:

- виявлення малоінформативних ознак (наприклад, з усіма однаковими значеннями чи з, переважно, пропущеними або аномальними значеннями), які можна видалити, зменшивши зашумлення рішення та підвищивши швидкодію роботи моделі;

- виявлення і видалення ознак, які мають детермінований зв'язок з іншими (кореляція дорівнює 1 – досвід показує, що таке буває на практиці: наприклад, фахівці з моніторингу якості вод одні показники вимірюють, а інші по них визначають по формулі і усі вони потрапляють в один датасет);

- виявлення висококорельованих пар ознак (коефіцієнт кореляції є вищим за 0.95, а то й – 0.99), але щодо видалення однієї з них слід поставитись обережно – їх цінність обов'язково слід окремо дослідити;

- виявлення і видалення «витоків», коли якась ознака містить в собі значення таргету чи їх можна отримати з неї на основі детермінованих залежностей, тобто модель просто може по ній напряму точно обчислити таргет;

- аналіз важливості кожної ознаки з використанням як відносно простих моделей (регресій, дерев рішень), так і – складних, у разі проходження цього етапу повторно вже після етапу побудови складних моделей;

- синтез нових більш інформативних ознак за значеннями наявних, про що буде йтись нижче;

- удосконалення значень ознак (можуть використовуватись операції передоброблення, але відмінність від попереднього розділу в тому, що ці операції проводяться вже після етапу EDA).

З досвіду авторів, одна із проблем машинного навчання – це значна кількість пропущених даних у таблиці, наприклад, у медичних даних, зібраних з різних лікарень за різні роки. Якщо видалити усі рядки, де є пропущені, чи – усі стовпці, тоді може видалитись усе. Тоді йдуть на компроміс: спочатку видаляють рядки з найбільшою кількістю пропущених, потім – стовпці (ознаки) і так повторюють декілька разів. В результаті, таблиця зменшується, але, у ній може залишились достатньо даних для аналізу і розв'язання задачі.

Методи FE, як і EDA, застосовуються як до усіх даних, так і до їх кластерів та окремо до тренувальних, валідаційних і тестових даних.

Залежно від особливостей задачі та результатів дослідження, FE може містити наступні етапи (якщо не зазначене інше, то мається на увазі таблиця даних):

1. Синтез принципово нових ознак на основі знань про предметну область. Для цього вивчають зміст ознак, постановку задачі, вивчають аналоги розв'язання подібних задач у GitHub, Kaggle, у фахових статтях у Google Scholar та в іншій спеціальній літературі. Іноді, достатньо просто проявити ерудицію або почитати матеріали на сайті організації, яка надала дані, про її діяльність. Наприклад, коли автори брали участь у Kaggle-змаганні «New York City Taxi Fare Prediction» у 2018 р., де треба було передбачити вартість поїздки на жовтих таксі Нью-Йорку, не знаючи маршрут. Датасет містив 40 млн. рядків за 5 років. Допускалось використання додаткових ознак. Це був перший конкурс 2-х із 3-х авторів у Kaggle. Ми згенерували цілий ряд нових дуже інформативних ознак, які суттєво дозволили підвищити точність:

1) проглянули сайт таксі і узнали, що там – гнучка тарифікація у години «пік», вихідні дні і свята, для поїздок в аеропорти та ін. і розробили систему правил, які дозволили додати відповідні ознаки у датасет за цими закономірностями, для цього довелось використати просторові шари у форматі shp з муніципального сайту Нью-Йорка (для візуалізації використовували безкоштовний пакет програм QGIS, але й на Python є відповідні засоби оброблення та візуалізації просторових даних), а ще можна використати й просто доступні сервіси на кшталт OpenStreetMaps чи ін.;

2) таксі часто беруть від чи до станцій метро і вокзалів (залізничних, автомобільних, морських, річкових) – додали такі ознаки (теж використали карту, а ще – супутникові дані, на яких добре було видно де саме там паркуються таксі, тобто – на якій відстані від вокзалів та станцій);

3) врахували метеоумови (температуру, дощ, сильний вітер та ін.), які впливають на рішення людей взяти таксі, а не йти пішки – завантажили їх з сайтів аеропортів;

4) врахували через скільки мостів чи тунелів пролягав маршрут (провели умовні лінії між частинами міста, розділеними водою, апроксимували їх рівняннями прямих, розробили систему правил для усіх можливих варіантів кількості мостів та/або тунелів (від 0 до 2) і з використанням функції apply розмітили весь датасет – засобами бібліотеки pandas це спрацювало дуже швидко. А з використанням циклу for з розгалуженням це було б набагато довше. Важливо не тільки придумати ознаку, а – й швидкі засоби її обчислення, враховуючи великий обсяг даних. Для деяких датасетів це може зайняти надто багато часу.

2. Аналіз типів даних усіх ознак. Як правило, з використанням функції для датафреймів за допомогою df.info() бібліотеки pandas. Крім того, у бібліотеці numpy є спеціальні константи, які дозволяють виявляти які з ознак точно є числовими, а які – ні. Дивись приклади використання цього у порадах «Тір 5.1» та «Тір 5.2» в [10]).

3. Виявлення найбільш інформативних ознак, які задовольняють таким мінімальним вимогам:

- не є витоком,
- не є константою;

- не має пропущених значень (чи їх одразу не було, чи вже проведене відповідне передоброблення: видалення рядків чи імпутинг);

- не є ознакою, висококорельованою з іншою(ими).

Додатковою вимогою є висока важливість ознаки, але про це – у підрозд. 3.3.

4. Дискретизація інформативних ознак шляхом утворення малої множини (3-10) значень числової ознаки замість великої кількості або дробових значень у певному діапазоні, наприклад, шляхом ділення націло на певне число, яке є 1/5 від максимального – див. пораду «Tip 5.4» в [11].

5. Формальний синтез нових («вторинних» або «синтетичних») ознак з наявних за таким алгоритмом (див. приклад там же у «Tip 5.4» в [11]):

1) перетворити значення наявних інформативних ознак на тип «str»;

2) об'єднати значення 2, 3 чи більше ознак через якийсь символ (знак підкреслення «_», дефіс чи ін.), а тоді, наприклад, для *i*-ї та *j*-ої ознак датафрейму df:

```
df[i + "_" + j] = df[i].astype('str') + "_" + df[j].astype('str');          (3.1)
```

3) закодувати новоутворені значення числами (номерами значень у списку).

Для виявлення дійсно цікавих закономірностей варто об'єднувати у такий спосіб тільки ознаки з малою кількістю можливих значень – або текстові ознаки, або дискретизовані (див. п.1) аналоги числових ознак, тоді є шанс виявлення нових цікавих класів. Наприклад, в конкурсі «Microsoft Malware Prediction» у 2018 році треба було знайти закономірності появи вірусу на біля 9 млн. комп'ютерів з MS Windows. Об'єднання 2 чи 3 ознак разом дозволяло виявити ряд класів, які мали найбільшу ймовірність появи вірусів, залежно від типу ліцензії, мови браузера, країни, версії операційної системи та ін.

Важливо пам'ятати, що у разі такого об'єднання і виявлення дійсно цікавих закономірностей, не варто використовувати в моделі показники, з яких утворились нові, принаймні не усі, інакше вони усі разом не будуть нести корисної інформації. Якщо ж в (3.1) використати не додавання, а – якусь нелінійну функцію (множення, ділення, корінь, степінь та ін.), тоді нова ознака буде принципово новою, яку більшість моделей не зможуть синтезувати самостійно, і тоді є сенс враховувати й усі ознаки, з яких вони синтезована. Прикладом такого синтезу є синтез технічних показників криптовалюти у підрозділі 2.4 ноутбуку Crypto - BTC : Advanced Analysis & Forecasting.

Цікавим прикладом є синтез ознаки в конкурсі «Google Analytics Customer Revenue Prediction» (Інтернет-магазин Google), де треба було аналізувати коли користувач, який проглядає сторінки Інтернет-магазину, нарешті щось купить і передбачати обсяг транзакцій, тобто на яку суму буде здійснено купівлю. Один із дослідників запропонував обчислити ознаку «pageviews/hits», яка є відношенням кількості переглянутих веб-сторінок, поділеної на загальну кількість так званих «хітів» – запитів до сервера усіх

типів (перегляд сторінки, натискання на кнопки, події або інші взаємодії, що призводять до передачі даних аналітичній системі). У ноутбуці побудована така цікава закономірність між цією ознакою і сумарним доходом від транзакцій «transactionRevenue» за даними 2018-2019 рр. (рис. 3.1).

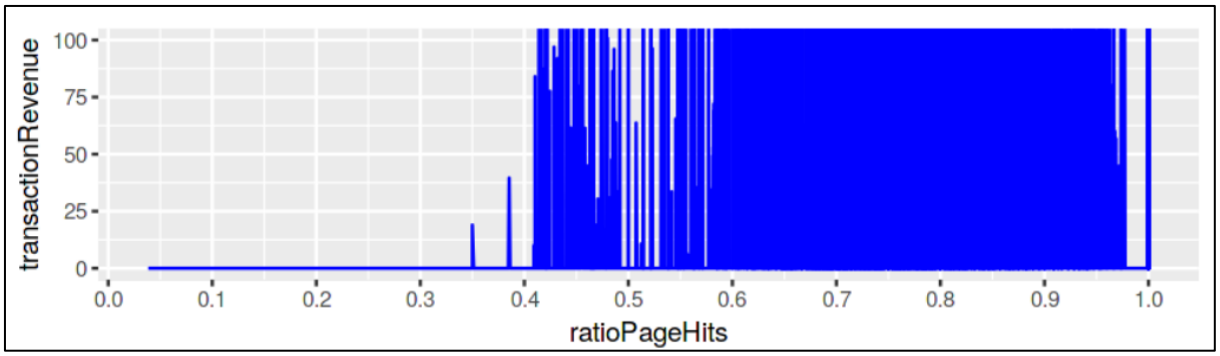


Рисунок 3.1. Залежність між сумарним доходом від транзакцій в Інтернет-магазині Google «transactionRevenue» та «pageviews/hits» – кількістю переглядів веб-сторінок, поділеною на кількість хітів

Як видно з рис. 3.1, по ній одній вже можна отримати розв'язок задачі чималої точності, оскільки добре видно, після якого порогового значення клієнт практично точно щось купить, а до якого – ні. Звичайно, для отримання рішення високої точності треба враховувати більше факторів.

Взагалі, досвід показує, що в більшості задач не варто взагалі видаляти ніякі інформативні ознаки – рідко, коли видалення таких ознак дійсно підвищує точність, але точно підвищує швидкодію, економить оперативну пам'ять і дозволяє застосувати більш ефективні моделі, які, якраз можуть підвищити точність.

До етапу FE ще часто відносять деякі операції передоброблення (див. п.1 і 2 у підрозд. 2.1) з економії пам'яті за рахунок уточнення формату ознак та трансформацію форматів. Однак, ми ці операції відносимо до передоброблення, оскільки їх варто застосовувати одразу після чи під час завантаження датасету і до виконання EDA. А етап FE, зазвичай, має місце після етапу EDA.

6. Зменшення розмірності (кількості) ознак. Іноді датасет може мати велику кількість (сотні, тисячі) однотипних ознак і це суттєво ускладнює його оброблення. Тоді варто спробувати застосувати кластеризацію чи зменшення розмірності. Однак, варто пам'ятати такі особливості:

1) перед застосуванням цієї операції варто здійснити стандартизацію даних (див. наступний підрозділ), щоб ознаки аналізувались однотипно;

2) новоутворені ознаки не будуть мати фізичного сенсу – це будуть просто математично агреговані деякі безрозмірні величини;

3) усі операції кластеризації у Python-бібліотеках здійснюють кластеризацію в рядках таблиці, а в даному випадку її слід робити у стовпцях, отже, перед кластеризацією таблицю слід транспонувати, а потім, після кластеризації зробити це знов (при цьому слід дуже уважно зберігати номери

рядків і стовпців, оскільки кластеризація оперує тільки числами в комірках матриці) (приклад кластеризації 300 стовпців медичного датасету див. в авторському ноутбуці).

7. Факторний аналіз. Як відомо, однією з основних цілей факторного аналізу є виявлення прихованих (латентних) факторів, які можуть пояснити спостережувані зв'язки між ознаками в датасеті. У разі виявлення таких факторів на їх основі можна синтезувати нову ознаку чи ознаки. А у разі, заміни великої кількості похідних ознак від цих факторів на ці фактори, що їх спричиняють, можна зменшити розмірність датасету, як з використанням попереднього прийому. Деякі датасайнтисти відносять FE-прийоми факторного аналізу, кластеризації та зменшення розмірності до одного типу, оскільки результатом їх усіх є зменшення розмірності. Однак, факторний аналіз дає іншу інтерпретацію виявленим закономірностям, на відміну від операцій кластеризації, що може бути цінним на етапі EDA, який може слідувати після етапу FE.

8. Автоматичний синтез нових ознак (до 1200 штук) з використанням статистичних ознак з різною періодичністю та агрегацією для заданого часового ряду з використанням бібліотеки TSFresh, що більш детально буде розглянуто у підрозд. 8.1. Також, там буде розглянуто ще ряд специфічних прийомів для часових рядів: взяття різниці diff() чи різниці різниць diff().diff() сусідніх значень, усереднення з ковзним вікном rolling() тощо.

9. До інженерії ознак можуть відноситись операції зі значеннями ознак (також вони можуть застосовуватись і на етапі передоброблення), якщо їх необхідність була виявлена на етапі EDA.

Наприклад, якщо на етапі EDA по гістограмі було виявлено, що якась ознака має невелику кількість основних значень і велику кількість (іноді по 1 шт) інших малоцінних значень, тоді є сенс усі такі малоцінні значення замінити на одне «Other» (див. пораду «Tip 4.4» в [10]).

10. Часто датасет містить ознаку з датою (іноді – дата і час з точністю до мілісекунд). Вона має цінність для моделей часових рядів, а для моделей машинного навчання для таблиць з ознаками вона має цінність лише, як джерело інформації для синтезу нових ознак: рік, квартал, місяць, декада, тиждень, доба, 12-, 6-, 4-годинний інтервал, години «пік» тощо. Як було зазначено вище, якщо дату залишити, тоді модель «заоверфітиться» під неї і буде враховувати тільки її та прогнозувати тільки ті значення у ті самі дати навчального датасету і не зможе бути достатньо узагальненою як для інших дат.

11. Балансування різних класів таргета у тренувальному датасеті. Бувають випадки, коли один варіант таргета є дуже рідким, у порівнянні з іншим(и). Це ускладнює машинне навчання. Тоді застосовують балансування даних. Загалом-то, це – псування даних, тому цю операцію слід використовувати, якщо немає чи не є ефективними інші способи, наприклад, використати спеціальні параметри моделей для врахування факту дисбалансу зна-

чень. Для балансування даних можна використати, наприклад метод Synthetic Minority Oversampling Technique (SMOTE) – див. приклад у статті по аналізу хворих на коронавірус у Великобританії [20].

У ноутбуку [«Titanic Top 3% : cluster analysis»], згаданому у підрозд. 2.2, застосовано ще один підхід: відбираються пари різних фічерів, одним з яких, щоразу, є фічер "WomanOrBoySurvived"). Для кожної пари ознак автоматично визначається оптимальний метод кластеризації за критерієм максимальної косинусної подібності з цільовою ознакою в навчальному наборі даних (повна подібність дорівнює 1). Якщо цей критерій для оптимального методу перевищує задане порогове значення limit_opt, тоді на основі цієї пари ознак синтезується новий фічер і використовується в подальшому під час машинного навчання. Візуалізація оптимальних методів для різних пар фічерів наведена на рис. 2.7 вище як приклад роботи методів кластеризації.

Наведені операції FE потребують певної творчості і проводяться, як правило, по черзі з етапом EDA, в циклі. Але є операції FE, які проводяться вже з фінальним датасетом, який на етапі EDA визначений як такий, що відповідає мінімальним вимогам і є перспективним для побудови моделі. Це операції стандартизації і нормалізації даних.

Крім того, є важливий етап FE як аналіз важливості ознак. Для цього потрібні моделі машинного навчання – або спрощені (на першому етапі), або – вже після етапу, наступного за FE – етапу побудови моделі. Розглянемо ці операції більш детально.

3.2 Стандартизація та нормалізація ознак

Більшість моделей машинного навчання намагаються підлаштуватись під усі ознаки однаково, але ті ознаки, які мають більшу дисперсію, більшу різноманітність значень та відхилення від них, будуть більш впливовими, відтак модель може під них «заоверфітитись» (саме тому вище давалась порада видаляти ознаку з моментами дати і часу). Щоб цього уникнути, дані варто стандартизувати (рис. 3.2).

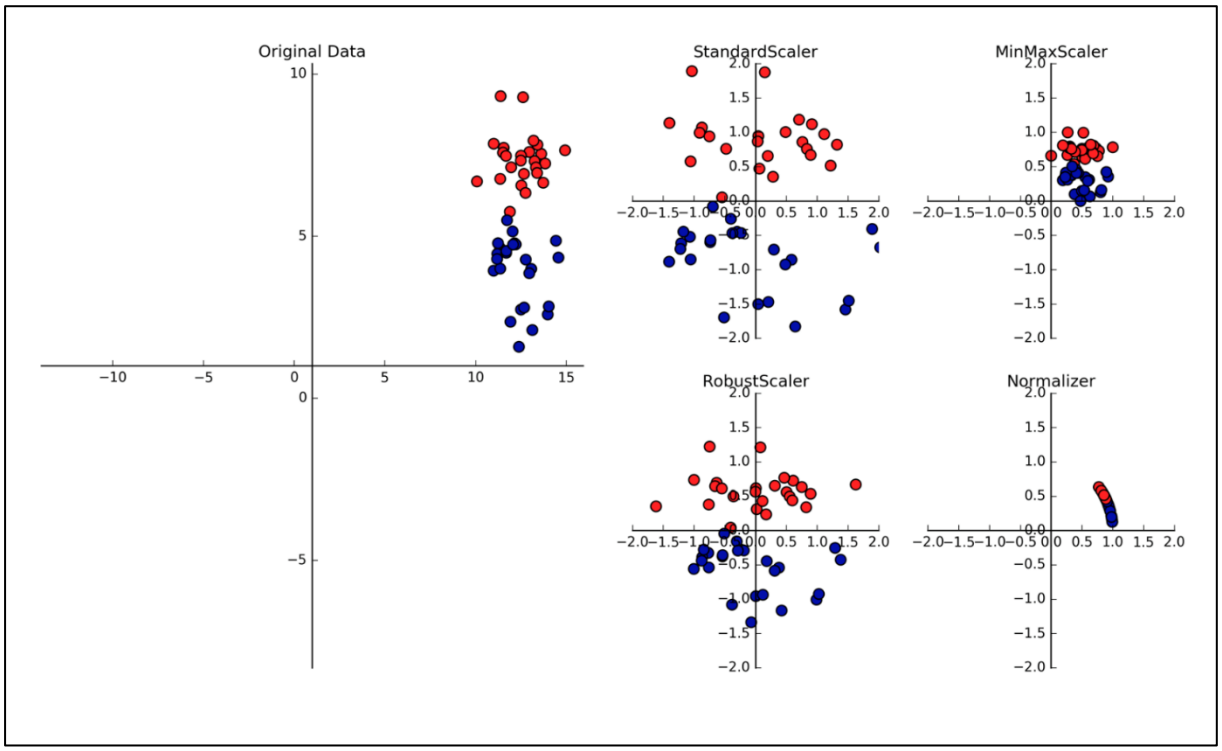


Рисунок 3.2 – Типові методи стандартизації та нормалізації ознак Python-бібліотеки sklearn з документації: а) приклад даних, де проєкція даних на вертикальну вісь явно є більшою за проєкцією на – горизонтальну; б) результат класичної стандартизації; в) результат мінімаксного масштабування; г) результат робастної стандартизації; д) результат нормалізації даних

У бібліотеці Sklearn є ряд методів для цього:

1. Стандартизація (sklearn.preprocessing.StandardScaler) (див. рис. 3.2б), яка виконується до усіх значень ознаки x, за винятком цільової (таргета) з отриманням центрованої величини $\tilde{x}$ за формулою (хоча, в загальному випадку функції StandardScaler, вирази у знаменнику та чисельнику є опціональними, тобто можна задати, щоб вони не виконувались: не віднімалось середнє m_x або не було ділення на середньоквадратичне відхилення σ_x):

$$\tilde{x} = \frac{x - m_x}{\sigma_x}. \quad (3.2)$$

Завдяки цій трансформації усі ознаки перетворюються на рівнозначні з однаковою дисперсією та розкидом значень симетрично навколо нуля (випадкову величину у чисельнику ще називають *центрованою* [21]). Головна незручність її в тому, що, по-перше, бібліотека sklearn працює тільки з матрицями – числами датафрейму. Отже, для її застосування треба спочатку зберегти назви фічерів, стандартизувати, а потім почепити ті назви – назад. А по-друге, в тому, що на проміжних етапах модель стає важче інтерпретувати, оскільки фічери втрачають свій первинний зміст та розмірність. Втретє, слід не забувати після виконання оброблення виконати зворотну операцію, якщо важливим є отримання та інтерпретація якихось проміжних

результатів. Саме тому, ця операція не є обов'язковою і не є вбудованою у методи навчання моделей. Її варто використовувати, якщо вона дає позитивний ефект щодо підвищення точності моделей.

Головне, що слід пам'ятати, це те, що стандартизувати таргет не можна, можна – тільки вхідні ознаки. Якщо метою є тільки передбачення таргета, тоді усі вище зазначені недоліки зникають. Дані стандартизують, будують модель, проводять передбачення і аналізують вже тільки таргет.

2. Мінімаксне масштабування (sklearn.preprocessing.MinMaxScaler) (див. рис. 3.2в) ознаки x здійснюють по її мінімальному x_{min} і максимальному x_{max} значеннях за формулою (3.3).

$$x_m = \frac{x - x_{min}}{x_{max} - x_{min}}. \tag{3.3}$$

Його найбільш популярне застосування – для відображення різних ознак на одному графіку в діапазоні від 0 до 1 по осі ординат. Для побудови моделей це перетворення не є ефективним, оскільки моделі машинного навчання, як правило, самостійно враховують таку лінійну трансформацію.

3. Стандартизація за формулою (3.2) може призводити до значних спотворень даних, у разі, якщо є значні аномалії. За таких випадків, застосовують робастну стандартизацію sklearn.preprocessing.RobustScaler (див. рис. 3.2г).

4. Деякі методи кластеризації є більш ефективними для різнорідних даних, якщо їх спочатку нормалізувати (див. рис. 3.2д). Можливі 3 варіанти нормалізації, в залежно від вибраної норми: на основі суми модулів вектора, на основі євклідової норми (довжини вектора), що є варіантом за замовчуванням), на основі максимального значення вектора.

Зазвичай, усі ці функції стандартизації та нормалізації налаштовують (scaler.fit_transform) на навчальних даних, а застосовують (scaler.transform) і до навчальних, і до тестових, інакше буде – витік даних, тобто модель стане непрацездатною за реальних умов, коли тестові дані наперед невідомі.

На рис. 3.3 наведена інфографіка інструментарію, згаданого у підрозділах 3.1 та 3.2, у системі координат *S(I)*.

3.3 Побудова діаграм важливості ознак та автоматизація вибору ознак на основі бібліотек Sklearn, SHAP, LIME. Інтерпретабельність моделей.

Як було зазначено вище, важливо не тільки побудувати модель машинного навчання, а й – вміти зробити по ній правильні висновки. Часто адекватна модель потрібна не для передбачення, а – для забезпечення обґрунтованих висновків за нею. Тому важливою є інтерпретабельність моделей. А її часто здійснюють на основі відносного порівняння важливості ознак за певних умов.

За першого застосування FE для аналізу важливості ознак застосовуються прості моделі, наприклад, лінійна регресія чи дерева рішень з бібліотеки Sklearn і визначається важливість ознак (англ. «feature importance» - FI). Будується відповідна FI-діаграма за зменшенням цієї важливості та аналізується (рис. 3.3).

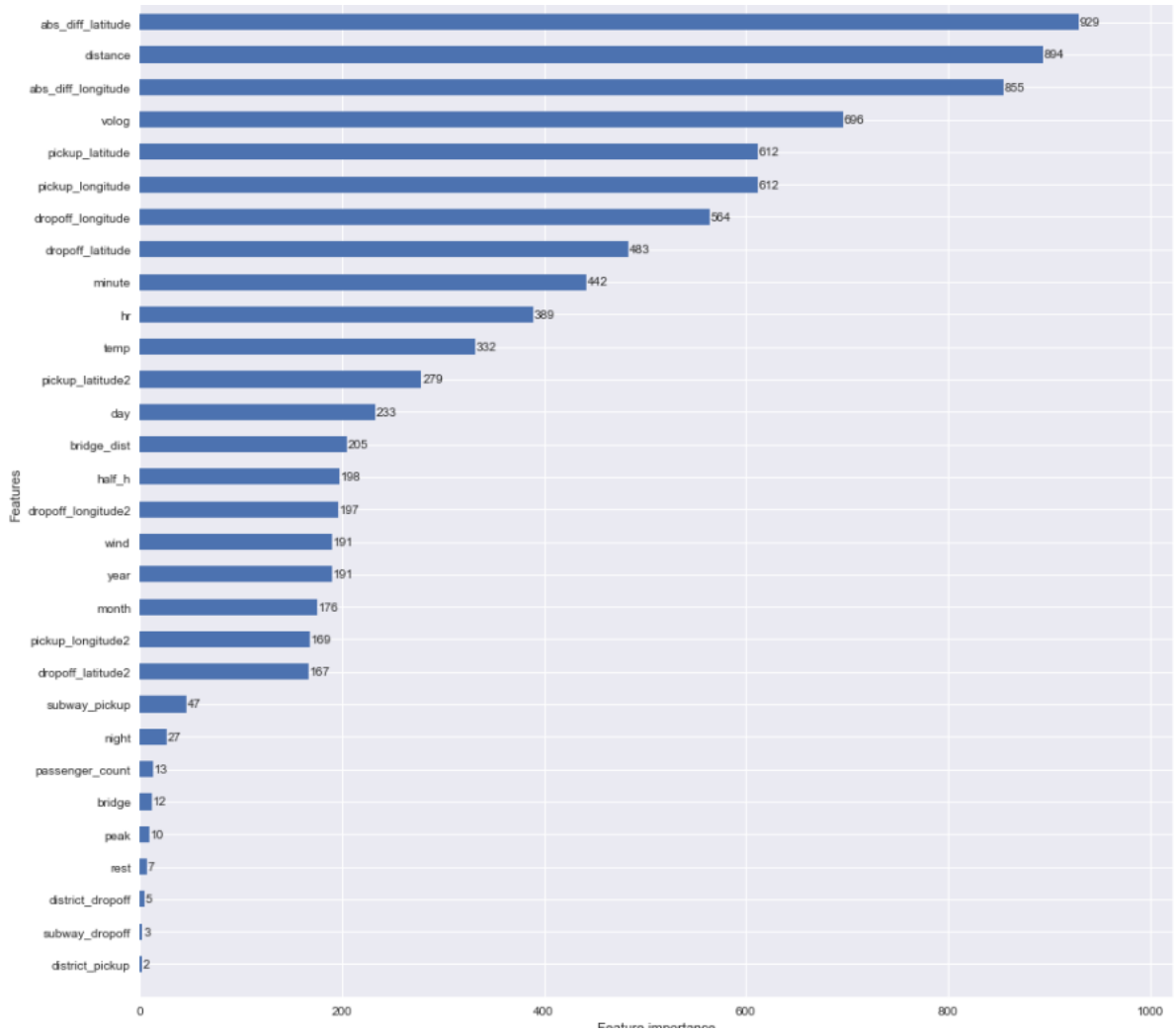


Рисунок 3.3 – Приклад діаграми важливості ознак для дерева рішень в задачі передбачення вартості поїздки в таксі м. Нью-Йорк у змаганні Kaggle у 2018 р. з авторського рішення

По діаграмі важливості ознак варто робити такі висновки:

1. Якщо ознака зустрічається дуже часто (може й в усіх рядках датафрейму), але її важливість дуже низька, тоді її варто видалити як неінформативну, яка тільки вносить додатковий шум (див. на рис. 3.3 на ознаку «passenger_count» – кількість пасажирів).
2. Якщо ознака є дуже важливою, тоді її варто залишити.
3. Якщо ознака має низьку важливість, але зустрічається вона рідко, тоді її цінність слід аналізувати тільки по точності моделі (якщо з нею вона підвищується, тоді слід залишити, якщо – ні, тоді можна видалити).

Видаляти ознаки на основі тільки їх важливості в FI-діаграмах – помилковий підхід. Наприклад, ознаки на рис. 3.3 з найменшою важливістю «bridge», «peak», «rest», «subway_dropoff», які враховували через скільки мостів або тунелів проходив маршрут таксі, чи їхало воно у години пік чи вихідні, чи пасажир сів або вийшов біля станцій метро, відповідно, а також ознаки з назвами 3-х аеропортів Нью-Йорка, які, через їх мале значення, на рис. 3.3 не показані, дозволили на 20% підвищити точність передбачення вартості поїздки. Причиною в тому, що діаграма нарис. 3.3 показує важливість ознак не для передбачення даних, а – для побудови дерева рішень. Чим у більшість кількості вузлів (умов розгалуження) використовується ознака, тим вона є важливішою. А тому, ознаки, які стосувались тільки поїздок за певних умов (у певний час чи у певних координатах) і складали невеликий відсоток від загальної кількості поїздок, фігурують у порівняно невеликій кількості правил. Однак, це були поїздки з найбільшою вартістю і вони найбільше впливали на похибку. А отже, їх видаляти не можна.

Вже на цьому прикладі видно, що інструмент FI-діаграм на основі простої моделі дерев рішень чи ін. є не дуже надійним та однозначним. В наступному підрозділі будуть розглянути більш об’єктивні способи побудови діаграм важливості ознак, які враховують саме вплив на точність передбачення моделі.

На етапі FE важливими є не тільки методи аналізу важливості ознак, а й – автоматизовані методи відбору найкращих ознак. Для цього існують спеціальні методи автоматизації вибору ознак (англ. «feature selection» – FS) з бібліотеки Sklearn [22]. Більшість з них використовують вибрану аналітиком модель машинного навчання, більш детально про які буде викладено у наступному розділі.

1. Відбір ознак за коефіцієнтом кореляції Пірсона (corr у бібліотеці pandas – для лінійних залежностей чи ознак, розподілених за нормальним законом) чи Спірмена (corr(method='spearman') у бібліотеці pandas – для нелінійних залежностей чи ознак, розподілених не за нормальним законом).

2. Відбір ознак за допомогою методу SelectFromModel за допомогою певної моделі машинного навчання, як правило, лінійної. Наприклад – з моделлю LinearSVC – лінійний метод опорних векторів, який використовує гіперплощину для розділення класів у просторі ознак (більш докладно про цей метод див. у п. 4.1.7). Важливість кожної ознаки оцінюється або за модулем вагового коефіцієнта (для лінійного ядра), або за розташуванням відомої гіперплощини (для нелінійного ядра). Але можливим є використання й інших лінійних моделей (див. нижче підрозд. 4.1).

3. Метод SelectKBest. Його основний принцип полягає в тому, щоб за допомогою визначеної метрики (статистичного критерію) вибрати K найкращих ознак з набору даних, які мають найбільший вплив. SelectKBest може використовуватися з різними метриками та статистиками, залежно від типу задачі (класифікація чи регресія) і природи даних. Метрики і статистики для класифікації: F-статистика, для категоріальних ознак - Chi-2 (χ^2-

критерій); для регресійних задач: ANOVA (англ. «ANalysis Of VAriance»), коефіцієнт кореляції Пірсона чи Спірмена тощо.

4. Відбір ознак за допомогою методу Recursive Feature Elimination (RFE). Алгоритм поступово видаляє менш важливі ознаки до досягнення заданої кількості. Ключовим є параметр «step». Якщо він є натуральним числом, тоді – задає кількість ознак, які слід видалити на кожній ітерації. Якщо він від 0 до 1, тоді – вказує який відсоток (округлений до меншого цілого) ознак слід видалити з усієї множини. Параметр n_features_to_select вказує скільки ознак слід залишити. За замовчуванням, це – 50%. Або може задаватись точна кількість чи відсоток, який слід залишити. У цьому методі слід задавати модель машинного навчання, за якою визначається важливість ознак на кожній ітерації. Метод не має обмежень на це – дослідник може вибирати, теоретично, будь-яку модель.

5. Відбір ознак за допомогою VarianceThreshold — видаляються ознаки з низькою варіативністю (малою кількістю унікальних значень). Звідси й назва: поріг (англ. «Threshold») для дисперсії (англ. «Variance»). Важливо, що цей метод – інваріантний до значень цільової ознаки. Він застосовується тільки до вхідних ознак і може бути ефективним і для навчання без вчителя.

У порадах «Тір 5.8-5.15» в [11]) наведено приклади застосування цих методів FE, а також – варіант, який дозволяє ранжувати скільки разів кожна ознака була найкращою (наприклад, у першій 20-ці) за кожним методом, потім додати цю кількість і ранжувати. Це є багатокритеріальним методом відбору ознак.

Можливо реалізувати власний алгоритм вибору найкращих ознак. Він може включати в себе різні критерії, такі як важливість, інформативність чи відповідність завданню. Фінальний вибір може базуватися на аналізі важливості ознак, знайдених під час попередніх етапів відбору ознак. Див. наприклад, як це зроблено в авторському ноутбуці [22] з використанням біля 10 методів одночасно з подальшим узагальненням за кількістю разів потрапляння в найкращі.

Більш ефективними інструментами для дослідження та візуалізації важливості ознак є методи бібліотек, основані на теорії ігор та на апроксимації спрощеними моделями в околі конкретного прикладу даних. Розглянемо їх детальніше.

Найбільш цікаві і детальні пояснення ознак із гарною візуалізацією забезпечують методи бібліотеки SHAP.

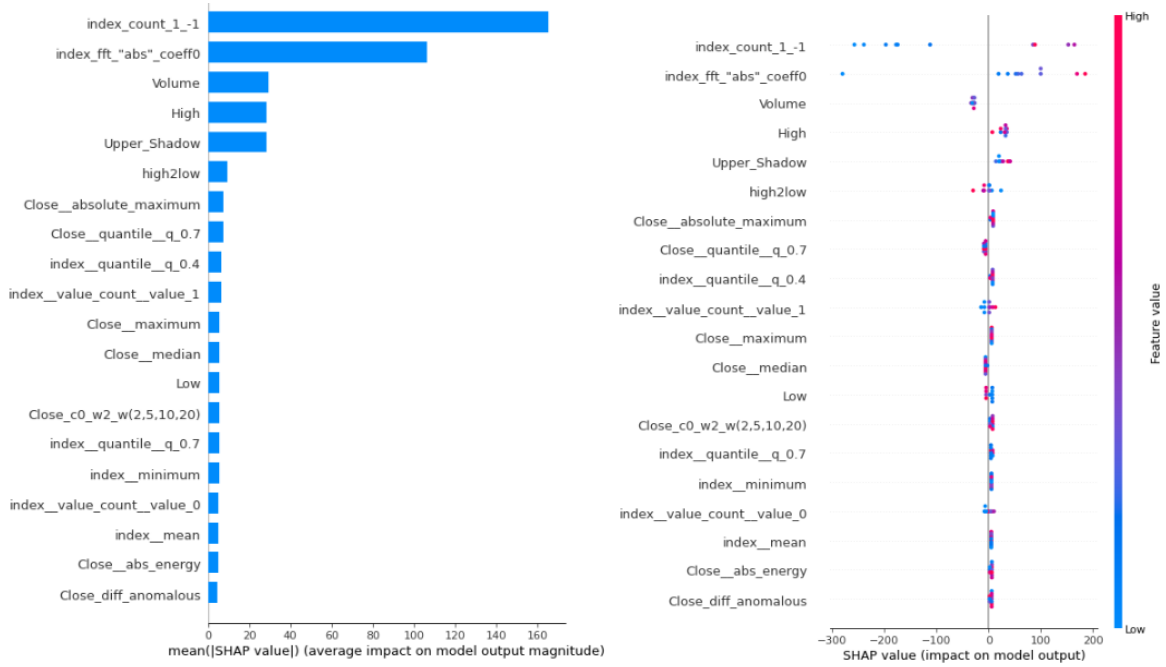
SHAP (англ. «SHapley Additive exPlanations») – це теоретико-ігровий підхід для пояснення передбачень будь-якої моделі машинного навчання з використанням класичних значень Шеплі з теорії ігор. Значення Шеплі визначають, на скільки зміна кожної ознаки впливає на передбачення моделі, тобто збільшує чи зменшує його значення.

Особливо цінним є те, що SHAP враховує усі можливі підмножини ознак та дозволяє враховувати взаємодію між ними. Це дозволяє створювати

інтерпретації, які відображають не тільки важливість окремих ознак, але і їх взаємовплив. Тобто це дозволяє розуміти причинно-наслідкові зв'язки і внесок кожної ознаки у прогноз.

Ефективними і популярними є такі діаграми SHAP:

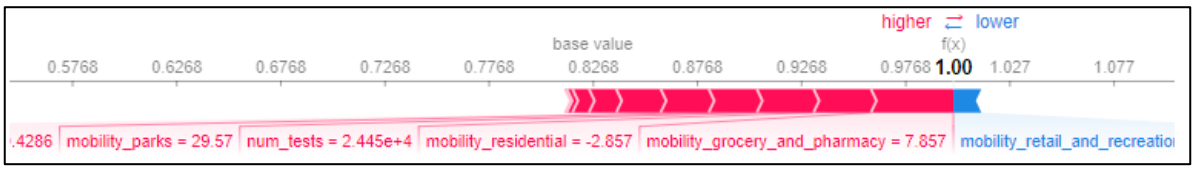
1. Summary Plot – діаграма, яка показує внесок кожної ознаки у кожен конкретний прогноз (рис. 3.4).



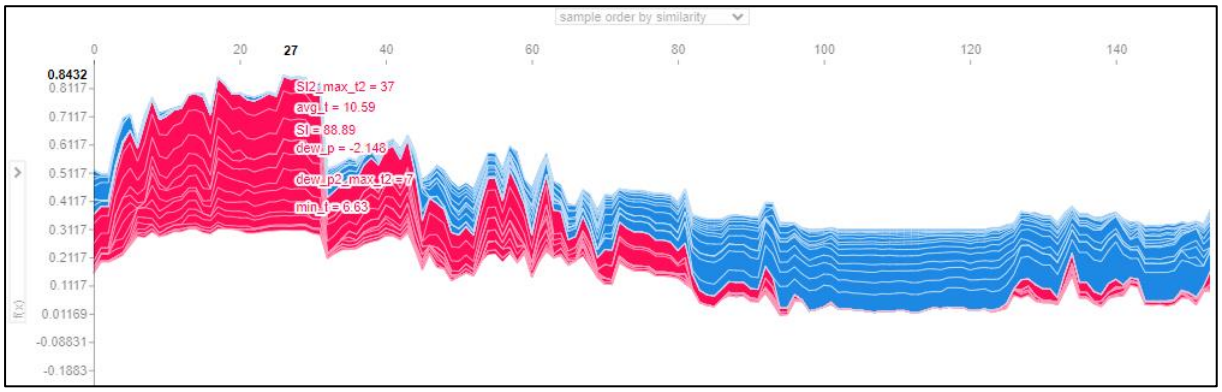
а) б)

Рисунок 3.4 – SHAP-діаграма Summary Plot для курсу біткоїна:
а) з параметром plot_type = «bar», б) з параметром plot_type = «dot» [18]

2. Force Plot – графік, який виводить розмір та знак внеску кожної ознаки для конкретного прогнозу, враховуючи базовий прогноз для всієї вибірки (рис. 3.5). Червоним відображаються ознаки, які впливають на збільшення таргета, а синім – ті, які – на зменшення. Цікаво, що можна будувати такий графік як для кожного передбачення окремо (рис. 3.5а), так і для усіх значень разом, тоді він матиме вигляд інтерактивного графіка, де можна вибирати різні ознаки (у випадних віконцях зверху і зліва) та діапазони (мишею на графіку) і більш ретельно вивчати закономірності (рис. 3.5б).



а)



б)

Рисунок 3.5. SHAP-діаграми Force Plot для приросту кількості хворих на коронавірус в Україні: а) для однієї дати 20.10.2020 р.; б) компілятивна інтерактивна діаграма на основі TreeExplainer-методу (для моделі на основі дерева рішень) для більш широкого набору ознак за квітень-жовтень 2020 р. (з авторського ноутбуку)

На діаграмі 3.5а видно, що за даними Google-трендів збільшення перебування людей у місцях проживання (mobility_residential) значно зменшує кількість нових хворих на коронавірус (цей показник показаний червоним, але є від'ємним, а отже, впливає на зменшення, а не – на збільшення), збільшення перебування в місцях відпочинку та роздрібної оптової торгівлі (mobility_retail_and_recreation) дещо менше сприяє зменшенню (показано синім, але внесок є у 3-4 рази меншим, за вплив mobility_residential. Інші показники показані червоним і є додатними, отже їх збільшення сприяє зростанню кількості нових хворих: мобільність у продуктових магазинах, у т.ч. супермаркетах, та аптеках (mobility_grocery_and_pharmacy), кількість тестів (більше спостерігають, то більше й виявляють – це логічно), мобільність у парках та лісах (mobility_parks) та ін. А на діаграмі 3.5б видно, що закономірності зазнавали суттєвих змін протягом усього діапазону спостережень, отже, для прогнозування у середньо- і довгостроковій перспективі ці ознаки можуть, скоріше, заважати (вносити додатковий шум), аніж допомагати підвищити точність прогнозування.

3. Dependency Plot – діаграма розсіювання, яка відображає, як зміна значення конкретної ознаки впливає на внесок цієї ознаки у прогноз (рис. 3.6).

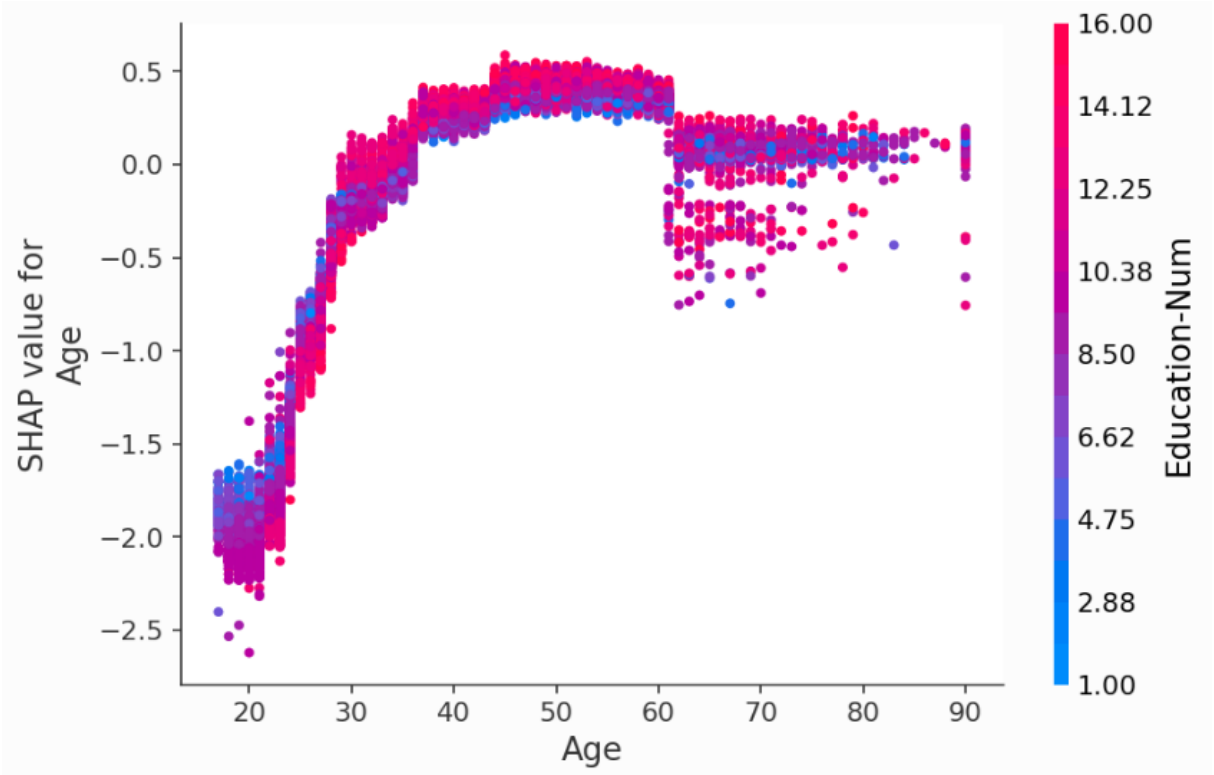


Рисунок 3.6. SHAP-діаграма Dependency Plot, яка ілюструє логарифм шансів заробити понад 50 тисяч доларів на рік, в залежності від віку людини та її освіти

На рис. 3.6 видно, що логарифм шансів заробити понад 50 тисяч доларів на рік значно зростає у віці від 20 до 40 років, досягаючи найбільших значень, за наявності максимально можливого рівня освіти, десь з 38 до 60, а максимальних – з 45 до, приблизно, 53 років.

4. Waterfall Plot: Графік, який ілюструє шлях, яким модель приходить до конкретного прогнозу, зазначаючи внесок кожної ознаки на кожному кроці у вигляді водоспаду приростів із різним знаком та кольором. Нижня частина діаграми водоспаду починається як очікуване значення результату моделі за вказаних на графіку сірими числами значень вхідних ознак, а потім кожен рядок на діаграмі показує, як позитивний (червоний) або негативний (синій) внесок кожної функції переміщує значення від очікуваного результату моделі до цього прогнозу (рис. 3.7).

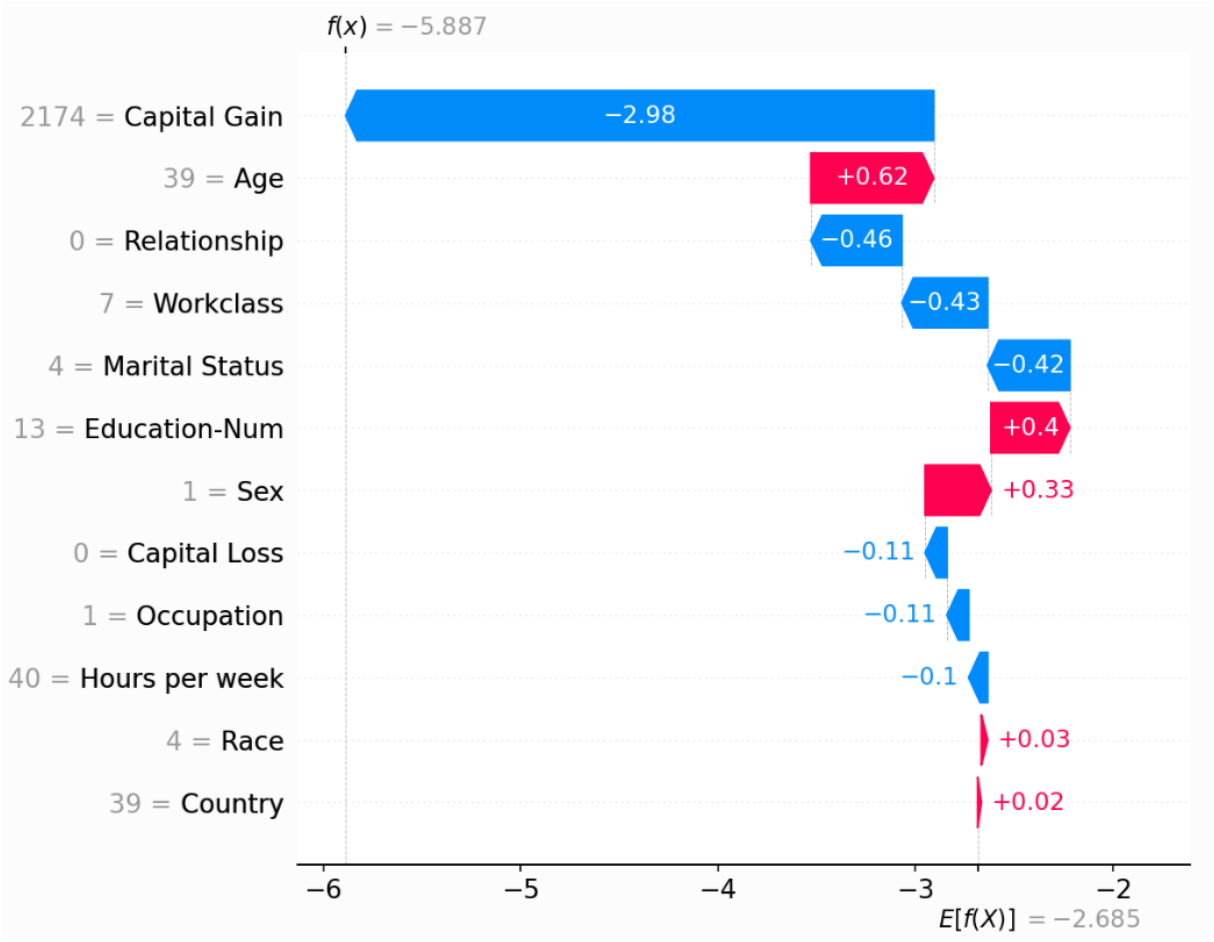


Рисунок 3.7. SHAP-діаграма Waterfall Plot, яка ілюструє характер і силу впливу певних значень ознак на логарифм шансів заробити понад 50 тисяч доларів на рік

На рис. 3.7 видно, що (див. опис значень параметрів у датасеті «US Adult Income Dataset», наприклад, у репозиторії чи в іншому місці): на зменшення доходів найбільше вплинула відсутність приросту капіталу; те, що цей афроамериканець (Sex=1, Race=4) ніколи не працював (Workclass); що він готовий працювати тільки на ручних роботах, пов'язаних з ремонтом (Occupation=1); що він – одинак (Marital Status = 4). Деяке зростання обумовлено віком, близьким до оптимального, коли, в середньому, люди заробляють найбільше (38-60 р.), що він є чоловіком (Sex=1), що в нього є освіта, хоча й незначна (Education_Num=13: лише 5-6-й класи).

Більш детально дивись інші приклади та їх опис [Data Science for tabular data: Advanced Techniques, Crypto - BTC : Analysis & Forecasting (розділ 3.4), або у статті або у документації].

Головним недоліком бібліотеки SHAP є те, що вона потребує багато обчислень, у разі, коли аналізуються великі та складні нейромережеві моделі чи ансамблі. Тому часто застосовують бібліотеку LIME, яка використовує ряд спрощень і тому, потребує менше обчислювальних витрат.

Основним принципом роботи бібліотеки LIME є те, що будь-яку складну модель легше апроксимувати в околиці конкретного прикладу даних.

Модель представляється як «чорний ящик». А її поведінка в заданій точці (як правило, це – багатовимірний вектор значень комбінації певних ознак) апроксимується лінійною моделлю і саме по ній формується пояснення усієї моделі в цілому. Таке спрощення дозволяє формувати пояснення та аналізувати вплив ознак на довільні прогнози для довільних моделей машинного навчання, що особливо цінно для нейронних мереж складної (часто невідомої) архітектури. На рис. 3.8 наведено приклад, який ілюструє цей принцип.

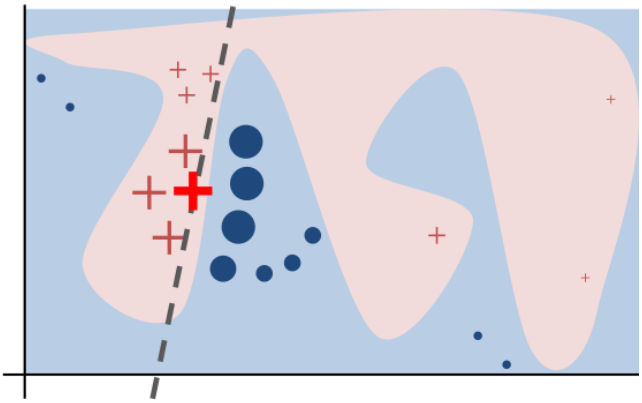


Рисунок 3.8 – Ілюстрація принципу роботи методу LIME [29]

Функція на основі моделі, яка пояснюється, представлена синім та рожевим кольором. Очевидно, що вона є нелінійною. Червоний великий хрестик є прикладом даних у точці X0. Беремо інші значення даних в певному околі біля X0, враховуючи міру близькості до X0 щодо ваги цих даних (на рис. 3.9 розмір хрестика є пропорційним такій вазі). Формується сукупність точок (хрестиків), яка далі апроксимується прямою (на рис. 3.8 – пунктирна лінія). Ця пряма дозволяє охарактеризувати певний зріз закономірностей, але це, скоріше – локальні, а не – глобальні закономірності. В цьому й полягає основний недолік методів цієї бібліотеки – в тому, що вона описує локальні закономірності, а не – глобальні, з використанням ряду наближень і спрощень. Однак, з використанням алгоритму SP-LIME можна отримувати інтерпретацію, хоч і локальних, але репрезентативних в глобальному сенсі, вибірок даних [23].

На рис. 3.9 і 3.10 наведено приклади з ноутбука щодо інтерпретації передбачень моделлю для пасажирів «Титаніка» виживуть вони чи ні за даними відомого змагання у Kaggle.

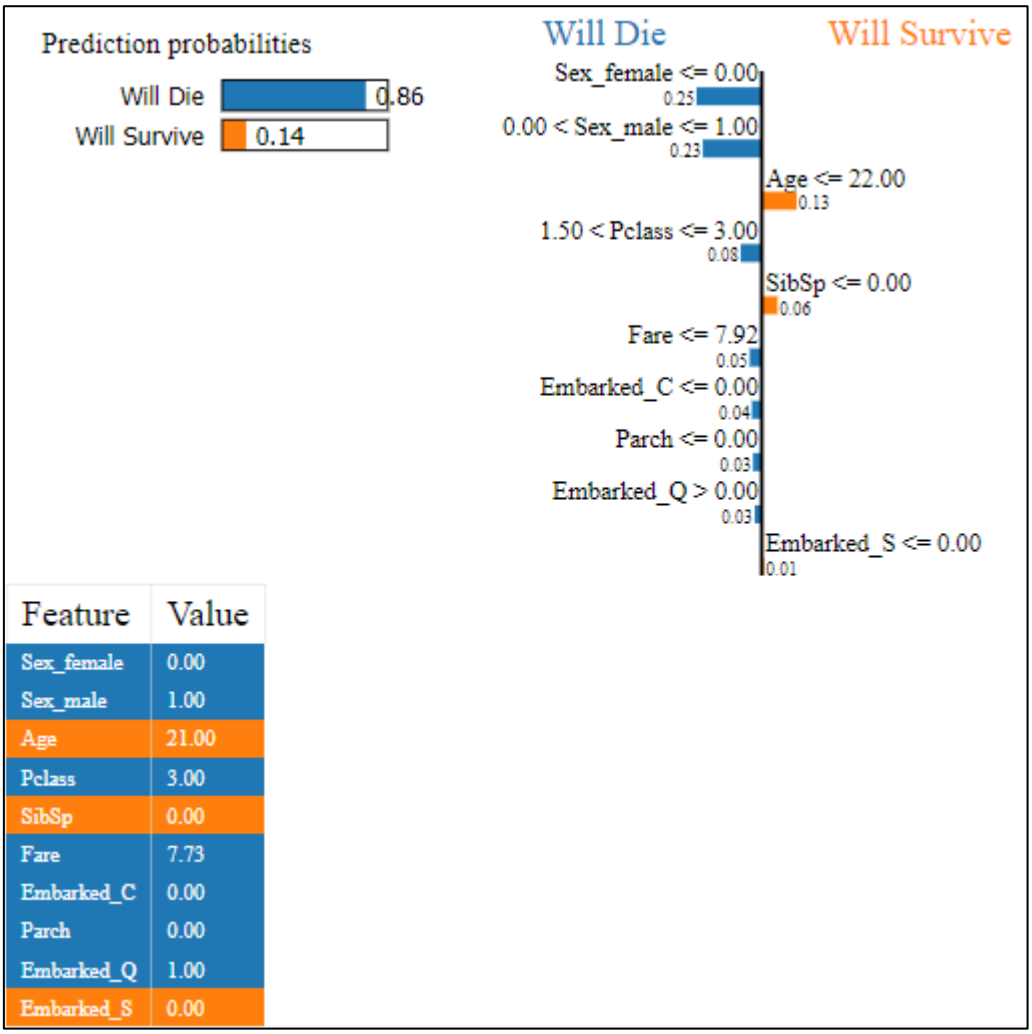


Рисунок 3.9 – Приклад інтерпретації передбачення чи виживе після катастрофи пасажир «Титаніка» з дешевим квитком 3-го класу, який сів в останньому порту – у Квінстауні (Queenstown – «Q»)

Як видно, на рис. 3.9, пасажир «Титаніка» з дешевим квитком 3-го класу, який сів в останньому порту – у Квінстауні, загине з ймовірністю 0,86. Той факт, що він їхав в трюмі з іншими пасажирами 3-го класу і був чоловічої статі, практично не залишило йому шансів на виживання. Крім того, як відомо з історії, ті, хто сіли в останньому порту (у Квінстауні), зайняли найменш зручні місця (далеко від трапу, з якого йшло чисте повітря), які ще були вільними. Хоча, деякі шанси йому дає те, що він – молодий (Age=21) і те, що у нього немає ні братів, ні сестер, ні дружини (SibSp=0), про яких треба турбуватися.

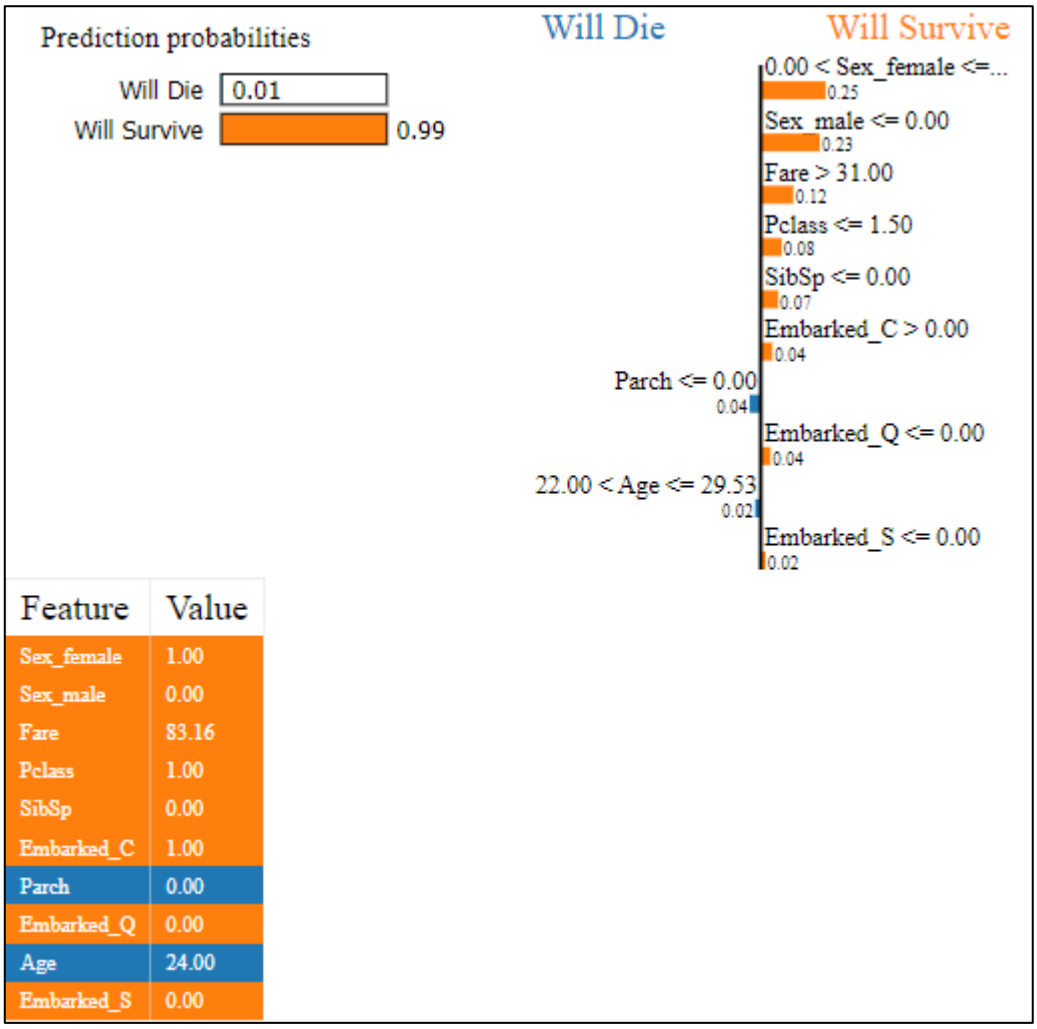


Рисунок 3.10 – Приклад інтерпретації передбачення чи виживе після катастрофи пасажирка «Титаніка», яка має дорогий квиток 1-го класу

А як видно з рис. 3.10, пасажирка «Титаніка» має усі шанси, щоб врятуватись. Цьому найбільше сприяє її жіноча стать та дорогий квиток 1-го класу. Не має особливого значення в якому порту вона сіла, оскільки пасажири 1-го класу займали окремі гарні каюти. Дещо дивно, що її шанси вижити збільшує відсутність братів, сестер чи чоловіка, адже чоловіки та брати рятували жінок і сестер, відповідно, в першу чергу. І також дивно, що на її шанси загинути впливає молодий вік (Age=24) та відсутність батьків (Parch=0). Адже екіпаж чи інші чоловіки рятували б таку молоду леді, а ще їй би не треба було піклуватись про своїх батьків. Вочевидь, ці дещо дивні результати демонструють, що в глобальному плані висновки бібліотеки LIME можуть бути не зовсім адекватним. Конкретно така жінка у тренувальному датасеті мала б шанси загинути, якщо б міцно спала, чи рятувала когось з дітей її знайомої родини, тобто через якісь ознаки, які відсутні у цьому датасеті. Це – приклад того, що по одному даному, яке може мати ознаки аномалії, важко робити глобальні висновки.

У табл. 3.2 наведено порівняльну характеристика бібліотек SHAP та LIME у сфері інтерпретації передбачень моделей машинного навчання та аналізу важливості їх фічерів.

Таблиця 3.2 Порівняльна характеристика методів бібліотек SHAP та LIME для задачі інтерпретації і візуалізації ознак моделі

Критерій	**SHAP**	**LIME**
Принцип роботи	Аналізує на скільки зміна кожної ознаки впливає на передбачення моделі для кожного прогнозу моделі з використанням теорії ігор та чисел Шаплі.	Генерує зразки, які є подібними до прикладу, обчислює прогнози та апроксимує важливість ознаки.
Тип моделей	Підтримує широкий спектр моделей, включаючи дерева рішень, лінійні моделі, нейронні мережі тощо.	Може застосовуватися до різних моделей, але найбільш ефективний для моделей типу «чорний ящик», таких як нейронні мережі.
Важливість ознаки	Глобальна та локальна важливість ознаки, залежно від типу діаграми.	Локальна апроксимація важливості ознаки. Але можна отримувати інтерпретацію й репрезентативних в глобальному сенсі вибірок даних
Переваги	Ефективний для великих даних та різноманітних моделей. Здатний розкривати взаємодії між ознаками. Гарна візуалізація результатів.	Працює як для табличних, так і для неструктурованих даних. Підходить для моделей довільної складності з невідомою архітектурою – типу «чорний ящик».
Недоліки	Може бути обчислювально витратним для великих моделей та даних.	Потребує багато зразків для стабільних результатів. Залежить від різноманітності даних.
Діаграми для відображення важливості ознак	Summary Plot, Force Plot, Dependency Plot, Waterfall Plot	Feature Importance Plot, Local Surrogate Model
Відображення взаємодії ознак	Interaction Values Plot	Відсутнє

Отже, кожна з бібліотек має свої переваги і свої недоліки – треба щоразу вибирати яку використовувати. Але можна застосовувати і усі підряд.

3.4 Інтелектуальний аналіз й оброблення ознак даних, проведені призерами змагань Kaggle

У Kaggle часто проводяться конкурси для розв'язання певних задач, де для перемоги треба вміти саме аналізувати важливість ознак та вміти здійснювати їх оброблення, у т.ч. генерування нових на основі наявних. Важливо, що це вміння є корисним для розв'язання, практично, будь-яких реальних задач, де є багато різноматнітних ознак. Розвідувальний аналіз може допомогти виявити проблемні місця, але усувати їх слід на етапі FE.

Задачі, що відносяться до етапу Feature Engineering (FE), це ті, які вимагають обробки та підготовки вхідних даних для подальшого використання у моделях машинного навчання. Зокрема, задачі FE можуть включати:

1. Вибір та інженерія ознак для передбачення ризику неповернення кредиту від Home Credit.
2. Розробка методів автоматичного виявлення особистої ідентифікаційної інформації (PII) в навчальних даних від The Learning Agency Lab.
3. Виділення ключових характеристик для передбачення поведінки просумерів в енергетичному секторі від Enefit.
4. Обробка та агрегація даних з ордерного блоку та заключної аукції для передбачення рухів цін акцій в заключні хвилини торгів від Optiver.
5. Виділення важливих ознак для передбачення хімічних впливів на різноманітні типи клітин від Open Problems – Single-Cell Perturbations.

IEEE-CIS Fraud Detection основна задача конкурсу полягає в поліпшенні системи запобігання шахрайству з використанням методів машинного навчання. Учасники будуть аналізувати великі обсяги даних, що містять інформацію про транзакції електронної комерції, та розробляти та вдосконалювати моделі машинного навчання для виявлення фальшивих транзакцій. Метою конкурсу є підвищення точності виявлення шахрайських операцій, щоб зменшити збитки від шахрайства та покращити враження клієнтів від обслуговування.

Перше місце здобув учасник з ноутбуком XGB Fraud with Magic - [0.9600] основною частиною стратегії була комбінація трьох окремих моделей з високими результатами: LGBM і XGB та ні. Ці моделі були різноманітними, а також розроблені власні функції для їх підтримки, що зображено на рисунку 3.11.

Також використані два різні методи для пошуку UID (унікального ідентифікатора користувача) та ефективно використовували їх у моделі. Один з них полягав у створенні сценарію для знаходження UID, а інший - у тренуванні моделей для знаходження UID. Це дозволило покращити результати моделей і відповідно збільшити точність передбачень.

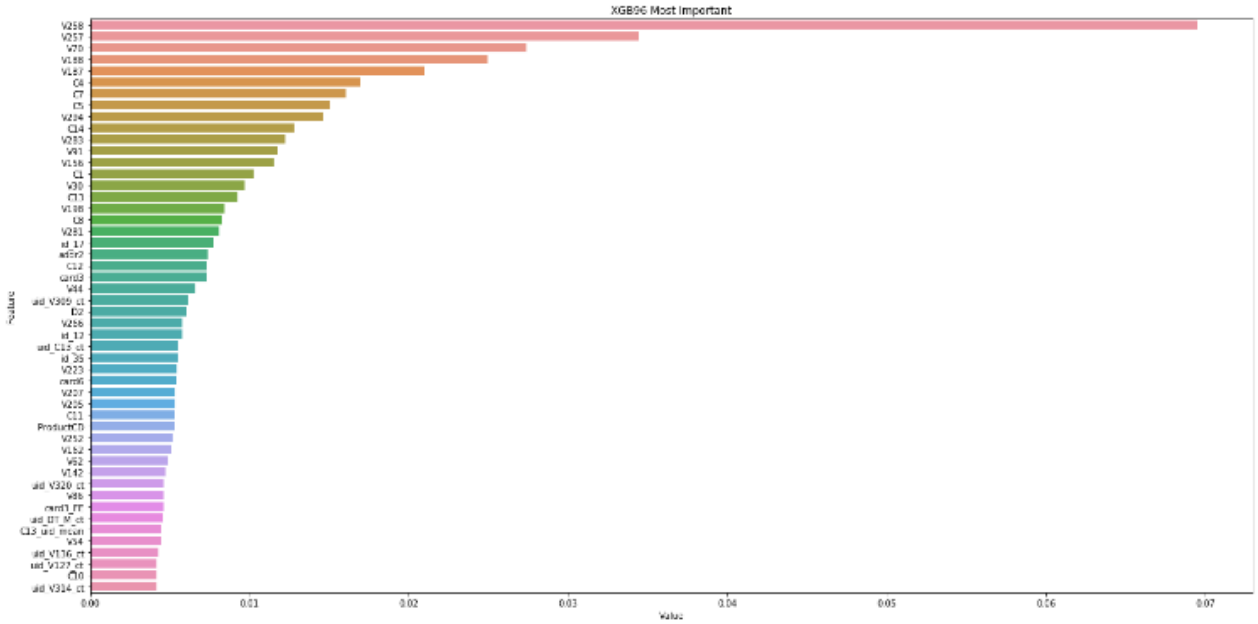


Рисунок 3.11– Важливість ознак з ноутбука [XGB Fraud with Magic - [0.9600]]

У конкурсі [Google Cloud & NCAA® ML Competition 2019-Men's], учасники мають можливість використовувати історичні дані NCAA Division I Men's Basketball Championships для створення та тестування моделей передбачення результатів. У першому етапі учасники будуть використовувати дані попередніх турнірів для побудови та тестування моделей, а в другому етапі - передбачати результати всіх можливих зіставлень у 2019 році.

Основна стратегія [команди], яка посіла третє місце в конкурсі, полягала у використанні моделі логістичної регресії для передбачення ймовірності перемоги команд у турнірі March Madness. Для ігор, команда випадковим чином вибирала переможця: у першій подачі вигравала команда з більшим ID, у другій - з меншим. Цей підхід дозволив їм знизити значення logloss з 0,473 до кінцевого показника 0,427, що зображено на рисунку 3.12.

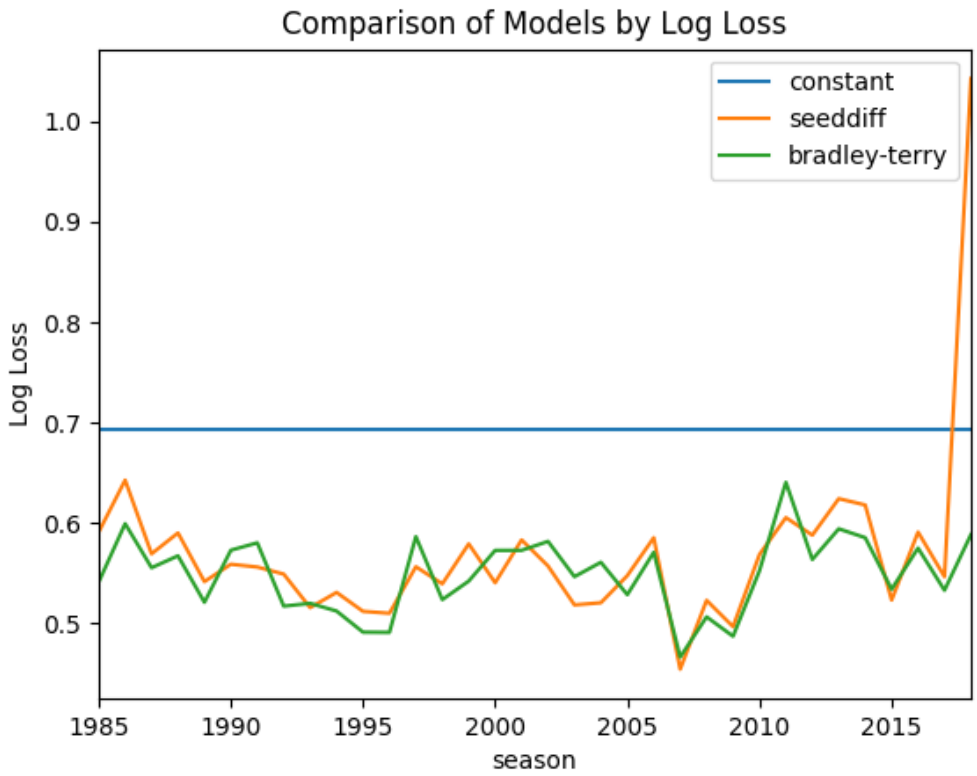


Рисунок 3.12 – Графік Log Loss декількох моделей в ноутбуці Bradley-Terry Model

Основною моделлю команди була логістична регресія з двома ознаками: TeamID та кількість очок, які кожна з команд виграла протягом звичайного сезону. Додатково вони використовували розширену модель логістичної регресії Bradley-Terry для оцінювання рівня кожної команди на основі результатів звичайних сезонних ігор.

Команда припускала, що непередбачувані ігри є непередбачуваними, тому вони випадково вибирали переможця для таких ігор. Цей стратегічний хід сприяв покращенню їхнього результату на конкурсі.

Нарешті, команда документувала своє рішення і код, який вони використовували, і робила це доступним для публіки за допомогою GitHub.

Учасник з команди Sharp Sports Analytics, Jordan Sundheim, використав ретельний підхід до відбору ознак у своїй моделі для участі в Google Cloud & NCAA ML Competition 2019-Men's. Він визначив, що ключовими факторами для його моделі є ефективність атаки та оборони, а також – інші параметри, що можуть впливати на результати матчів. Під час відбору ознак він використав три різні підходи:

1. Він розпочав з базового підходу, використовуючи лише ефективність атаки та оборони, які вже були документовані як ключові фактори успіху у спортивних аналітичних моделях.

2. Потім він вирішив перевірити всі доступні статистичні дані «з коробки» (наприклад, очки, підбирання, передачі тощо) і перевірити, як вони впливають на результати гри.

3. Для третього підходу він додав деякі додаткові фактори, такі як відсоток перемог за рахунками по розподілу, серії перемог (прямі перемоги), та попередні рейтинги перед початком сезону. Ці параметри він вибрав, припускаючи, що вони можуть мати значення для спортивних аналітичних моделей.

Після вибору ознак він створив кінцеву модель, яка була зваженою комбінацією цих трьох підходів. Важливим аспектом його методики було, також, коригування статистики для врахування сили суперників, що дозволило йому отримати більш точні прогнози результатів матчів.

Під час створення рекомендаційних систем часто важливо знати і враховувати побільше інформації про об'єкт чи суб'єкт, для якого формуються рекомендації. А для цього слід вміти з цієї інформації формувати побільше дійсно корисних фічерів. Це продемонстрував конкурс «OTTO – Multi-Objective Recommender System», де треба було розробити рекомендації покупцям у великих Інтернет-магазинах. Рішення-переможці основну увагу приділяли саме етапу FE. Учасник, що зайняв перше місце, описав стратегію у пості, що зображено на рисунку 3.13.

1. Учасник створював близько 1200 кандидатів, використовуючи інформацію про відвідуваність окремих ID у сесіях користувачів, матрицю співвідвідуваності, та низку моделей машинного навчання, таких як нейронні мережі та моделі градієнтного бустінгу (LGBM). Використання різних варіантів зважування та періодів агрегації, а також застосування матриці співвідвідуваності на різних етапах, таких як пошук по засічці (beam search), сприяло покращенню результатів.

2. Для покращення порядку рекомендацій було використано ансамбль з 9 моделей LGBMRanker з різними гіперпараметрами. Оцінювання ранжування проводилась за допомогою різних фічерів сесії та ID, включаючи інформацію про популярність ID, схожість за критерієм косинусної подібності, дублювання ID у сесії та інші.

3. Було створено близько 200 ознак для кожної сесії, включаючи характеристики сесії (такі як тривалість та частота дублювання ID), інформацію про ID (такі як популярність та тип), та зв'язок між сесією та ID.

4. Для тренування моделей було використано вибірки з негативними рейтингами, щоб збалансувати обсяги даних та підвищити швидкість обробки.

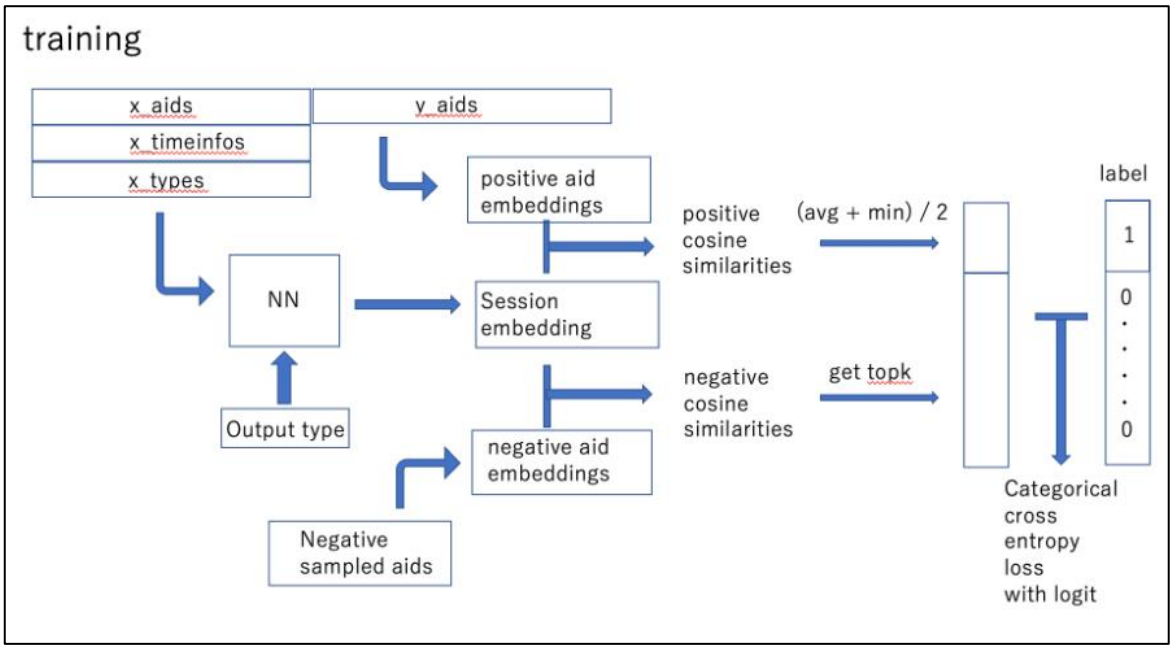


Рисунок 3.13 – Алгоритм роботи методу, що зайняв перше місце в конкурсі «OTTO – Multi-Objective Recommender System»

Важливо враховувати, що для ефективного розв'язання реальної задачі слід добре розбиратись у предметній області. Наприклад, автор рішення, яке посіло 3-тє місце у призовому конкурсі «Novozymes Enzyme Stability Prediction», у своєму пості наводить ряд тригонометричних співвідношень, як він рахував нові фічери на основі наявних.

На рис. 3.14 наведена інфографіка інструментарію, згаданого у розділі 3 в цілому, у системі координат *S(I)*.

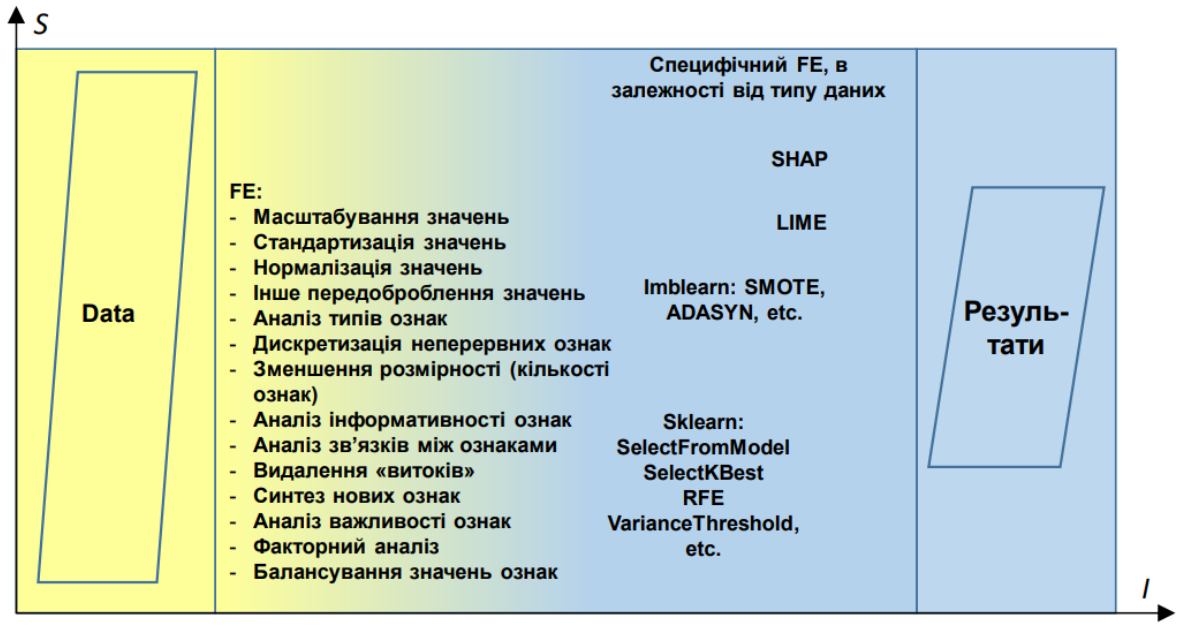


Рисунок 3.14 – Інфографіка усіх основних операцій і технологій інженерії ознак (FE)

Можливі теми практичних і лабораторних завдань

Тема № 1. Аналіз важливості ознак табличних даних.

Метою заняття є роботи є вивчення інформаційних технологій і Python-бібліотек для аналізу важливості ознак та проведення оброблення ознак (англ.: Feature Engineering) табличних даних й опанування практичних навичок застосування деяких із них на прикладі одного із датасетів Kaggle чи за даними, завантаженими через API.

План заняття:

1. Вибрати датасет (див. розд. 1).
2. Провести первинний EDA датасету та визначити завдання для FE.
3. Охарактеризувати етапи оброблення ознак (які видалено наявні і чому, які створено нові і для чого, які трансформовані і чому).
4. Навести діаграму важливості ознак однієї з моделей та які по ній можна зробити висновки.
5. Зазначити, які оптимальні ознаки було відібрано на основі вивчення результатів пп. 3, 4 (ці пункти можуть застосовуватись почергово і неодноразово: 2, 3, 2, 3…). Також, може бути наведена об'єднана (чи комплексна) діаграма важливості ознак, побудована за декількома моделями одночасно – див. приклад у ноутбуках:

- FE - Feature Importance - Advanced Visualization (для класифікаційної задачі)
- FE-FI for Regression Task - Advanced Visualization (для регресійної задачі).

Приклади ноутбуків:

Для виконання цієї роботи рекомендується використовувати такі ноутбуки:

- для класифікаційної задачі: FE - Feature Importance - Advanced Visualization (для використання цього ноутбуку для регресійної задачі слід в усіх місцях, де в коментарях написано «for regression task», замінити відповідний параметр на те, що там же рекомендовано);

- для регресійної задачі: FE-FI for Regression Task - Advanced Visualization (регресійна задача прогнозування якості води на основі датасету River Water Quality EDA and Forecasting щодо р. Південний Буг за даними 20 років, сформованого за даними Держводагентства і data.gov.ua (для цілочисельного таргета моделі з відповідною метрикою забезпечують більш точні прогнози);

- підвищений рівень складності: Heart Disease - Automatic AdvEDA & FE & 20 models.

Також, можна подивитись інші приклади ноутбуків із діаграмами важливості ознак:

- Autoselection from 20 classifier models & L curves
- Biomechanical features - 20 popular models
- Suspended substances prediction in river

- WQ SB river : EDA and Forecasting

Контрольні питання

1) Що включає в себе інженерія ознак (FE) та чому вона є важливою у процесі аналізу даних та побудови моделей?
2) Які є основні етапи інженерії ознак, і які задачі вони вирішують?
3) Які методи можна використовувати для синтезу нових ознак на основі наявних даних?
4) В чому полягає стандартизація та нормалізація ознак, і чому це важливо перед побудовою моделі?
5) Як можна побудувати діаграму важливості ознак, і яка її роль у виборі найбільш важливих ознак для моделі?
6) Які бібліотеки Python можна використовувати для автоматизації вибору ознак, та які методи вони надають для цього?
7) Як забезпечити інтерпретабельність моделей машинного навчання з точки зору інженерії ознак?
8) Які інтелектуальні методи інженерії ознак часто використовуються призерами змагань на Kaggle?
9) Які стратегії використовуються для оброблення категоріальних ознак під час інженерії ознак?
10) Які можливі проблеми можуть виникнути під час інженерії ознак, і як їх можна уникнути або вирішити?

4 ПОБУДОВА МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

4.1 Види моделей машинного навчання, їх навчання та регуляризація

4.1.1 Види моделей та їх переваги

Моделі машинного навчання поділяють на такі *основні класи* (див. документацію бібліотеки Sklearn):

1) Лінійні моделі та методи їх ідентифікації (пакет sklearn.linear_model - біля 20 моделей і методів), найбільш поширені:
   - LinearRegression - лінійна регресія;
   - Ridge – гребенева регресія;
   - Lasso - регресія Лассо;
   - LogisticRegression - логістична регресія (саме так, вона є нелінійним перетворенням, але вона входить до пакету лінійних моделей);
   - Stochastic Gradient Descent – метод стохастичного градієнтного спуску для ідентифікації лінійних моделей;
2) Моделі на основі методу опорних векторів (пакет sklearn.svm – декілька моделей, але з багатьма варіаціями), основні:
   - SVC;
   - LinerSVC;
3) Моделі на основі методу найближчих сусідів (пакет sklearn.neighbors – біля 10 моделей), основна:
   - NearestNeighbors – дуже ефективна та поширена в задачах у сфері економіки та торгівлі, коли треба враховувати як себе поводять певні класи вхідних даних і як співвідносяться з таргетом;
4) Методи прогнозування на основі процесів Гаусса та коли прогнозуються на самі значення, а – їх імовірнісні характеристики (пакет sklearn.gaussian_process - декілька моделей, але з багатьма варіаціями), основна:
   - GaussianProcess;
5) Модель наївного Баєса (пакет sklearn.naive_bayes - декілька моделей), основна:
   - GaussianNB;
6) Дерева рішень;
7) Ансамблі моделей:
   - ансамблі дерев рішень:
   - багінг;
   - бустінг;
   - стакінг;

- голосування;

8) Нейронні мережі та їх ансамблі:
    - перцептрон Sklearn: Perceptron – найпростіша і найпримітивніша нейронна мережа з одним прихованим, є лінійною моделлю (усі більш складні нейромоделі є нелінійними), має більше історичне, аніж – прикладне значення;
    - багатошарова нейронна мережа Sklearn: MLP (Multi-layer Perceptron);
    - нейронні мережі фреймворків Keras, TensorFlow та PyTorch, яким присвячений весь наступний розділ 5.

Найбільш ефективними та популярними є 2 останніх класи, але, щоб їх краще зрозуміти, потрібно спочатку оволодіти першими 6-ма.

З особистого досвіду авторів та за результатами вивчення багатьох рішень і літератури у різних сферах застосування:

- лінійні моделі гарно працюють для малих чи дуже зашумлених датасетів, коли інші моделі не є ефективними (хоча це, часто, може означати хибне рішення, яке ще потребує ретельно перевірки);
- логістична регресія швидко працює в задачах класифікації і часто використовується для післяоброблення у задачах аналізу природномовного тексту з використанням нейронних мереж;
- моделі на основі методу найближчих сусідів часто є ефективними в задачах у сфері економіки, коли треба досліджувати як різні групи (кластери) вхідних даних пов’язані з таргетом, з урахуванням якихось своїх особливостей, при цьому, вона швидше за інші і працює за малих датасетів;
- моделі на основі процесів Гаусса є ефективними, коли вхідні дані не дають повного уявлення про класи вхідних даних, а лише окреслюють їх і тоді, за рахунок агрегування в 2 статистичні параметри функції Гаусса (середнє значення та середньоквадратичне відхилення) можна моделювати ці класи в цілому, тобто за малих вибірок різноманітних даних отримувати модель, яка зможе узагальнювати дані у широкому діапазоні входів – це особливо ефективно працює для класифікації природномовного тексту, коли тренувальний датасет – порівняно невеликий, але – різнорідний;
- дерева рішень дозволяють гарно розібратись в закономірностях даних, мають гарні можливості щодо візуалізації, відбору ознак за їх важливістю.

Ансамблі є найбільш ефективними, але вони основані на цих моделях. Там головне, вдало підібрати параметри та відібрати моделі для ансамблю. Якщо він є послідовним, тоді ще треба правильно задати послідовність.

Нейронні мережі та їх ансамблі потребують більших датасетів, ніж лінійні моделі чи дерева рішень, але є значно більш ефективними для великих даних.

4.1.2 Поняття навчання моделі та гіперпараметрів

Кожна модель машинного навчання, наприклад деяка модель **name_model** (з параметрами **model_params**) пакету **name_package** бібліотеки **sklearn** за навчальними даними у вигляді датафрейма **train** та таргетом у вигляді датафрейму з одним стовпцем типу «series» (або просто у вигляді списку) **target**, ідентифікується в однаковий спосіб:

```
from sklearn.name_package import name_model
model = name_model(model_params)
model.fit(train, target)
```

а тоді передбачення значень *y_pred* за даними тестового датафрейму **test** здійснюється таким чином (хоча є можливими й інші варіанти, наприклад, коли треба передбачити ймовірність появи значень, а не самі значення: *predict_proba*):

```
y_pred = model.predict(test)
```

Ідентифікована модель є об'єктом *model* складеного типу. За його допомогою спеціальними командами можна переглянути оптимальні ідентифіковані параметри, важливість фічерів тощо.

Далі будемо наводити по кожній моделі тільки значення **name_package, name_model** та приклади **model_params**.

Практично усі моделі, окрім лінійної регресії, регресії Лассо, GaussianNB, в бібліотеці sklearn мають варіант для задачі класифікації (в кінці назви додається слово «Classifier») і для задачі регресії («Regressor»). Не існує варіанту моделі LinearClassifier, LassoClassifier чи GaussianNBClassifier.

Логістична регресія – єдина регресія, яка не може застосовуватись до розв'язання регресійних задач. Вона призначення тільки до класифікаційних задач, не дивлячись на назву.

В усіх моделях, як правило, є параметр random_state, який обов'язково слід фіксувати, наприклад: random_state=42, або 0, 1, 2 чи інше ціле число. Це забезпечує відтворюваність результатів, інакше результат розрахунку не можна буде повторити.

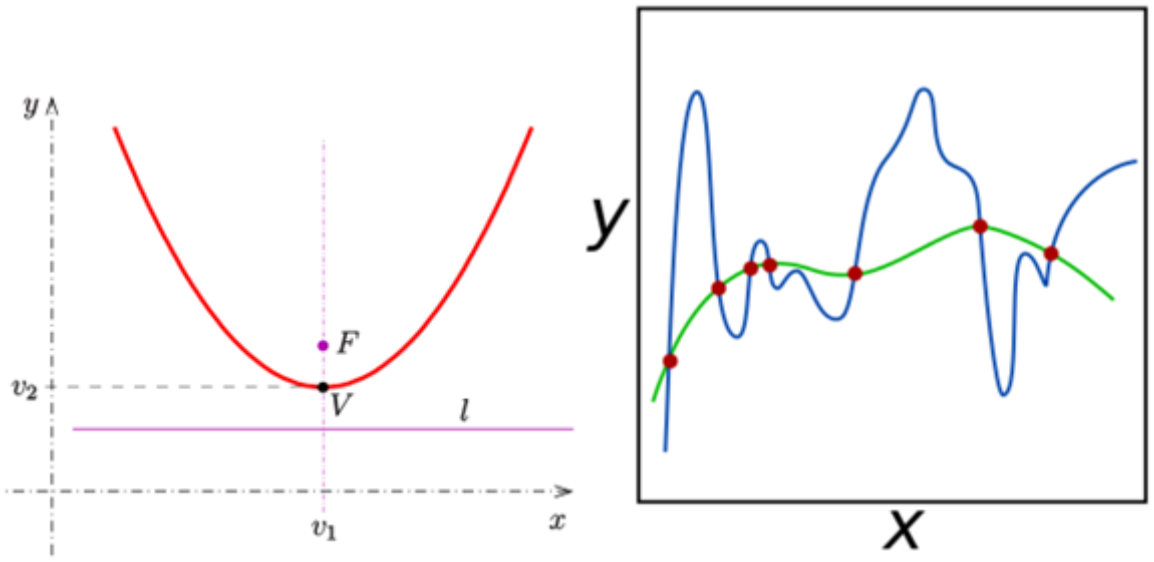
Параметр verbose, який, як правило, приймає значення 1 або 0, означає виводити чи ні проміжні результати розрахунку, відповідно. Для складних моделей на кшталт нейронних мереж чи ансамблів, може бути й більше значень, наприклад: 0 – не виводити нічого, 1 – виводити все, 2 – виводити у скороченій формі (більш точно дивіться у документації). Коли модель тільки будується і проводяться експерименти, тоді варто задавати verbose=1, а якщо модель вже оптимізована і використовується для передбачень, тоді задають verbose=0.

Усі моделі, окрім лінійної регресії, як правило, мають певні **model_params**. У машинному навчанні їх називають *гіперпараметрами* – це параметри, які не навчаються самі по собі під час навчання моделі, але визначають архітектуру чи конфігурацію моделі. Вони встановлюються перед початком навчання і визначають, яким чином модель повинна навчатися та як вона повинна адаптуватися до даних.

У підрозділі 4.3 буде більш докладно пояснено як саме слід навчати модель та автоматизувати визначення гіперпараметрів.

4.1.3 Регуляризація задля мінімізації ризику перенавчання

Важливо так налаштовувати гіперпараметри, щоб не допускати чи мінімізувати ризик перенавчання (англ. «overfitting»). Для цього застосовується *регуляризація* – прийом в машинному навчанні для мінімізації значень параметрів моделі з метою покращення її узагальнюючої здатності та уникнення перенавчання. Найкраще її сутність пояснити на ідентифікації поліноміальної регресії для одного фічера (рис. 4.1).



а) б)

Рисунок 4.1 Приклади побудови поліноміальної регресії: а) парабола (поліном n=2 порядку), яка точно проходить через n+1=3 точки; б) поліноми m=6-го і меншого порядків, які наближені до m+1=7 точок

Як відомо, через n=2 точки можна точно провести пряму, тобто – поліном n-1=1 порядку. Через n=3 точки – параболу, тобто – поліном n-1=2 порядку (рис. 4.1а), через n=7 точок – поліном n-1=6 порядку (рис. 4.1б, лінія синього кольору). Однак, ця дуже точна крива буде сильно відрізнятись від основного тренду лінії між відомими точками. Теоретично, параметри можуть бути знакозмінними і мати значення 10^6 чи 10^7. Через що, коли надійдуть тестові дані зі значеннями, які будуть відрізнятись від тренувального датасету, то модель дасть сильно неадекватні результати. Щоб цього уник-

нути, запроваджують обмеження штучне значень параметрів. Вочевидь, необхідно обмежувати і позитивні, і негативні значення, а тому, використовують 2 види функцій 1 і 2 порядку з відповідним позначенням L1 і L2:

- L1: функція модулю:

$$\min_w \frac{1}{2n_{\text{samples}}} ||Xw - y||_2^2 + \alpha||w||_1 \tag{4.1}$$

- L2: функція квадрату:

$$\min_w ||Xw - y||_2^2 + \alpha||w||_2^2 \tag{4.2}$$

Практично усі моделі машинного навчання мають 3 параметри: L1 та L2, які приймають значення True чи False, або – один (наприклад «*penalty*»), що приймається значення L1, L2, None. Третім параметром є ступінь регуляризації (як правило, позначається як «*alpha*», якщо більші значення – більша регуляризація, або як «*C*», якщо – навпаки) – позитивне число типу float. Бувають моделі й з обома видами регуляризації одночасно, наприклад ElasticNet.

Розглянемо основні моделі перших 7 класів більш детально. Ансамблі будуть розглянути у наступному підрозділі, а нейромережам присвячений окремий розділ 5.

4.1.4 Лінійна регресія

Лінійна регресія (*linear regression*) – метод, який описує взаємозв'язок між n фічерами (вхідними даними) $x_1, \ldots, x_n$ і таргетом y (вихідною ознакою) за допомогою лінійних функцій (рис. 4.2).

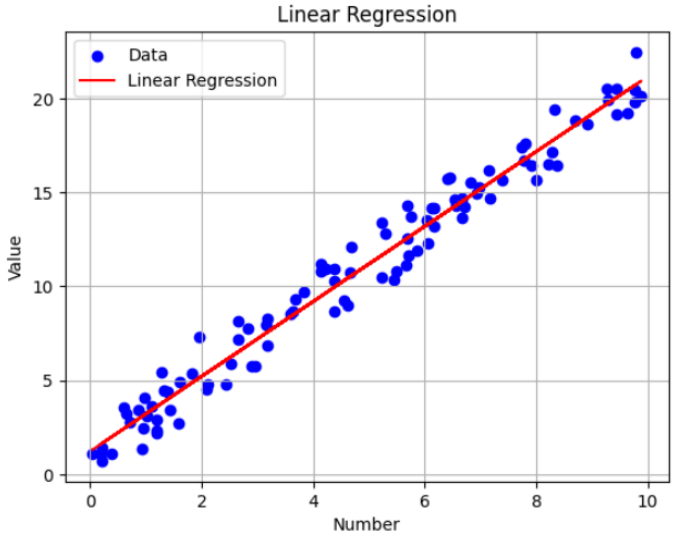


Рисунок 4.2 Лінійна регресія з авторського ноутбуку

Може застосовуватись як для ознак із цілими (класифікаційна задача), так і з дробовими числами (регресійна задача), не змінюючи, при цьому, назву:

name_package = linear_model

name_model = LinearRegression()

Математично ця модель має вигляд:

$$y = a_0 + a_1 x_1 + \cdots + a_n x_n + \alpha, \quad (4.3)$$

де a_0 – стала складова (зсув), $a_1, \ldots, a_n$ – коефіцієнти важливості фічерів, α – випадковий шум з нульовим середнім і фіксованою дисперсією, меншою, ніж у фічерів.

Результатом є a-коефіцієнти моделі, які відображають важливість відповідних фічерів та значення зсуву. Саме їх аналізу присвячений розділ 3.

Сама лінійна регресія не має параметрів. Але має 2 популярні різновиди, залежно від виду регуляризації:

1. Регуляризація Лассо (Lasso) – лінійна регресія з L1-регуляризацією (4.1).

name_package = linear_model

name_model = Lasso (регресор)

model_params = *alpha=1.0, *, fit_intercept=True, precompute=False, copy_X=True, max_iter=1000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic'*

Модель Лассо, як і лінійна регресія, не має окремого варіанту для класифікаційних задач. Для них вона застосовується в такому ж вигляді.

2. Гребенева регуляризація (Ridge) – лінійна регресія з L2-регуляризацією (4.1).

name_package = linear_model

name_model = Ridge (регресор), RidgeClassifier (класифікатор)

model_params = *alpha=1.0, *, fit_intercept=True, copy_X=True, max_iter=None, tol=0.0001, solver='auto', positive=False, random_state=None*

Параметри для регресора і класифікатора Ridge – однакові. Більші значення *alpha* – більша регуляризація.

Встановлювати в обох видах моделей *alpha* = 0 не рекомендується. Замість цього, краще використовувати модель LinearRegression.

Головними параметрами, які варто варіювати для підвищення точності, є такі:

- *tol* – точність (додатне float), за якої зупиняється алгоритм;
- *max_iter* – максимальна кількість ітерацій, якщо *tol* не буде досягнуто;
- *solver* – алгоритм оптимізації.

Часто регресія Лассо використовується для відбору ознак (FS), оскільки може зануляти важливість неважливих ознак, тобто прибирати їх (рис. 4.3).

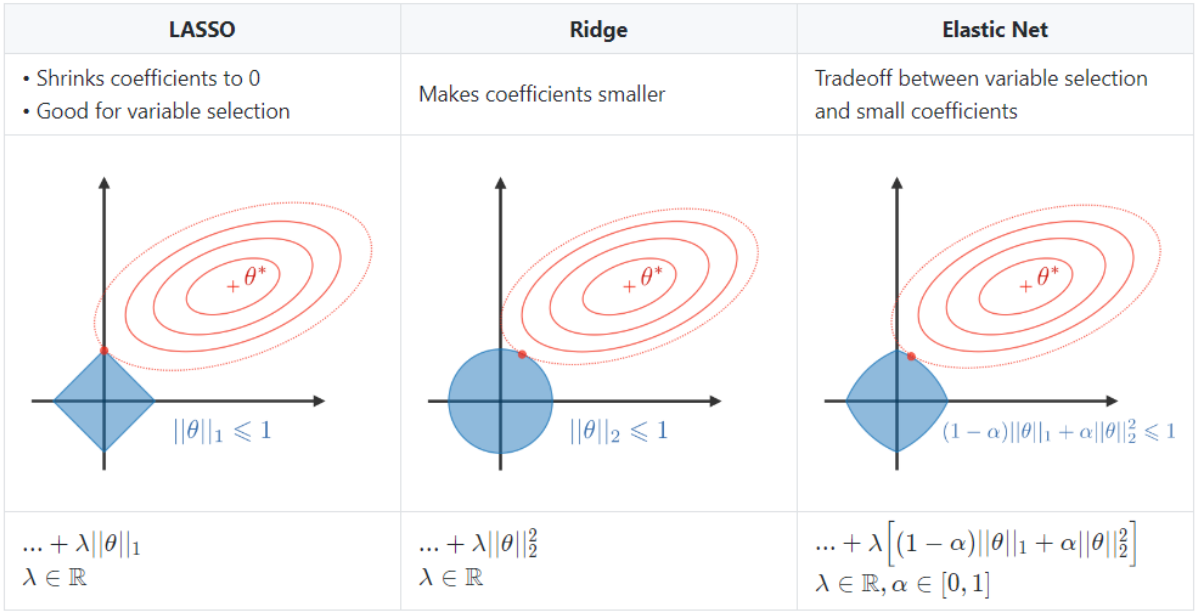


Рисунок 4.3 – Зони обмежень для коефіцієнтів, які задаються параметром регуляризації alpha: а) регресія Лассо (LASSO); б) гребенева регресія (Ridge); в) змішаний варіант регуляризації типу «Elastic Net»

Як видно на рис. 4.3, для двовимірного простору фічерів побудовані зони обмежень *alpha*. Якщо зменшувати регуляризацію *alpha*, тоді червоні еліпси зі значеннями збільшаться і, врешті-решт, потраплять у центр кола (рис. 4.3б). Це – той випадок, коли гребенева регресія і регресія Лассо перетворюються на лінійну регресію. В іншому випадку (рис. 4.3а) – обидва способи визначають коефіцієнти, знаходячи першу точку, де червоні еліптичні потрапляють в область («торкаються») обмежень. Регресія Лассо має кути на осях, на відміну від гребеневої регресії, і кожного разу, коли еліптична область потрапить у таку точку, один з фічерів повністю зникає (його важливість дорівнює нулю)!

Є ще один підвид цих регресій і видів регуляризації, коли одночасно застосовуються обидва види регуляризації L1 та L2 (рис. 4.3в). Це – модель sklearn.linear_model.ElasticNet. Вона є корисною, коли деякі фічери сильно корелюють, але видаляти їх не варто (модель Лассо залишить один такий, ElasticNet – усі).

Важливо відмітити, що L1, L2, ElasticNet – це не тільки моделі, а й – варіанти значень параметра (наприклад «penalty»), який в багатьох моделях відповідає за вид регуляризації. Як правило, ще є й інший параметр, який відповідає за силу регуляризації, залежно від вибраного виду регуляризації, але про це буде нижче.

На рис. 4.4 наведено приклад передбачення 5 різних датасетів різними видами лінійної регресії з різною регуляризацією.

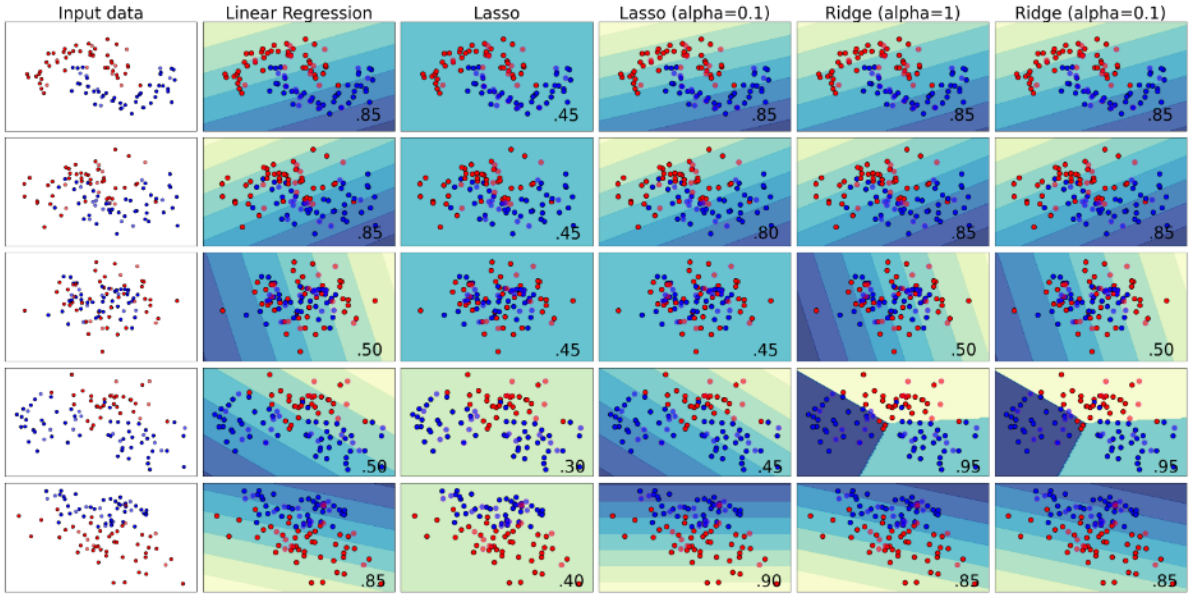


Рисунок 4.4 – Класифікація даних 5 різних датасетів різними видами лінійної регресії з різною регуляризацією (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [24]

Як видно на рис. 4.4, на прикладі Lasso, менший рівень регуляризації alpha збільшує точність моделей, але на деяких датасетах точність моделей є дуже низькою. Порогом мінімально допустимої точності умовно вважаються значення, більші за 0,5, оскільки, якщо не проводити жодне передбачення, а просто випадково генерувати відповіді, то ймовірність, теоретично, сягне якраз 0,5. Тому, якщо точність вища, то це вже краще. Задовільним рівнем, зазвичай, є 0,7. Відмінним – 0,9. Але є задачі, де і 0,9999 може бути замало, особливо на змаганнях датасайнтистів в Kaggle. Якщо точність є меншою 0,5, то це – неприйнятно!

На рис. 4.4 однією з найкращих моделей за точністю на тестових даних є Ridge(alpha=0.1), принаймні на першому (0,85) і останньому (0,9) датасетах.

4.1.5 Логістична регресія

Логістична регресія (англ. «logistic regression») — статистичний регресійний метод, оснований на логістичній функції, який застосовують для передбачення таргета y, за випадку, коли вхідні змінні $x_1, \dots, x_n$ є категорійними, тобто можуть набувати фіксовану кількість значень (2-10, рідко більше). Застосовується тільки для класифікаційних задач (рис. 4.5).

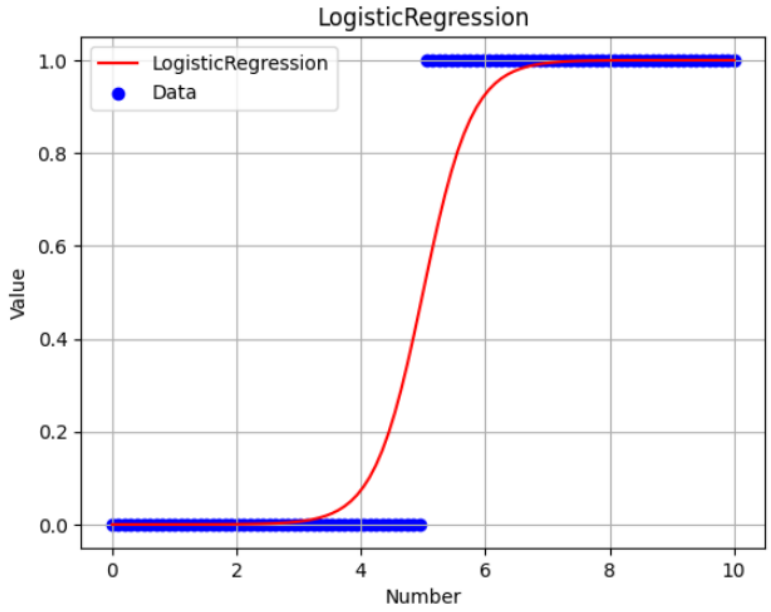


Рисунок 4.5 Логістична регресія з авторського ноутбуку

Як видно з графіку на рис. 4.5 та з авторського ноутбуку, для вхідних значень обчислюється ймовірність заданого класу model.predict_proba(x_values) – саме вона відображена на графіку червоним кольором, а потім за простими умовами її відносять до певного класу. Найбільш поширений варіант – це 1, якщо більше 0.5, та – 0, в іншому випадку (як на рис. 4.4). Важливо усвідомлювати, що LogisticRegression може застосовувати це правило і до великої кількості фічерів одночасно, і для багатокласових задач, коли класів більше 2-х. Головне, щоб цих класів не було забагато, як правило, не більше 10-20.

Може застосовуватись як для ознак з цілими (класифікаційна задача), так і з дробовими числами (регресійна задача), не змінюючи, при цьому, назву:

name_package = linear_model

name_model(model_params) = LogisticRegression (*penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None*)

Найбільш часто змінюють ті самі параметри, що й для моделей гребеневої регресії та регресії Лассо (див. вище), але замість alpha: *C* – зворотна сила регуляризації (додатне float): чим менші значення, тим – сильніша регуляризація, тобто обмеження a-коефіцієнтів;

Математично ця модель дозволяє обчислити ймовірність кожного класу для вхідних даних і потім вибрати варіант з найбільшою ймовірністю P. Наприклад, для бінарного таргета ймовірність класу 1 ($Y = 1$) для одного фічера x обчислюється за формулою:

$$P(Y=1|X=x)=\frac{1}{1+e^{-k(a_0+a_1x)}}, \quad (4.4)$$

де a_1 – коефіцієнт важливості фічера x, a_0 – зсув.

Результатом є a-коефіцієнти моделі, які відображають важливість відповідних фічерів та значення зсуву для аналізу методами FE.

На рис. 4.6 наведено приклад передбачення 5 різних датасетів логістичною регресією з різними параметрами.

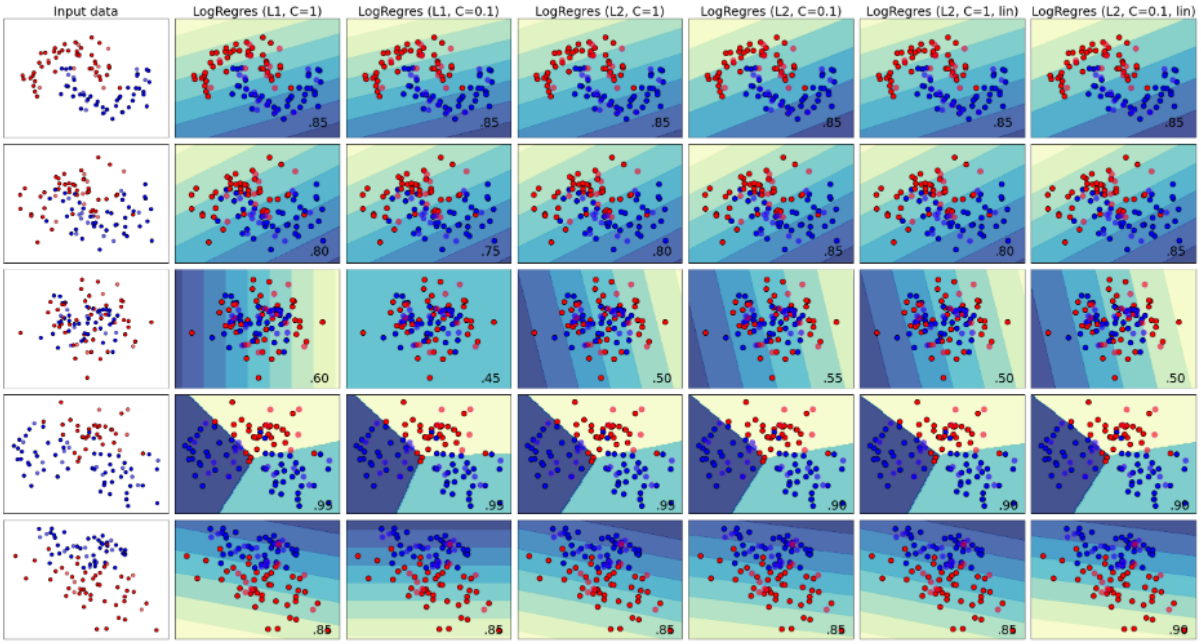


Рисунок 4.6 – Класифікація даних 5 різних датасетів логістичною регресією з різними параметрами (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [24]

Як видно на рис. 4.6, регуляризація і вибір метрики (L1 чи L2) дає різні результати. Перший датасет – інваріантний до змін параметрів, другий – вибір регуляризації L2 з більш сильною регуляризацією (менші значення C) підвищує точність. На третьому L1 дає кращу точність, але все одно – низьку (0,6). На четвертому кращими і L1, і L2. На 5-му гарну точність 0,9 забезпечує L2 з більш сильною регуляризацією (C=0,1) та методом оптимізації solver='liblinear' (див. параметр «lin» на графіку). Висновок: універсальних порад немає – в кожному випадку треба проводити індивідуальний старанний тюнінг (налаштування) моделі. Про автоматизацію тюнінгу буде нижче – у підроз. 4.3.

На рис. 4.6 однією з найкращих моделей за точністю на тестових даних є LogisticRegression(L1, C=1). Вона є найкращою на першому (0,85), третьому (0,6) і четвертому (0,95) датасетах. На другому (0.8) і п’ятому (0.85) вона посідає друге місце.

4.1.6 Стохастичний градієнтний спуск

Стохастичний градієнтний спуск (англ. «Stochastic Gradient Descent») – ітеративний метод оптимізації градієнтного спуску за допомогою стохастичного наближення. Використовується для прискорення пошуку таргета шляхом використання обмеженої вибірки (партії, батча) тренувальних даних, які вибираються випадково на кожній ітерації. Його можна розглядати як стохастичне наближення до методу оптимізації градієнтного спуску, фактичний градієнт для усього тренувального датасету замінюється його оцінкою (рис. 4.7). Метод суттєво зменшує обсяг обчислень, дозволяє працювати з великими даними, коли дані завантажуються партіями.

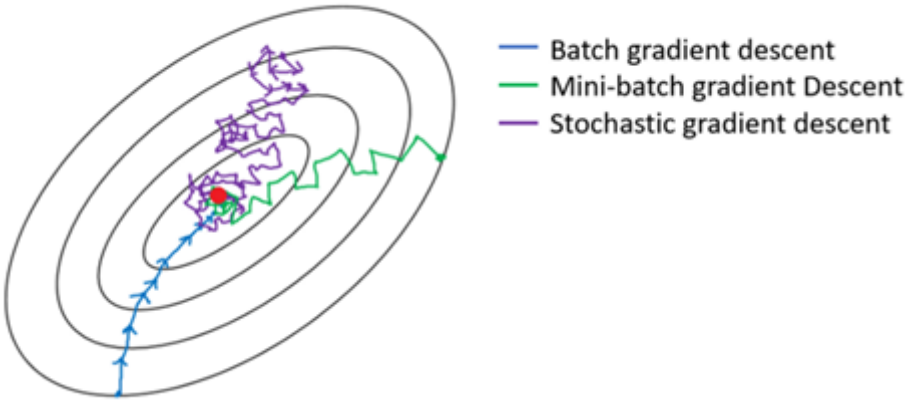


Рисунок 4.7 Ілюстрація роботи методу стохастичного градієнтного спуску зі статті

name_package = linear_model
name_model(model_params):

- SGDClassifier(*loss='hinge', *, penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1, n_jobs=None, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False, average=False*);
- SGDRegressor(*loss='squared_error', *, penalty='l2', alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0, epsilon=0.1, random_state=None, learning_rate='invscaling', eta0=0.01, power_t=0.25, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, warm_start=False, average=False*).

Головним популярним параметром, який варіюють для підвищення точності, окрім згаданих вище *tol*, *alpha*, *max_iter*, є *learning_rate* – швидкість навчання на кожній епосі (ітерації) – позитивне float.

Приклад передбачення 5 різних датасетів буде в наступному пункті, разом з іншими методами.

4.1.7 Методи опорних векторів

Метод опорних векторів (опорно-векторних машин) - це метод аналізу даних за допомогою моделі, яка відносить нові дані до однієї чи іншої категорії (класу, кластеру) шляхом проведення найширшого "коридору" між опорними векторами, які є "стінками" сусідніх класів у багатовимірному просторі і потім відносить дані до кожного з цих класів (рис. 4.8).

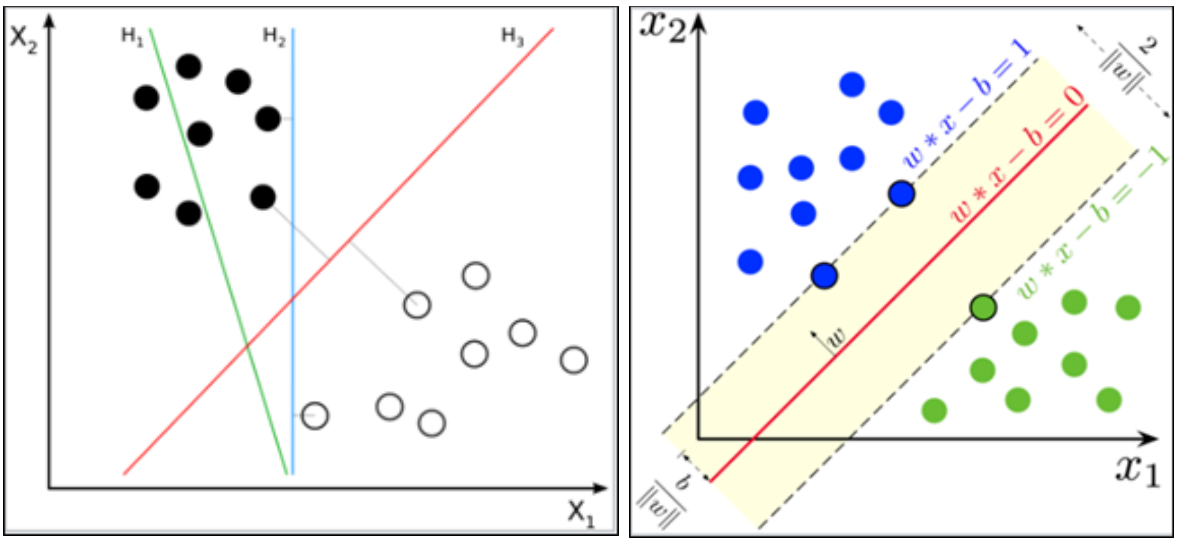


Рисунок 4.8 – Ілюстрація проведення «коридору» між опорними векторами («стінками» цього «коридору») у методі опорних векторів для двовимірної площини за випадку двох вхідних фічерів

Для ускладнення алгоритму, здійснюється попередня нелінійна трансформація даних з використанням одного з видів так званих ядер (кернелів) (рис. 4.9). Окрім наведених на рис. 4.9, ще бувають: «sigmoid», «precomputed».

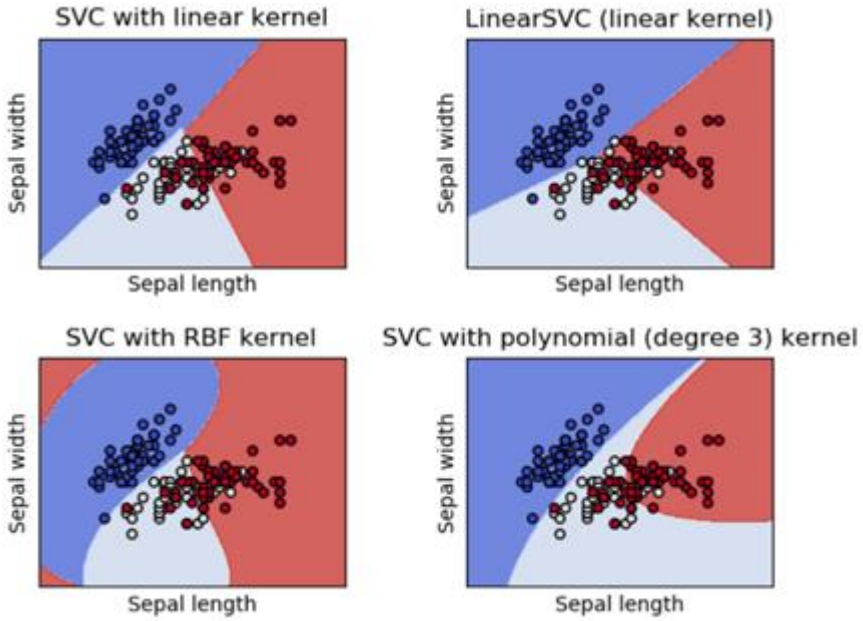


Рисунок 4.9 – Види ядер (кернелів) методу опорних векторів з ілюстраціями їх застосування з документації: а) лінійне ядро «linear» методу SVC; б) лінійне ядро методу LinearSVC; в) «rbf» (радіально-базісні функції); г) поліноміальне ядро 3-го порядку «poly»

Вид ядра є гіперпараметром.

Метод LinearSVC є подібним до SVC(*kernel*=«linear»), але використовує дещо іншу модель і краще працює для більших датасетів.

name_package = svm
name_model(model_params):
- класифікатори:
  - SVC(*, *C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None*);
  - LinearSVC(*penalty='l2', loss='squared_hinge', *, dual='warn', tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000*).

- регресори:
  - SVR(*, *kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False, max_iter=-1*);
  - LinearSVR(*, *epsilon=0.0, tol=0.0001, C=1.0, loss='epsilon_insensitive', fit_intercept=True, intercept_scaling=1.0, dual='warn', verbose=0, random_state=None, max_iter=1000*).

Головним параметром, який варіюють для підвищення точності, окрім згаданих вище *tol*, *C*, *max_iter*, у моделях SVC та SVR є *kernel*.

На рис. 4.10 наведено приклад передбачення 5 різних датасетів методом стохастичного градієнту та методом опорних векторів із різними параметрами.

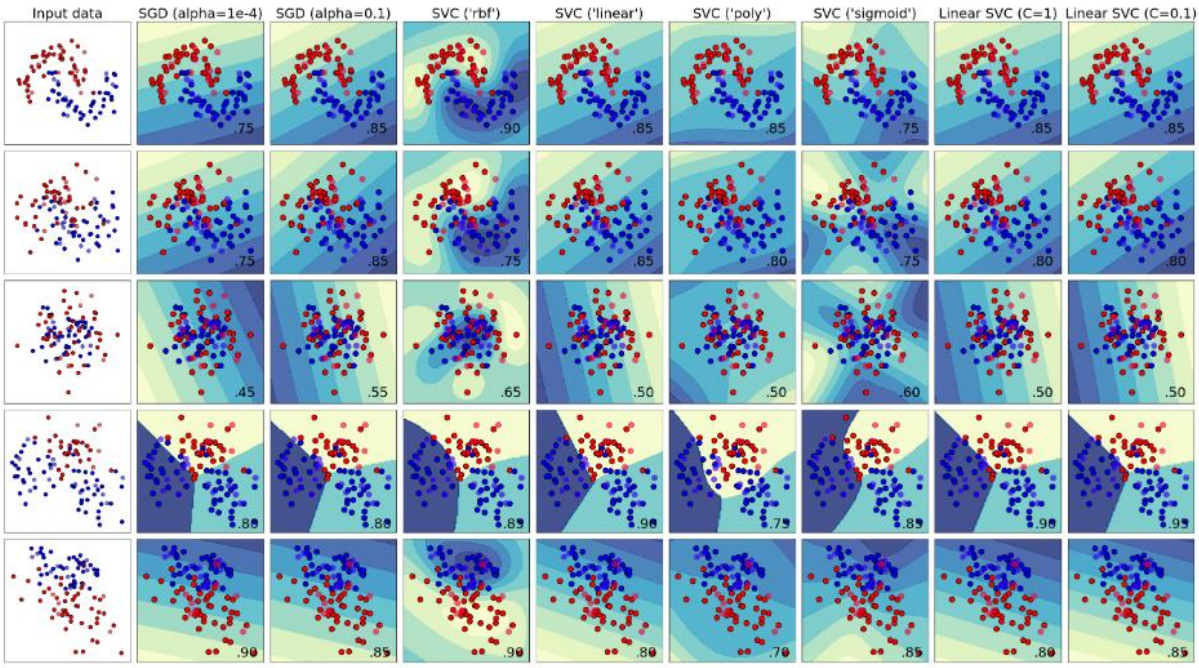


Рисунок 4.10 – Класифікація даних 5 різних датасетів методом стохастичного градієнту (SGD) та методом опорних векторів (SVC) із різними параметрами (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [24]

Як видно на рис. 4.10, збільшення регуляризації (більші alpha) для моделі стохастичного градієнту (SGD) підвищує точність на перших 3-х датасетах. Для моделі методу опорних векторів SVC ядро «RBF» забезпечує найбільшу точність для усіх датасетів, окрім другого, де найкращим є ядро «linear». Збільшення регуляризації (менші C) для моделі LinearSVC підвищує точність на останніх 2-х датасетах. А загалом, найкращими за точністю на тестових даних є такі:

- на першому датасеті (0,9): SVC з ядром «RBF»;
- на другому датасеті (0,85): SGD (alpha=0.1), SVC з ядром «linear»;
- на третьому датасеті (0,65): SVC з ядром «RBF»;
- на четвертому датасеті (0,95): LinearSVC (C=0.1);
- на п’ятому датасеті (0,9): SGD (alpha=10^{-4}), SVC з ядром «RBF».

Отже, найкращими у цьому прикладі можна вважати SVC з ядром «RBF» та LinearSVC (C=0.1).

4.1.8 Метод k-найближчих сусідів

Метод k-найближчих сусідів (англ. «k-nearest neighbor method») (KNN) - простий непараметричний класифікаційний метод, де для класифікації об'єктів у багатовимірному просторі фічерів вибираються об'єкти, до яких відстань - найменша, і вони виділяються в окремий клас. Аналізують об'єкти різних класів на заданій відстані k і об'єкт відноситься до того із них, який є найчисленнішим серед них.

На рис. 4.11 наведено приклад, який пояснює сутність методу. Тестовий зразок (зелене коло) слід віднести або до синіх квадратів, або до червоних трикутників. Якщо k = 3 (коло суцільної лінії), тоді він відноситься до червоних трикутників, оскільки всередині внутрішнього кола є 2 трикутники і лише 1 квадрат. Якщо k = 5 (коло штрихової лінії), тоді він відноситься до синіх квадратиків (3 квадрати проти 2 трикутників всередині зовнішнього кола).

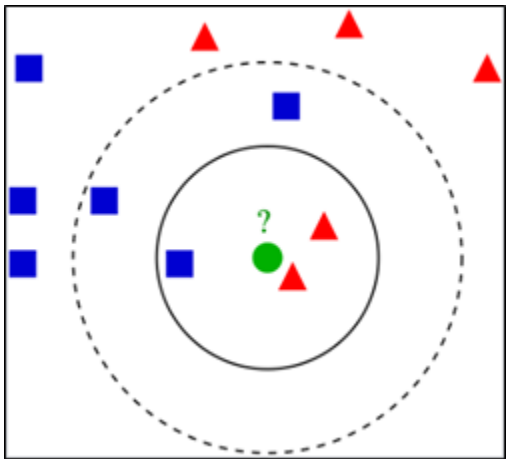


Рисунок 4.11 – Ілюстрація до пояснення принципу роботи методу k-найближчих сусідів

На рис. 4.12 наведено приклади роботи методу.

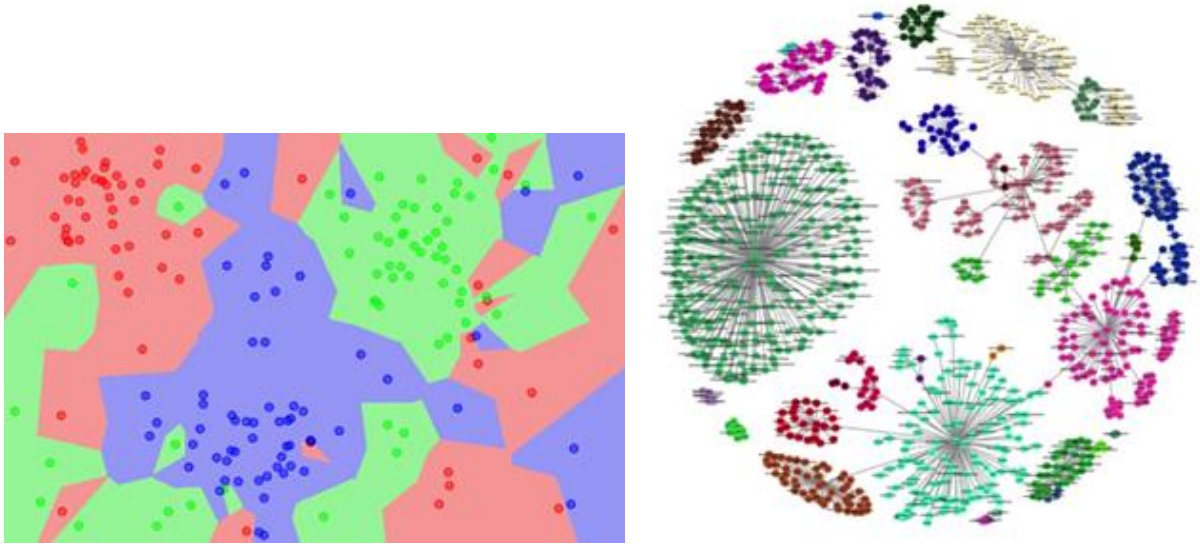


Рисунок 4.12 – Приклади роботи методу k-найближчих сусідів:
а) з джерела; б) з джерела

name_package = neighbors
name_model: KNeighborsClassifier, KNeighborsRegressor
model_params (*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*)

Для класифікатора та регресора параметри однакові, окрім першого, який задає «сусідів і саме його слід налаштовувати для моделі, у першу чергу:

- для класифікатора: *n_neighbors* – позитивне натуральне int;
- для регресора: *radius* – радіус, в якому слід шукати «сусідів» – позитивне float.

Параметр *weights* дозволяє задавати різні ваги для точок, залежно від відстані до заданої.

Має переваги і недоліки, як методах кластеризації, оскільки їх принципи роботи – схожі. Для роботи потрібні усі дані одразу, тому він не є ефективним для великих даних. Може працювати без вчителя. По суті, він не має моделі – метод просто кластеризує дані. Але натренований на тренувальних даний алгоритм вміє застосовувати і до тестових даних.

Є ефективним для даних у сфері економіки, коли діє багато різних чинників, які погано відстежуються, і може не бути єдиної адекватної моделі. Треба просто, по суті, кластеризувати дані, а потім за таким же алгоритмом здійснювати передбачення.

Цікаво, що ця модель не має параметра *random_state.*

Приклад передбачення 5 різних датасетів буде в одному з наступних пунктів, разом з іншими методами.

4.1.9 Методи прогнозування на основі процесів Гаусса

Методи прогнозування на основі процесів Гаусса здійснюють передбачення не самих значень даних, а – їх імовірнісних характеристик (середнього значення та середньоквадратичного відхилення). Передбачення є ймовірнісним, а тому можна легко обчислити емпіричні довірчі інтервали (рис. 4.13).

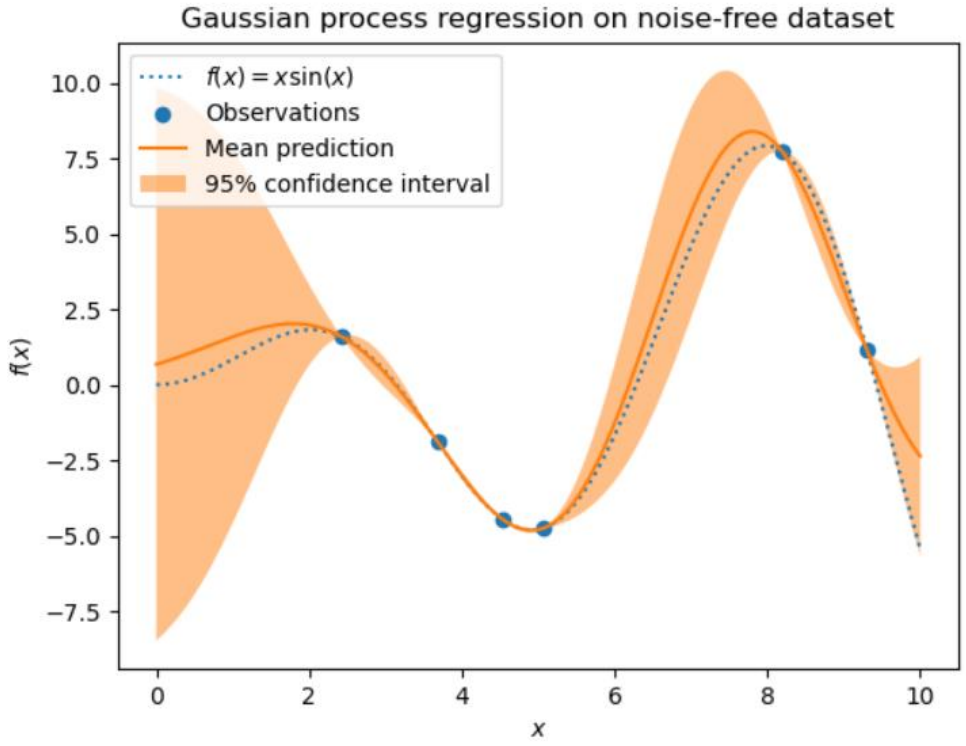


Рисунок 4.13 Ілюстрація з документації прикладу роботи методу прогнозування на основі процесів Гаусса

Можна вказувати ядра додаткової трансформації даних. Метод працює одразу з усіма даними, тому – непридатний для великих даних.

name_package = gaussian_process
name_model(model_params):
- GaussianProcessClassifier(*kernel=None, *, optimizer='fmin_l_bfgs_b', n_restarts_optimizer=0, max_iter_predict=100, warm_start=False, copy_X_train=True, random_state=None, multi_class='one_vs_rest', n_jobs=None*);
- GaussianProcessRegressor(*kernel=None, *, alpha=1e-10, optimizer='fmin_l_bfgs_b', n_restarts_optimizer=0, normalize_y=False, copy_X_train=True, n_targets=None, random_state=None*).

Головним популярним параметром, який варіюють для підвищення точності, є *kernel* – ядро (RBF, ConstantKernel, DotProduct, ExpSineSquared, Matern, можливі комбінації ConstantKernel і DotProduct), яке визначає коваріаційну функцію гауссівських процесів.

Приклад передбачення 5 різних датасетів буде в наступному пункті, разом з іншими методами.

4.1.10 Модель «наївного» Баєса

«Наївний» баєсівський класифікатор — ймовірнісний класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності даних до одного із класів, використовуючи "наївну" гіпотезу про незалежність фічерів (рис. 4.14).

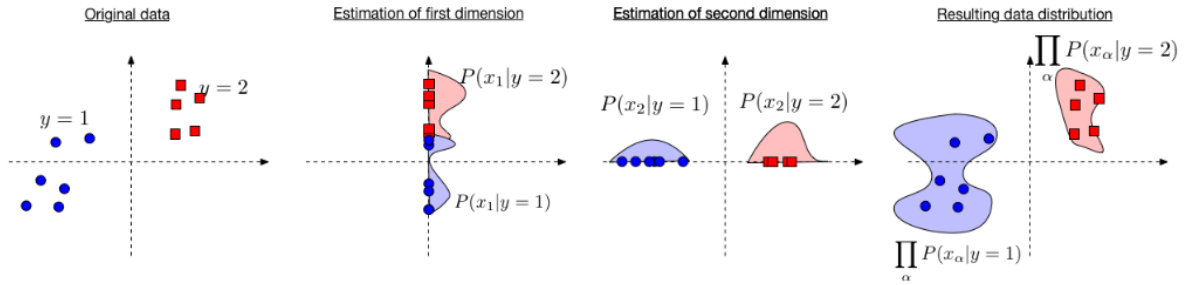


Рисунок 4.14 Ілюстрація прикладу роботи моделі «наївного» Баєса

Метод має високу швидкодію, мінімальне використання пам’яті і може давати результати тоді, коли інші методи просто не можуть запуститись. Однак, він є ефективним тільки, за умови, що гіпотеза про незалежність ознак дійсно схожа на істинну. На перших стадія розвитку алгоритмів фільтрування спаму саме цей метод давав гарні результати, а потім спамери навчились додавати корисний текст у назву та зміст і цей метод перестав бути ефективним.

name_package = naive_bayes
name_model(model_params): GaussianNB(*, *priors=None, var_smoothing=1e-09*).

Метод може запускатись і без параметрів: GaussianNB(). Цікаво, що ця модель не має параметра *random_state.*

На рис. 4.15 наведено приклад передбачення 5 різних датасетів методом k-найближчих сусідів, класифікатором на основі гауссівських процесів та наївним баєсівським класифікатором із різними параметрами.

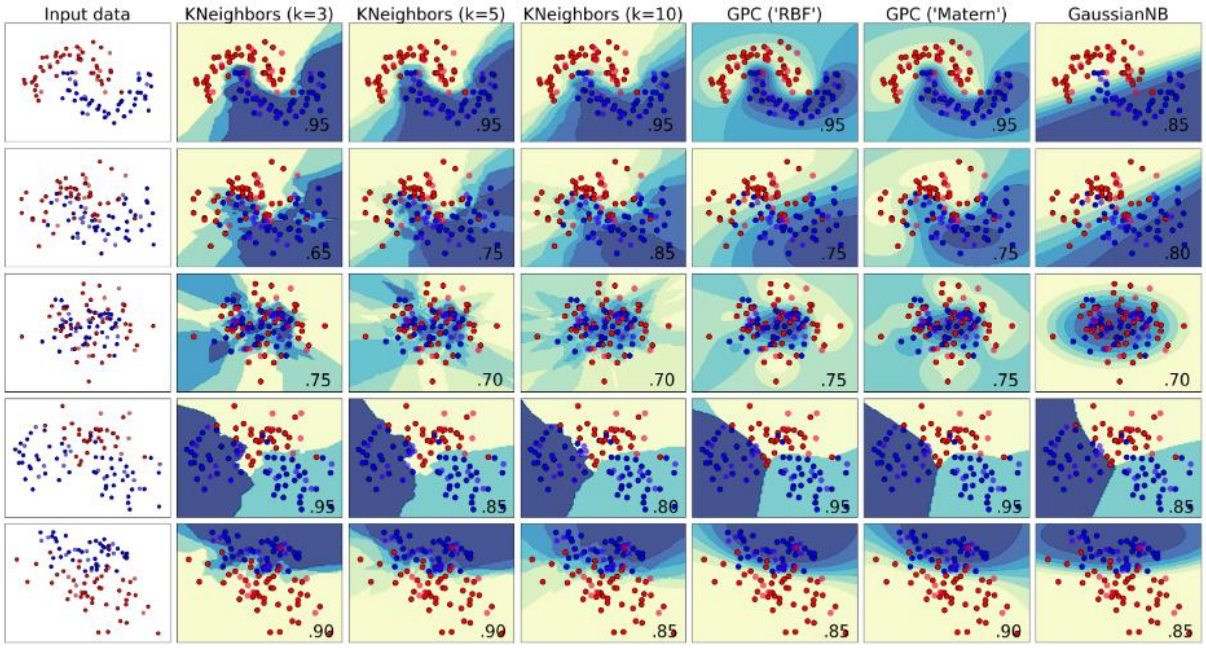


Рисунок 4.15 – Класифікація даних 5 різних датасетів методом k-найближчих сусідів (KNeighbors), класифікатором на основі гауссівських процесів (GPC) та наївним баєсівським класифікатором (GaussianNB) (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [24]

Як видно на рис. 4.15, чим менше k у методі k-найближчих сусідів (KNeighbors), тим – вища точність, що – доволі очікувано. Фактично, більше k є аналогією вищому рівню регуляризації. І для другого датасету це спрацьовує, оскільки найкращою є модель KNeighbors з k=10.

У класифікаторі на основі гауссівських процесів (GPC) кращі результати забезпечує ядро «Matern» (з англ. – «материнське»).

Модель на основі наївного баєсівського класифікатору не є найкращою ні на жодному датасеті.

Отже, найкращими у цьому прикладі можна вважати KNeighbors з k=3 і 10 та GPC з ядром «Matern».

Варто додати, що, іноді є ефективним варіант моделі наївного Баєса під назвою «Multinomial Naive Bayes». Основна відмінність між ними поля-

гає в тому, що модель «Multinomial Naive Bayes» використовується для класифікації документів, у яких ознаки є дискретними, наприклад, слова в документі, у той час як модель Naive Bayes може використовуватися для класифікації документів, у яких ознаки можуть бути як дискретними, так і неперервними. Прикладом використання «Multinomial Naive Bayes» є Kaggle-конкурс «LLM - Detect AI Generated Text», де в багатьох ноутбуках побудовано ансамбль на основі саме цієї моделі.

4.1.11 Дерева рішень

Дерево рішень у машинному навчанні – це структурована модель прийняття рішень з умов розгалуження if...then...else, яка виглядає як «дерево» і складається з вузлів, гілок та листя. Кожен вузол представляє рішення для певного піднабору даних, гілки вказують напрямок, який слід взяти, щоб дійти до нового вузла чи кінцевого результату у вигляді листя. Зазвичай, задається скільки даних максимально може бути у листку – це й задає умову, за якою вузли відрізняються від листя.

Дерева рішень є зручними для розуміння закономірностей, для візуалізації рішення, яке є простим у розумінні. Щоб уникнути перенавчання, для дерев задають умови «*обрізання*» (англ. «pruning»). Теоретично, дерево глибиною n буде мати 2^n листків. Наприклад, якщо глибина дорівнює 6, тоді це буде 64 листя. І тоді задають один з двох варіантів «обрізання»: або за глибини 6 вимагають, щоб було не більше, наприклад 40 < 64 листя, або за кількості листя 64 вимагають, щоб максимальна глибина не перевищувала, наприклад 8 > 6. І тоді алгоритм вимушений будувати «обрізане» дерево, яке буде значно більш узагальненим, аніж «необрізане». Параметри моделі:

name_package = tree
name_model(model_params):
- DecisionTreeClassifier *(*, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*),
- DecisionTreeRegressor*(*, criterion='squared_error', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, ccp_alpha=0.0*)

Параметри моделі-класифікатора та регресора відрізняються тільки метрикою (критерієм):

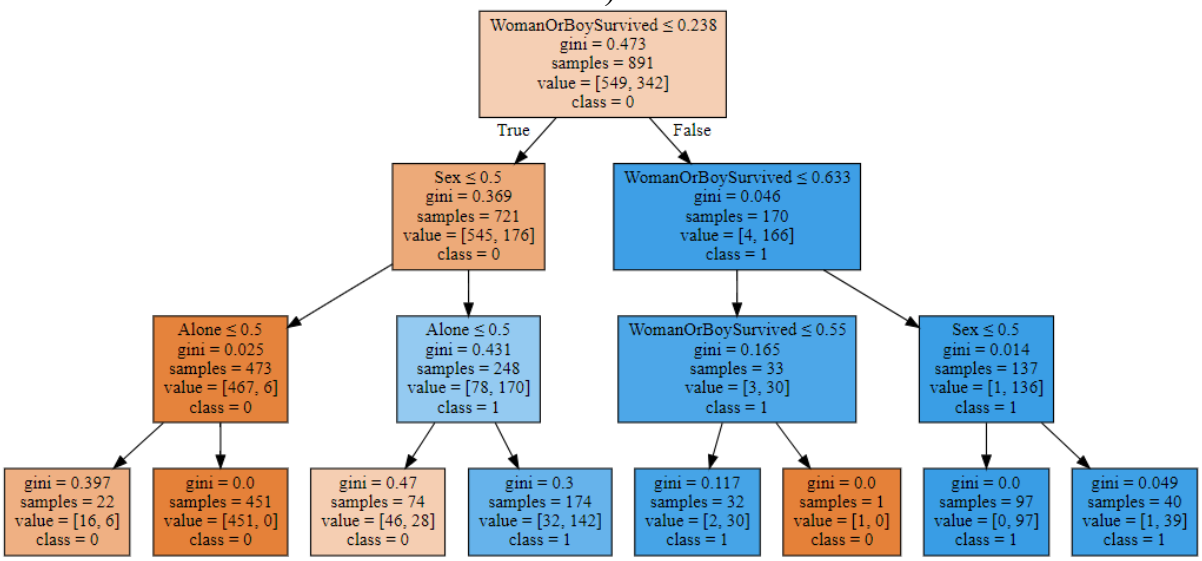
- класифікатор: {“gini”, “entropy”, “log_loss”}
- регресор: {“squared_error”, “friedman_mse”, “absolute_error”, “poisson”}

Розробники моделі наполегливо рекомендують вказувати параметр max_depth як якесь ціле число, починаючи від 2 (замість 1 краще вже писати одразу одне розгалуження if …then… else).

Головною перевагою дерева рішень є гарна інтерпретабельність. Є метод відображення дерева рішень саме як дерева з усіма умовами і потім його можна замінити на послідовність розгалужень if …then… else. Такий прийом використаний одним із авторів посібника під час розв'язання конкурсу Kaggle з передбачення тих, хто вижив на «Титаніку» («Titanic - Machine Learning from Disaster»), у своєму публічному ноутбуку «Titanic - Top score : one line of the prediction» (рис. 4.16).

```
import graphviz
from sklearn.tree import export_graphviz
dot_data = export_graphviz(model, out_file=None, feature_names=train_x.columns,
                           class_names=['0', '1'], filled=True,
                           rounded=False,special_characters=True, precision=3)
graph = graphviz.Source(dot_data)
graph
```

а)



б)

Рисунок 4.16 – Візуалізація дерева рішень для конкурсу «Titanic - Machine Learning from Disaster» з параметрами max_depth=3, min_samples_leaf=2: а) код для візуалізації дерева рішень model з використанням бібліотеки graphviz; б) результат запуска коду

На рис. 4.16б добре видно усі гілки, вузли і листя. Вузи – це прямокутники, в які входить стрілка і з яких 2 стрілки виходять. У листя стрілка тільки входить. Колір прямокутника відповідає класу «class» (0 чи 1 – людина не вижила чи вижила, відповідно). Насиченість кольору зворотно пропорційна значенню критерія «gini» (чим він ближчий до 0, тим – він більш «чистий», тобто містить дані тільки одного класу).

Рис. 4.16 – це класична візуалізація дерева рішень. Як бачимо, воно «росте» в глибину, а не – у висоту, і тому у параметрах обмежується його глибина max_depth, а не – висота. Хоча, за аналогією, з біологією, перша вершина називається «root» (з англ. – «корінь»). Біологам може бути це не звичним, але так домовились у машинному навчанні. Ймовірно, причина знаходиться в особливостях візуалізації. Як правило, на комп'ютерах перша точка на екрані знаходиться зверху зліва. У разі відображення ходу «зростання» дерева в циклі доцільно почати зверху вниз. Тому й усі звикли, щоб воно «росте» вниз, хоча, у більшості випадків, дерево спочатку розраховується програмно-математично, а потім – візуалізується і цієї проблеми б вже не було, але усі звикли до такого підходу.

Зручність інтерпретабельності дерева рішень полягає в тому, що його можна просто «прочитати»: якщо … тоді…, а тоді, якщо … тоді… і так – до кінця. А тоді все дерево можна замінити однією умовою, яка й буде рішенням задачі. Так і було зроблено у ноутбуку «Titanic - Top score : one line of the prediction» (2019 р.), саме тому він і має назву «один рядок коду передбачення» (рис. 4.17. Звичайно, цьому рішенню передує етап FE, де синтезуються нові ознаки, які дають таке гарне і просте рішення. Це рішення дає точність 0.80383, що станом на листопад 2023 р. дає рівень «Top4%» (600-те місце з 15,3 тис. команд, хоча, насправді, й – вище, оскільки чимала частина команд має точність 1,0, завантаживши просто відповіді, всупереч правилам конкурсу).

```
# The one line of the code for prediction : LB = 0.80382 (Titanic Top 6%)
test_x['Survived'] = (((test_x.WomanOrBoySurvived <= 0.238) & (test_x.Sex > 0.5) & (test_x.Alone
> 0.5)) | \
                      ((test_x.WomanOrBoySurvived > 0.238) & \
                       ~((test_x.WomanOrBoySurvived > 0.55) & (test_x.WomanOrBoySurvived <= 0.63
3))))
```

Рисунок 4.17 – Один рядок коду передбачення тестових даних у задачі («Titanic - Machine Learning from Disaster», який відповідає дереву рішень на рис. 4.16б і дає точність рівень «Top4%» конкурсу

Побудова дерев рішень основана на теорії інформації Шеннона та теорії ймовірності. Зупинимось на цьому детальніше, оскільки це – гарний привід зрозуміти прикладну цінність цих двох теорій. Використаємо методологію статті і код, наведений на рис. 4.18: беремо 20 прямокутників, у т.ч. 8 – жовтого і 12 – блакитного кольорів, та сформуємо з них 2 нових вузли (а може й – листя), використовуючи поняття ентропії теорії інформації Шеннона та теорії ймовірностей.

```
total_squares = 20
yellow_squares = 8
blue_squares = total_squares - yellow_squares

colors = ['yellow'] * yellow_squares + ['aqua'] * blue_squares
np.random.shuffle(colors)

fig, ax = plt.subplots()

for i, color in enumerate(colors, start=1):
    square = plt.Rectangle((i - 0.5, 0), 1, 1, fc=color)
    ax.add_patch(square)
    plt.text(i, 0.5, str(i), color='black',
             ha='center', va='center', fontsize=10)

ax.set_xlim(0, total_squares + 1)
ax.set_ylim(0, 1)
ax.set_aspect('equal', adjustable='box')
plt.axis('off')
plt.show()
```

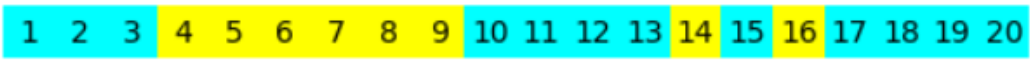


Рисунок 4.18 – Код генерування 20 прямокутників, у т.ч. 8 – жовтого і 12 – блакитного кольорів, для пояснення як формується вузол дерева рішень

Як відомо, ентропія – це міра хаосу, яка відображає наскільки є неупорядкованими елементи у нашій вибірці. Наша задача – збільшити кількість інформації шляхом упорядкування цих елементів, по суті - кластеризації. З теорії інформації Шеннона відомо, що кількість інформації S обчислюється по формулі:

$$S = -\sum_{i=1}^{N} p_i log_2 p_i, \quad (4.5)$$

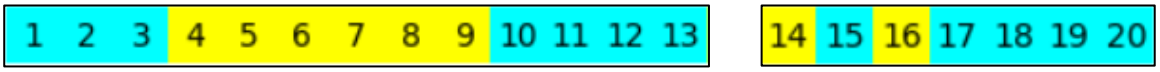
де N=2 – кількість класів (на рис. 4.18 – кольорів), p_i – ймовірність того, що, якщо ми візьмемо один елемент із вибірки, то він буде певного класу.

Для вибірки умовно 0-ої («root» - з англ. «корінь») вершини на рис. 4.18, де є 2 кольори (N = 2), у т.ч. 8 жовтих та 12 блакитних із 20 елементів загалом, формула (4.5) матиме вигляд:

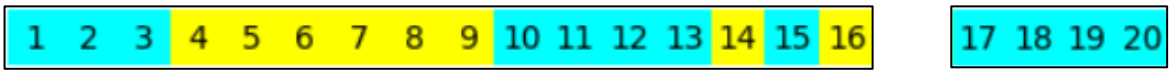
$$S_0 = -\frac{8}{20} log_2 \frac{8}{20} - \frac{12}{20} log_2 \frac{12}{20} = 0{,}53 + 0{,}44 = 0{,}97. \quad (4.6)$$

Важливо відмітити, що формула (4.5) характеризує саме ступінь хаосу. Отже, якщо у вибірці усі елементи будуть одного кольору, наприклад 20 з 20 будуть жовтими, тоді ентропія буде відсутня і тоді формулу (4.6) не застосовують ($log_2 0$ взяти й не вдасться), а одразу пишуть, що вона дорівнює 0.

Теоретично, є багато варіантів розбиття вибірки з рис. 4.18 на 2 підмножини. Розглянемо, як, при цьому, буде змінюватись кількість інформації у кожній з цих підмножин. Здійснимо розбиття після 13-го і після 16-го елементу (рис. 4.19):



а)



б)

Рисунок 4.19 – Варіанти розбиття 20 прямокутників з рис. 4.18 на 2 підмножини: а) по 12 і 8 прямокутників, б) по 16 і 4

Тепер застосуємо формулу (4.5) до лівої «left» і правої «right» підмножин обох варіантів розбиття рис. 4.19а і 4.19б (використаємо індекси 1 і 2, відповідно):

$$S_1^{left} = -\frac{6}{13}log_2\frac{6}{13} - \frac{7}{13}log_2\frac{7}{13} = 0{,}51 + 0{,}48 = 0{,}99,$$
$$S_1^{right} = -\frac{2}{7}log_2\frac{2}{7} - \frac{5}{7}log_2\frac{5}{7} = 0{,}52 + 0{,}35 = 0{,}87,$$
$$S_2^{left} = -\frac{8}{16}log_2\frac{8}{16} - \frac{8}{16}log_2\frac{8}{16} = 0{,}5 + 0{,}5 = 1, \qquad (4.7)$$
$$S_2^{right} = 0.$$

Як бачимо, у другому варіанті права підмножина – це вже не вузол, який ще треба далі розбивати, а – листя, тобто кінцева точка розбиття.

Для порівняння який варіант кращий використовується формула для обчислення приросту інформації IG_q, тобто зменшення ентропії, у разі q-го варіанту розбиття:

$$IG_q = S_0 - \frac{n_q^{left}}{n_0}S_q^{left} - \frac{n_q^{right}}{n_0}S_q^{right}, \qquad (4.8)$$

де n_q^{left} і n_q^{right} – кількість елементів у лівій і правій, відповідно, підмножинах, у разі q-го варіанту розбиття; n_0 – кількість елементів у нульовій вершині (4.6).

Для виразів (4.7) формула (4.8) дає такі значення:

$$IG_1 = 0{,}97 - 0{,}64 - 0{,}30 = 0{,}03,$$
$$IG_2 = 0{,}97 - 0{,}8 - 0 = 0{,}17. \qquad (4.9)$$

Як бачимо, приріст інформації IG_2 є більшим за IG_1, а тому варто вибрати другий варіант розбиття (рис. 4.19б). Це – так званий «жадібний» алгоритм, коли щоразу вибирається максимально найкращий варіант, без урахування наступних рішень. Але, за реальних значень параметрів, де задається певна «обрізка» дерева, все працює набагато складніше і, у більшості випадків, слід вибирати не той варіант, коли одразу формується листя в одній з підмножин. А – той, який швидше дозволить розбити обидві множини і отримати не дуже розвинене дерево.

Приклад (4.5)-(4.9) оснований на критерії ентропії (англ. «entropy»). В загальному випадку в документації бібліотеки sklearn ці формули описуються дещо інакше:

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H\left(Q_m^{left}(\theta)\right) + \frac{n_m^{right}}{n_m} H\left(Q_m^{right}(\theta)\right), \qquad (4.10)$$
$$\theta^* = argmin_\theta\left(G(Q_m, \theta)\right),$$

де m – номер вершини, для якої аналізуються варіанти розбиття n_m елементів множини Q_m на підмножини $Q_m^{left}(\theta)$ та Q_m^{right}; $\theta = (j, t_m)$ – варіант розбиття j-го фічера з t_m порогом; $H(Q_m)$ – функція втрат (критерій):

- ентропія («entropy») (аналогічна формулі (4.5)):
$$H(Q_m) = -\sum_{k=1}^{N} p_{mk} \log_2(p_{mk}), \qquad (4.11)$$

- Джині (англ.: «gini»):
$$H(Q_m) = -\sum_{k=1}^{N} p_{mk}(1 - p_{mk}). \qquad (4.12)$$

Як видно, з порівняння (4.11) та (4.12), логарифм у (4.11) важче обчислювати, ще й треба враховувати виняток, коли ймовірність дорівнює 0. А критерій Джіні (Джині – це прізвище автора критерію) легше автоматизується і швидше обчислюється, тому саме він встановлюється за замовчуванням у моделі DecisionTreeClassifier.

З порівняння (4.10) та (4.8) видно, що у «жадібному» алгоритмі синтезу дерева рішень вибирається варіант розбиття, який або *максимізує кількість інформації* IG за виразом (4.8), або – *мінімізує приріст* (по суті – зменшення, оскільки цей приріст віднімається від константи) цієї інформації G за виразом (4.10), хоча суть – там сама.

Алгоритм працює рекурсивно і зупиняється, за виконання однієї з 2-х умов: досягнута максимальна глибина дерева max_depth, або в усіх листя кількість даних є меншою за значення параметра *min_samples_leaf*, який якраз означає мінімальну кількість даних у кожному листку дерева.

Регресійна модель використовує такий же алгоритм і параметри, але вона відрізняється критерієм $H(Q_m)$. Наприклад, за замовчуванням, у ній використовується середньоквадратичний критерій:

$$H(Q_m) = -\frac{1}{n_m}\sum_{y \in Q_m}(y - \overline{y_m})^2, \quad (4.13)$$

де $\overline{y_m}$ – середнє значення вибірки даних Q_m.

Інші критерії див. у документації.

На рис. 4.20 наведено приклад передбачення 5 різних датасетів деревами рішень з різними параметрами.

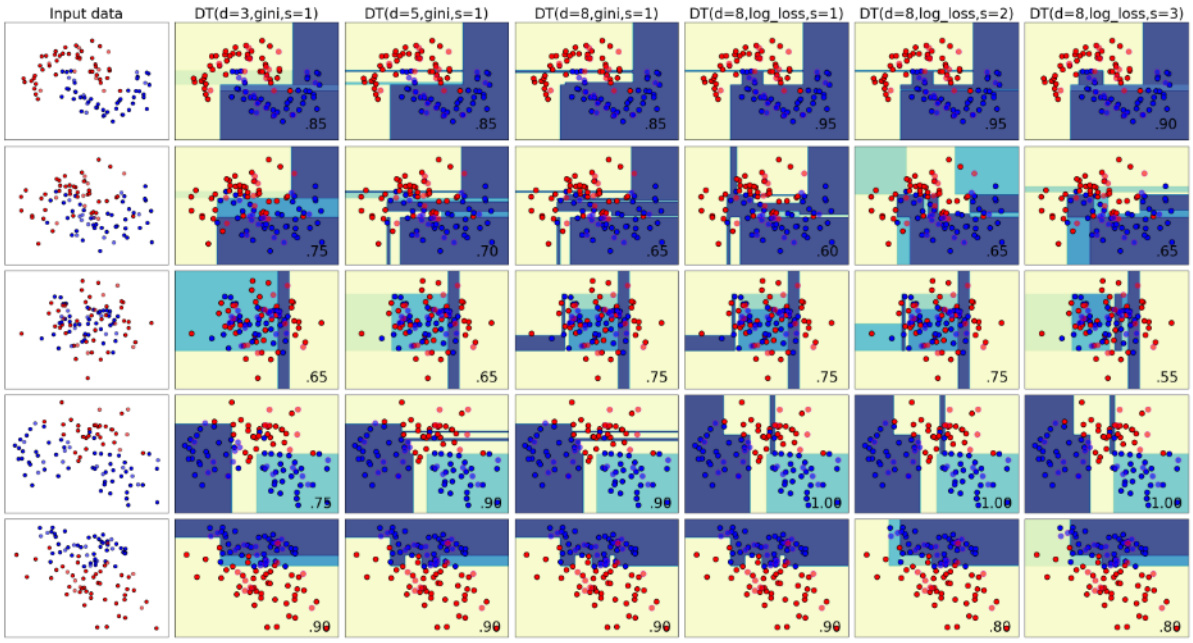


Рисунок 4.20 – Класифікація даних 5 різних датасетів деревами рішень з різними параметрами: максимальна глибина (d), метрика «gini» або «log_loss», мінімальна кількість зразків у листку (s) (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [24]

Як видно на рис. 4.20, метрика «log_loss» явно краще підходить для задачі бінарної класифікації, аніж метрика «gini». Мінімальна кількість зразків у листку є способом регуляризувати модель, але призводить до погіршення точності (однак, для більших датасетів це може мати зворотний ефект). Збільшення максимальної глибини дерева рішень дозволяє збільшити точність, окрім другого датасету, де це призводить до оверфітингу.

Отже, на рис. 4.20 найкращими моделями є «DT(d=3, gini, s=1)» (перше місце на першому, другому і п’ятому датасетах) та «DT(d=8, log_loss, s=2)» (перше місце на усіх, окрім – другого).

4.1.12 Приклад порівняння найкращих моделей

На рис. 4.21 наведено приклад передбачення 5 різних датасетів 9-ма найкращими моделями цього підрозділу, відібраних у пунктах 4.1.4-4.1.11.

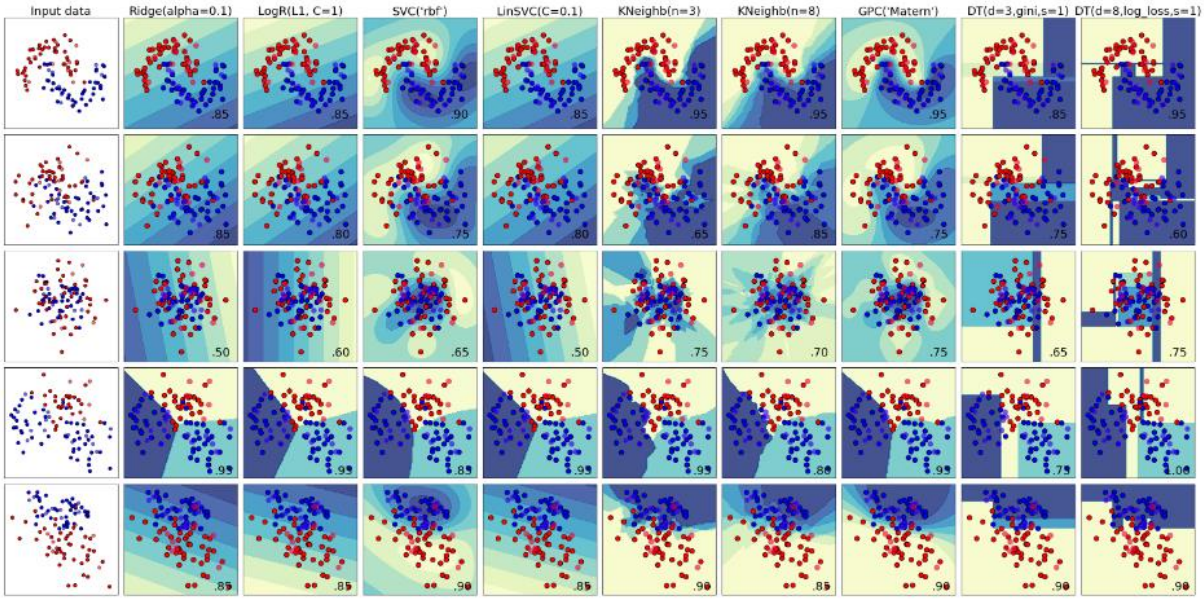


Рисунок 4.21 – Класифікація даних 5 різних датасетів 9-ма моделями (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [24]

Як видно на рис. 4.21, найкращими за точністю на тестових даних є такі:

- на першому датасеті (0,95): KNeighbors з k=3 і 10, GPC з ядром «Matern» та DT(d=8, log_loss, s=2);
- на другому датасеті (0,85): Ridge(alpha=0.1), KNeighbors з k=3;
- на третьому датасеті (0,75): KNeighbors з k=3, GPC з ядром «Matern», DT(d=8, log_loss, s=2);
- на четвертому датасеті (1,0): DT(d=8, log_loss, s=2);
- на п’ятому датасеті (0,9): SVC з ядром «RBF», KNeighbors з k=3, GPC з ядром «Matern», DT(d=3, gini, s=1), DT(d=8, log_loss, s=2).

Отже, найкращими у цьому прикладі можна вважати KNeighbors з k=3, GPC з ядром «Matern» та DT(d=8, log_loss, s=2).

Насправді, один класифікатор рідко будують. Їх побудова є тільки першим етапом аналізу. На практиці більш ефективними є ансамблі таких моделей. Розглянемо їх детальніше. І потім спробуємо збільшити точність передбачення для 5 датасетів на рис. 4.21.

4.2 Створення ансамблів моделей та узагальнення рішень

Бібліотека sklearn надає гарні можливості як для використання найбільш поширених ансамблевих моделей дерев рішень, так і для побудови власних ансамблів з будь-яких моделей. Крім того, популярними є й інші моделі, наприклад бустингові моделі від Google та Microsoft.

4.2.1 Рандомізовані ансамблі дерев: RandomForest та ін.

Одними з найбільш поширених класів моделей машинного навчання, які дають гарні результати на основі однотипної моделі, є ансамблі рандомізованих дерев рішень. У них рандомно (випадково) вибираються підмножини ознак і для кожної комбінації будуються дерева, а потім – у певний спосіб узагальнюються. Найбільш відомими серед них є такі ансамблі:

- RandomForest (скорочено: RF) – у задачі класифікації (RandomForestClassifier) найкращий прогноз визначається шляхом голосування передбачень дерев у складі ансамблю, а в задачі регресії (RandomForestRegression) – шляхом усереднення значень.
- ExtraTrees – аналізує ряд дерев рандомізованих рішень (екстра-дерев) на різних підмножинах набору даних та використовує усереднення для підвищення точності прогнозування.
- IsolationForest – добре виявляє аномальні значення під час побудови лісу дерев.

Для більшості задач RandomForest дає одні з найкращих результатів, хоча рідко – дійсно найкращі. ExtraTrees іноді дає ще кращі рішення, але, за більшості випадків, схильний до значного перенавчання. Більш детально їх порівняння наведено в табл. 4.1.

Таблиця 4.1 Відмінності моделей RandomForest та ExtraTrees

Відмінності	**RandomForest**	**ExtraTrees**
Процес обчислення порогів ознак у кожному дереві у кожному вузлі	Стандартний метод визначення оптимального порогу	Вибір порогу випадковим чином без його попереднього обчислення
Кількість ознак у вибірці	Зазвичай, випадково вибирає sqrt(кількість ознак)	Може випадково вибирати будь-яку кількість ознак
Тренувальний датасет для кожного дерева	Кожне дерево навчається на випадковій підмножині тренувальних даних з повторенням (*бутстреп-вибірка**)	Використовує весь тренувальний датасет для кожного дерева
Вартість обчислень	Зазвичай менше, у порівнянні з ExtraTrees	Зазвичай більше, у порівнянні з RandomForest
Застосування, залежно від розміру даних	Зазвичай, ефективний для середніх та великих наборів даних	Може бути ефективний для невеликих та великих наборів даних, але може бути дорожчий в обчислювальному плані

Варіантність рішень	Зазвичай менше, у порівнянні з ExtraTrees	Зазвичай більше, у порівнянні з RandomForest
Ризик перенавчання	Менше схильний до перенавчання, у порівнянні з ExtraTrees	Може бути більш схильний до перенавчання, через випадковість на рівні розгалуження
Важливість ознак	Точніше оцінює важливість ознак, оскільки вона визначається на основі усередненого внеску кожної ознаки у всі дерева	Може хибно визначати важливість ознак, через вплив випадковості розгалуження, оскільки вона обчислюється на основі внеску кожної ознаки у кожному дереві, з подальшим усередненням.

**Бутстрен* (англ. «bootstrap»)-*вибірка* (або вибірка з повторенням чи заміщенням) – це коли з датасету робиться вибірка, але, після цього, вибрані дані «повертаються назад», тобто одні й ті самі дані можуть багато разів бути присутніми у різних вибірках.

На рис. 4.22 наведено приклад передбачення 5 різних датасетів моделлю RandomForest з різними параметрами.

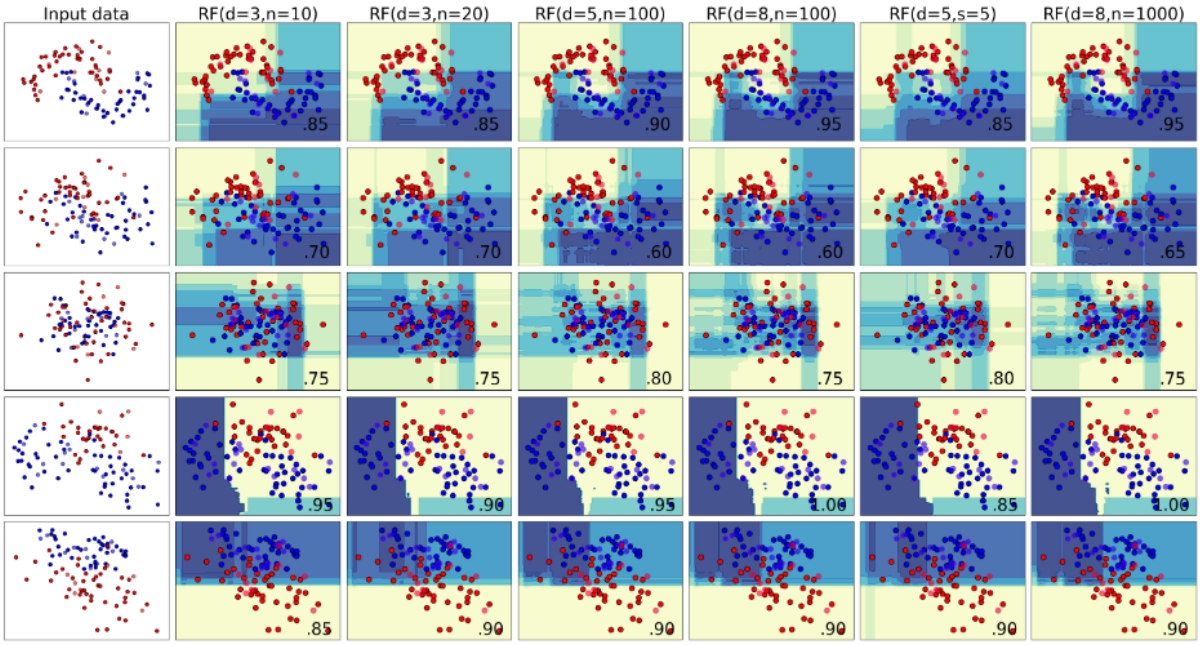


Рисунок 4.22 – Класифікація даних 5 різних датасетів моделлю RandomForest з різними параметрами: максимальна глибина (d), кількість дерев рішень в ансамблі (n), мінімальна кількість зразків у листку (s), метрика скрізь – «log_loss» (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [25]

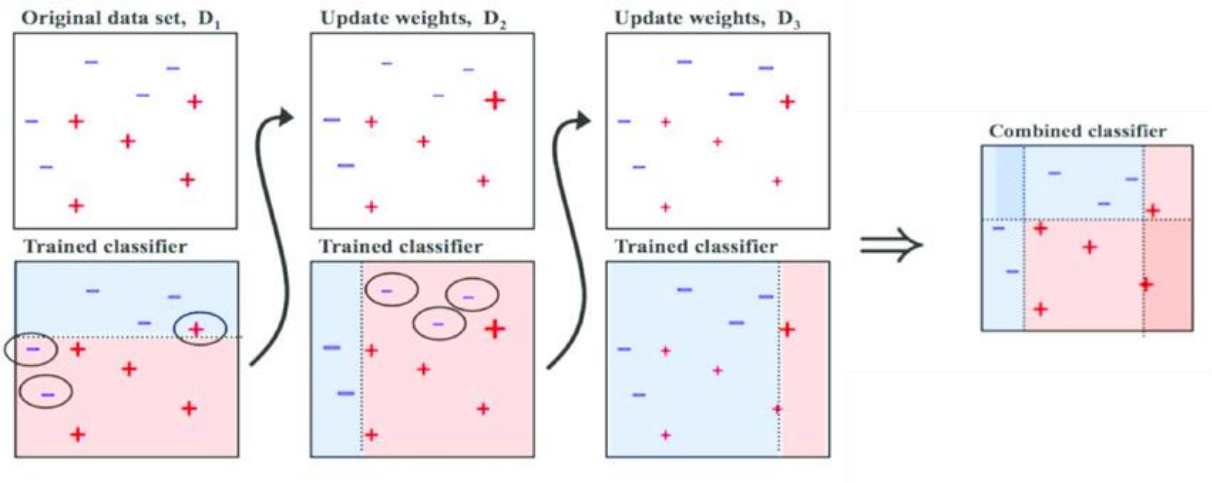
Як видно на рис. 4.22, збільшення кількості дерев рішень в ансамблі n очікувано збільшує точність. Аналогічно – більша максимальна глибина d,

за винятком другого і третього датасетів, на яких спостерігається оверфітінг. Збільшення кількість зразків у листку (s), тобто – більша регуляризація, має успіх якраз на другому датасеті. Також, на другому датасеті збільшується точність, у разі збільшення кількості дерев рішень в ансамблі у 10 разів.

Отже, на рис. 4.22 найкращими моделями є «RF(d=8, n=1000)» (1 місце на датасетах 1, 4, 5), «RF(d=5, s=5)» (1 місце на датасетах 2, 3, 5).

4.2.2 Бустінг моделей

Бустинг даних - це метод машинного навчання ансамблю слабких базових моделей, який полягає в тому, що моделі будуються послідовно таким чином, що кожна виправляє помилки попередньої. Цей принцип добре ілюструє рис. 4.23.



а) б) в) г)

Рисунок 4.23 – Ілюстрування бустингу (AdaBoost) 3-х дерев рішень: а) перша ітерація, б) друга, в) третя; г) відповідь – консолідація результатів

Як видно на рис. 4.23, стоїть задача розпізнати сині «-» і червоні «+» (класифікаційна задача). Перша модель (це може бути й дерево рішень з max_depth=1, тобто звичайна умова «if...then…else») здійснює поділ даних на 2 класи. Далі аналізуються помилки (їх в цьому контексті ще називають «*викиди*» - англ. «outliers») – на нижньому рисунку на рис. 4.23а їх обведено еліпсами. Наступна модель намагається, передусім, класифікувати саме ці дані, обведені еліпсом. При цьому утворюються і аналізуються нові помилки (див. обведені еліпсом на нижньому рисунку на рис. 423б), аналогічно виконується третя ітерація (рис. 4.23в). А тоді, якщо приймається рішення, що ітерацій достатньо, здійснюється консолідація моделей з усіх 3-х ітерацій і отримується фінальне рішення. На рис. 4.23г точність – accuracy_score на тренувальних даних досягає 100%. Зазвичай, ознак і точок більше, вони більше перемішані, а тому використовуються потужні методи і багато різних технологій та тисячі ітерацій.

Серед найбільш відомих бустингових моделей є такі:

- AdaBoost (з англ. - "адаптивний бустинг") (див. рис. 4.23);
- Gradient Boosting Machines (GBM) або просто "Gradient Boosting";
- XGBoost (скорочення від "eXtreme Gradient Boosting") від Google;
- LightGBM від : Microsoft є ще однією швидкою і ефективною бібліотекою для градієнтного бустінгу. Вона використовує особливий алгоритм для оптимізації обчислень та підтримує категоріальні ознаки.

Найбільш потужними є XGBoost і LightGBM є основними конкурентами для передбачення даних і, як правило, демонструють кращі результати, ніж моделі бібліотеки Sklearn, але вони містять дуже багато параметрів і треба вміти їх налаштовувати. Крім того, ці дві моделі мають можливості для роботи категоріальними та текстовими ознаками напряму без передоброблення, що поліпшує точність, у порівнянні з моделями Sklearn, в яких ще потрібне передоброблення, яке часто, призводить до втрати цінної інформації.

Моделі бібліотеки Sklearn, як правило, швидше і простіше налаштовуються, але добре налаштовані XGBoost і LightGBM є точнішими.

Моделі XGBoost та LightGBM вимагають попередню трансформацію даних у спеціальний формат з використанням методів xgb.DMatrix та lgbm.Dataset, відповідно. Приклад такої трансформації наведено в авторському ноутбуку.

Моделі XGBoost вважаються більш точними, ніж моделі на основі LightGBM, але, на практиці, набагато важче підібрати ту саму комбінацію параметрів, яка доведе це твердження. Моделі на основі LightGBM налаштовуються простіше і швидше, ніж моделі на основі XGBoost, а тому їх застосовують частіше.

Є ще більш швидкі і прості варіанти XGBClassifier, XGBRegressor, LGBMClassifier, LGBMRegressor, які не потребують попередньої трансформації даних і застосовуються як звичайні Sklearn-моделі, однак, вони не завжди спрацьовують чи дають прийнятний результат.

Для уникнення перенавчання моделей XGBoost розробники наполегливо рекомендують задавати параметр max_depth (максимальна глибина дерева рішень), а для LightGBM – num_leaves (максимальна кількість листя (вузлів)). Звичайно, є значення за замовчуванням, але запуск без його задавання як гіперпараметра в явному вигляді часто призводить до появи застережливого повідомлення про те, що варто його задати. Важливо, також, правильно вказувати метрику цих моделей. На рис. 4.24 наведено приклад документації XGBoost з цього питання, але цей список там – на декілька сторінок.

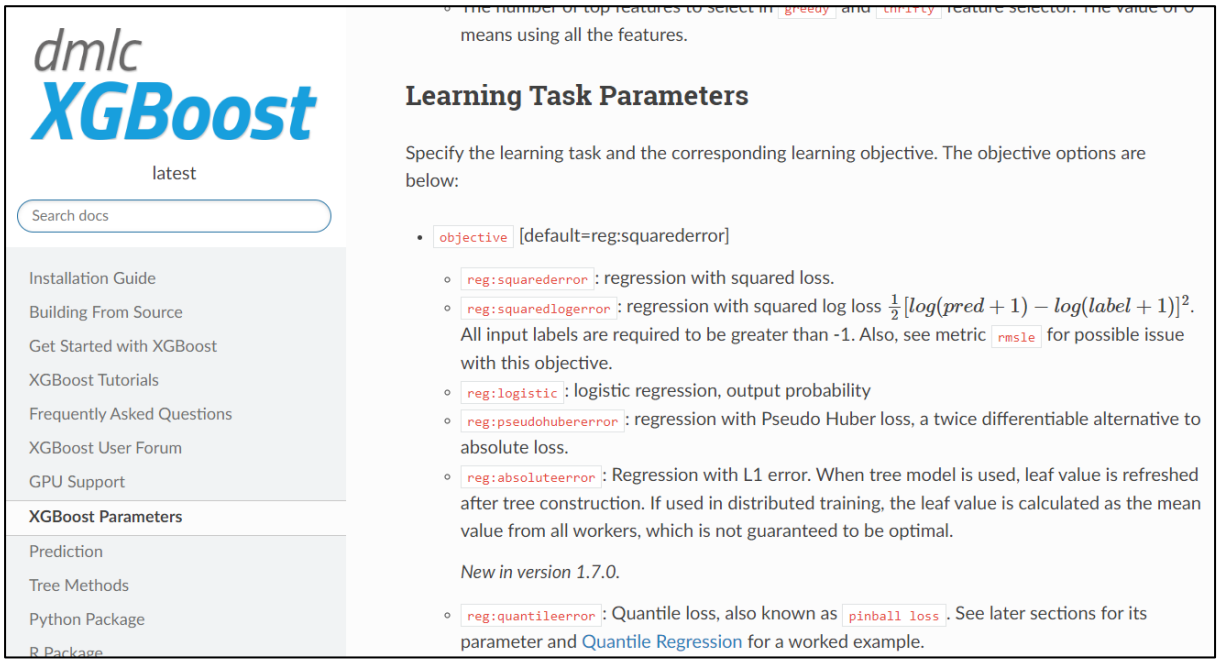


Рисунок 4.24 – Декілька варіантів метрики-параметра «objective» в моделі XGBoost

Аналогічні декілька сторінок для варіантів метрики в LightGBM є на рис. 4.25.

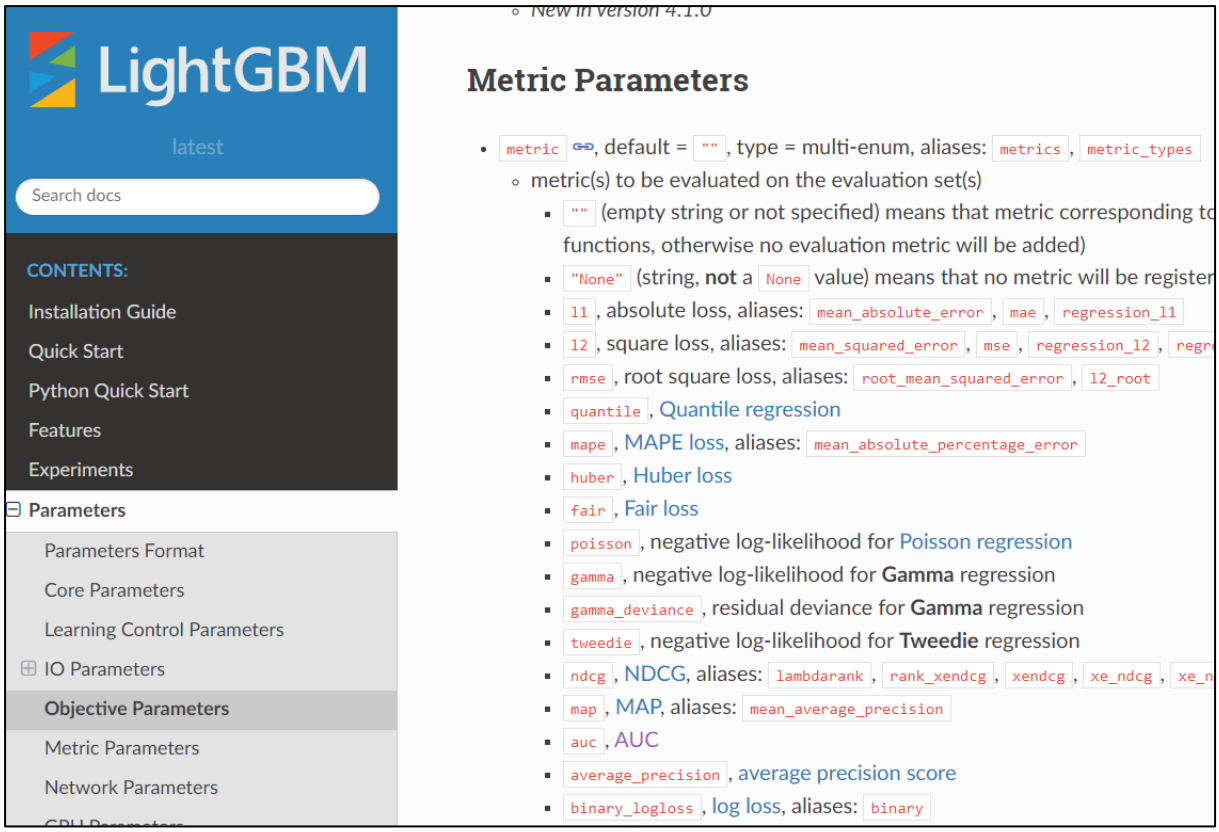


Рисунок 4.25 – Частина варіантів метрики для параметра «objective» в моделі LightGBM ()

На рис. 4.26 наведено приклад передбачення для 5 різних датасетів бустинговими моделями з різними параметрами.

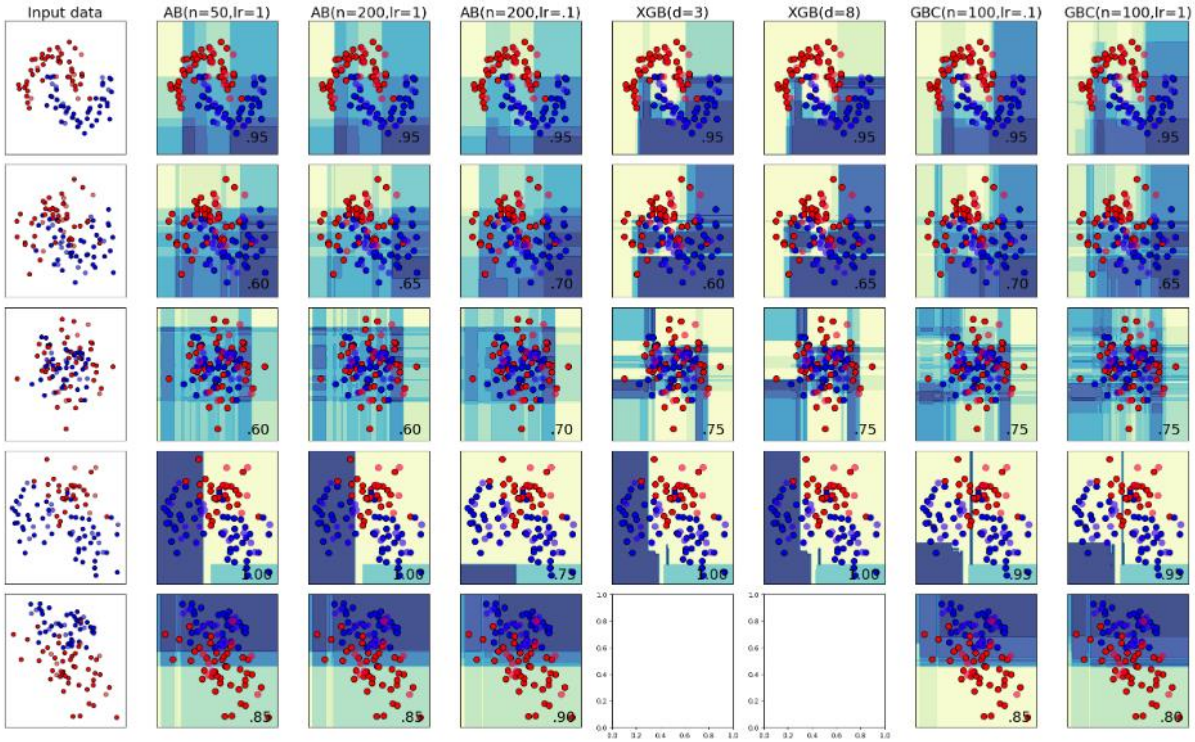


Рисунок 4.26 – Класифікація даних 5 різних датасетів бустинговими моделями: AdaBoostClassifier (AB), XGBoostClassifier (XGB), GradientBoostingClassifier (GBC) з різними параметрами: максимальна глибина (d), кількість дерев рішень в ансамблі (n), швидкість навчання (lr) (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [25]

Як видно на рис. 4.26, збільшення кількості дерев рішень n в ансамблі AdaBoostClassifier очікувано збільшує точність. Аналогічно – більша максимальна глибина d у моделі XGBoostClassifier та – менша швидкість навчання lr у моделях GradientBoostingClassifier і AdaBoostClassifier.

Отже, на рис. 4.26 найкращими моделями є «AB(n=200,lr=.1)» (1 місце на датасетах 1, 2, 5), «XGB(d=8)» (1 місце на датасетах 1, 3, 4) та «GBC(n=100,lr=.1)» (1 місце на датасетах 1, 2, 3).

У моделі XGBoost для 5-го датасету деякі комбінації параметрів видають помилку, тому його графіки на рис. 4.26 не наведені. Краще для цього датасету спробувати застосувати оригінальний xgb-метод, а не – спрощений варіант XGBClassifier. Вище зазначалось, що останній – не завжди спрацьовує чи дає прийнятну точність.

4.2.3 Ансамблі заданого типу моделей

Усі попередні види моделей, у т.ч. ансамблеві, основані на своїх, наперед сформованих методах аналізу даних та фіксованих типах моделей. Кожна має свої переваги і недоліки.

Але в бібліотеці **sklearn.ensemble** є ряд моделей, які працюють із заданим типом чи типами інших моделей бібліотеки sklearn, у т.ч. тими, які

були розглянуті вище у цьому розділі. В усіх них для цього є спеціальний параметр estimator чи estimators (з англ. – «оцінювач»). Такими моделями є наступні, назвемо їх агрегатори (їх англомовні назви і є назвами моделей у бібліотеці **sklearn.ensemble**):

- *бегінг* (англ. «**Bagging**» – скорочення від **B**ootstrap **agg**regat**ing**) – навчає моделі *одного* типу (estimator, за замовчуванням – це дерево рішень DecisionTree) на основі підмножин даних, а потім агрегує їх передбачення (для класифікаторів – голосування, для регресорів – усереднення); по суті, RandomForest та ExtraTrees це – теж бегінг дерев рішень, але, на відміну від цих моделей, бегінг може працювати й з іншими видами моделей-оцінювачів;

- *стекінг* (англ. «**Stacking**» – поєднання) – навчання кількох заданих у вигляді списку estimators моделей-оцінювачів і використання їхніх прогнозів як вхідних даних для головної моделі-оцінювача final_estimator (її ще називають *метамодель* – метакласифікатор чи метарегресор, залежно від типу задачі);

- *голосування* (англ. «**Voting**») – навчання кількох заданих у вигляді списку estimators моделей-оцінювачів і агрегування їхніх передбачень шляхом голосування одним із двох способів, залежно від параметра voting:

- Жорстке голосування (voting =«hard»), коли поелементно обирається клас, який передбачає більшість моделей (як середньозважене з однаковими вагами) – це є ефективним, коли моделі доволі різноманітні;

- М'яке голосування (voting =«soft»), коли вага голосів залежить від впевненості моделі у своєму передбаченні: фінальний клас обирається на основі суми ймовірностей для кожного класу, тобто – той, в якого ця сума є найбільшою – є ефективним, коли моделі співставні по точності.

У параметрі weights можна задати масив ваг для передбачень кожної моделі і тоді ці ваги будуть враховуватись під час голосування (працює для обох значень voting). Це є ефективним, коли деякі моделі більш впевнені, ніж інші.

Варіант утворення ансамблю для регресійної задачі з використанням VotingRegressor(voting =«hard»), де weights задається як масив ваг, часто заміняють звичайним середньозваженим. Тобто роблять передбачення різними моделями, зберігають у списках чи – у датафреймі і потім з певними вагами усереднюють. Див. наприклад, авторський ноутбук з прогнозом 4-ма моделями в конкурсі з передбаченням тих, хто вижив на «Титаніку». На рис. 4.27 наведена формула, за якою узагальнюються рішення моделей LightGBM, XGBoost, LogisticRegression, LinearRegression (дві останні мають значно менші ваги) – це рішення досягає Top4% у рейтингу за точністю з біля 16 тис. команд (насправді, – вище, оскільки перші місця займають особи, які, порушуючи правила конкурсу, просто скопіювали рішення, де були завантажені справжні, а не передбачені, дані про долю пасажирів, але, оскільки цей конкурс є тренувальним і не має терміну припинення, то їх не видаляють).

```
y_preds = w_lgb*y_preds_lgb + w_xgb*y_preds_xgb + w_logreg*y_preds_logreg + w_linreg*y_preds_linreg

submission['Survived'] = [1 if x>0.5 else 0 for x in y_preds]
```

Рисунок 4.27 – Приклад середньозваженого узагальнення передбачень 4-х моделей без використання ансамблевих агрегаторів бібліотеки Sklearn

Однак, варто зазначити, що варіант, наведений на рис. 4.27, можна покращити з використанням одного з наведених вище агрегаторів, оскільки вони мають ряд переваг, у порівнянні з простим об'єднанням передбачень. Деякі з них реалізовують більш складний алгоритм поєднання, деякі дозволяють реалізувати різну крос-валідацію під час тренування усіх моделей одночасно. Параметр n_jobs моделі Voting дозволяє реалізувати паралелізацію обчислень, що їх може значно прискорити.

У таблиці 4.2 наведено порівняльні характеристики агрегаторів бібліотеки Sklearn для утворення ансамблів моделей.

Таблиця 4.2 Характеристики агрегаторів бібліотеки Sklearn для утворення ансамблів моделей

Характеристика	**Стекінг**	**Беггінг**	**Голосування**
Принцип	Використання фінальної метамоделі-оцінювача агрегації передбачень базових моделей-оцінювачів	Використання одного типу моделі-оцінювача для навчання багатьох моделей на різних вибірках даних	Використання кількох моделей-оцінювачів для прийняття рішення шляхом голосування у різний спосіб
Тип моделей-оцінювачів	Різні типи, хоча може бути й один	Один тип	Різні типи, хоча може бути й один
Передбачення	Передбачення базових моделей використовуються як вхідні дані для метамоделі	Передбачення кожної моделі агрегуються	Передбачення кожної моделі агрегуються
Спосіб агрегування	Як вхідні ознаки для метамоделі	Для класифікаторів – поелементне голосування (подібне до режиму «hard» у Voting і	Агрегуються поелементно: є варіант «soft» і «hard», є варіант з використанням списку ваг weights

		це не можна змінити), для регресорів – усереднення	
Переваги	Можливість врахування взаємодії між базовими моделями	Зменшує перенавчання, поліпшує стійкість до викидів	Простий і ефективний спосіб комбінувати різні моделі
Застосування	Зазвичай використовується для великих датасетів та коли можна виявити корисні взаємодії	Застосовується до будь-якого алгоритму, особливо корисний для великих датасетів	Застосовується для поєднання результатів передбачень різних моделей в один фінальний результат

На рис. 4.28 наведено приклад передбачення для 5 різних датасетів ансамблями відібраних у підрозділах 4.1 і 4.2 найкращих моделей:

- Багінг дерев рішень («Bagging(DT)»),
- Багінг моделей RandomForest («Bagging(RF)»),
- Стакінг на основі передбачень методом опорних векторів, Ridge та логістичної регресії з фінальним класифікатором RandomForest («Stack(SVC,RD,LgR) RF»),
- Стакінг на основі передбачень деревом рішень, моделлю на основі гауссівських процесів та моделлю на основі методу k-найближчих сусідів з фінальним класифікатором RandomForest («Stack(DT,GP,KN) RF»),
- Ансамбль на основі м’якого голосування між передбаченнями RandomForest, моделі на основі гауссівських процесів та моделі на основі методу k-найближчих сусідів («Voting(RF,GP,KN,soft)»),
- Ансамбль на основі жорсткого голосування між передбаченнями RandomForest, моделі на основі гауссівських процесів та моделі на основі методу k-найближчих сусідів («Voting(RF,GP,KN,hard)»).

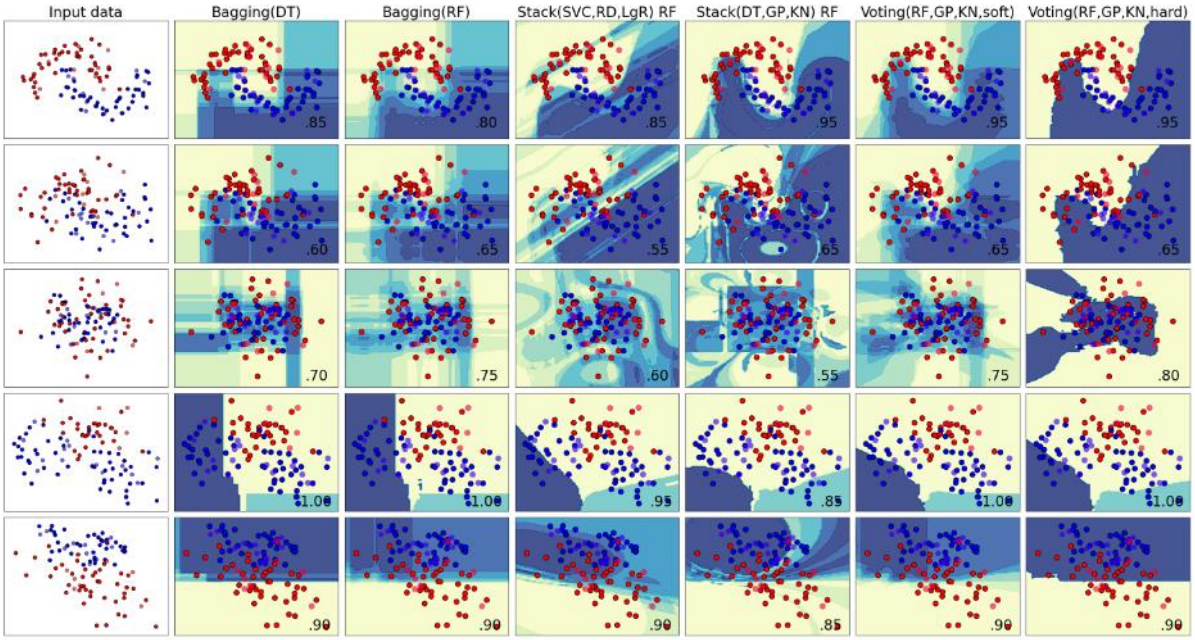


Рисунок 4.28 – Класифікація даних 5 різних датасетів ансамблями моделей: багінг, стакінг та голосування передбачень з різними комбінаціями моделей і параметрами (у правому нижньому куті – accuracy_score для тестових (валідаційних) даних, тестові дані – це кола, обведені чорним) [25]

Як видно на рис. 4.28, використання простих дерев рішень DT замість випадкових лісів RF під час багінгу погіршує точність для усіх цих датасетів, окрім першого, під час багінгу. Найкращі результати дає ансамбль на основі жорсткого голосування передбачень моделей.

Отже, на рис. 4.28 найкращими моделями є «Bagging(RF)» (1 місце на датасетах 2, 4, 5), «Voting(RF,GP,KN,hard)» (1 місце на УСІХ датасетах), моделі на основі стакінгу, вочевидь, оверфітяться – це видно, й по складній конфігурації графіків на рис. 4.28.

Важливо зауважити, що, як правило на етапі формування ансамблю використовують моделі з передтренованими оптимальними гіперпараметрами. Хоча, в ансамблі вони можуть спрацювати не так, як працювали, коли їх тренували відокремлено. Але на етапі ансамблювання моделей здійснювати тюнінг гіперпараметрів важко, через велику кількість їх комбінацій у різних моделях. Як правило, на етапі вже підбирають спосіб агрегування, ваги weights, операції післяоброблення та ін.

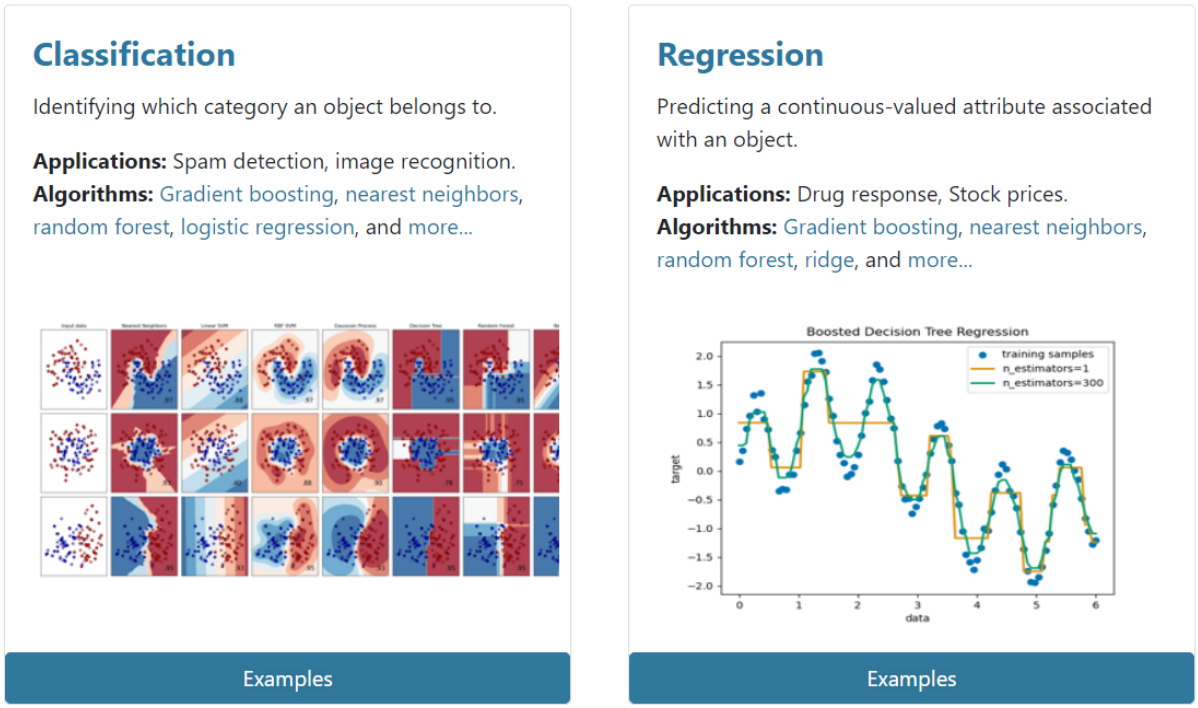
Іноді, до ансамблів ще відносять й використання конвеєрів операцій чи функцій. Іноді їх називають «пайплайни» (від англ. «pipeline»). В бібліотеці Sklearn є однойменна функція **Pipeline**, яка, дійсно, дозволяє задати послідовність (конвеєр) операцій з даними (рис. 4.29).

```
pipe = Pipeline([('scaler', StandardScaler()), ('svc', SVC())])
# The pipeline can be used as any other estimator
# and avoids leaking the test set into the train set
pipe.fit(X_train, y_train).score(X_test, y_test)
```

Рисунок 4.29 – Приклад конвеєра операцій в Sklearn

Однак, важливо зауважити, що, насправді, **Pipeline** не є ансамблем моделей, оскільки модель-оцінювач у ньому може бути тільки одного типу, а іншими можуть бути тільки різні операції передоброблення та післяоброблення даних. Якщо є бажання побудувати **Pipeline** саме з ансамблем, тоді як модель-оцінювач у ньому слід вказувати один з наведених вище агрегаторів, наприклад Voting зі списком декількох моделей і тоді до усіх них будуть застосовані ті самі операції передоброблення та післяоброблення даних.

Інші види моделей див. в документації Sklearn (рис. 4.30).



а) б)

Рисунок 4.30 – Моделі машинного навчання бібліотеки Sklearn: а) класифікатори, б) регресори

У ноутбуці [25, розділ 5] наведено найкращі з найкращих моделі з побудованих у підрозділах 4.1 і 4.2 для кожного із заданих 5 датасетів за метрикою accuracy_score. Відфільтровано моделі з ризиком оверфітінга, в яких різниця між похибкою на навчальних і тестових даних перевищує 0,1. На рис. 4.31 наведено по 4 найкращі моделі для кожного датасету. Щодо першого датасету (№ 0) таку ж точність мають ще моделі GPC, GBC, DT, AB. Отже, найчастіше найкращими є ансамблі моделей і бустингові моделі.

	model	num_dataset	acc_train	acc_test	diff
0	Voting(RF,GP,KN,soft)	0	1.000	0.95	0.050
1	Voting(RF,GP,KN,hard)	0	1.000	0.95	0.050
2	RF(d=8,n=1000)	0	1.000	0.95	0.050
3	RF(d=8,n=100)	0	1.000	0.95	0.050
4	KNeighbors (k=10)	1	0.838	0.85	-0.012
5	LogRegres (L2, C=0.1, lin)	1	0.775	0.85	-0.075
6	SGD (alpha=0.1)	1	0.762	0.85	-0.088
7	Ridge (alpha=1)	1	0.762	0.85	-0.088
8	RF(d=5,n=100)	2	0.850	0.80	0.050
9	Voting(RF,GP,KN,hard)	2	0.838	0.80	0.038
10	RF(d=5,s=5)	2	0.812	0.80	0.012
11	Voting(RF,GP,KN,soft)	2	0.838	0.75	0.088
12	XGB(d=8)	3	1.000	1.00	0.000
13	XGB(d=3)	3	1.000	1.00	0.000
14	RF(d=8,n=1000)	3	1.000	1.00	0.000
15	RF(d=8,n=100)	3	1.000	1.00	0.000
16	Voting(RF,GP,KN,soft)	4	0.988	0.90	0.088
17	Voting(RF,GP,KN,hard)	4	0.988	0.90	0.088
18	RF(d=5,n=100)	4	0.988	0.90	0.088
19	DT(d=5,gini,s=1)	4	0.988	0.90	0.088

Рисунок 4.31 – Класифікація даних 5 різних датасетів найкращими моделями [25]

Важливо пам’ятати, що це – лише приклад побудови моделей та їх аналізу. Зроблені по них висновки не можна поширювати на усі інші подібні задачі. Для кожної задачі потрібний свій аналіз. Більше того, було проаналізовано тільки декілька варіантів параметрів. Насправді, тюнінг моделей слід проводити в інший спосіб – з використанням спеціальних засобів, які дозволяють вибрати дійсно оптимальний набір параметрів (гіперпараметрів) цих моделей. Цьому буде присвячено наступний підрозділ.

На рис. 4.32 наведена інфографіка інструментарію, згаданого у підрозділах 4.1 та 4.2 у системі координат *S(I)*.

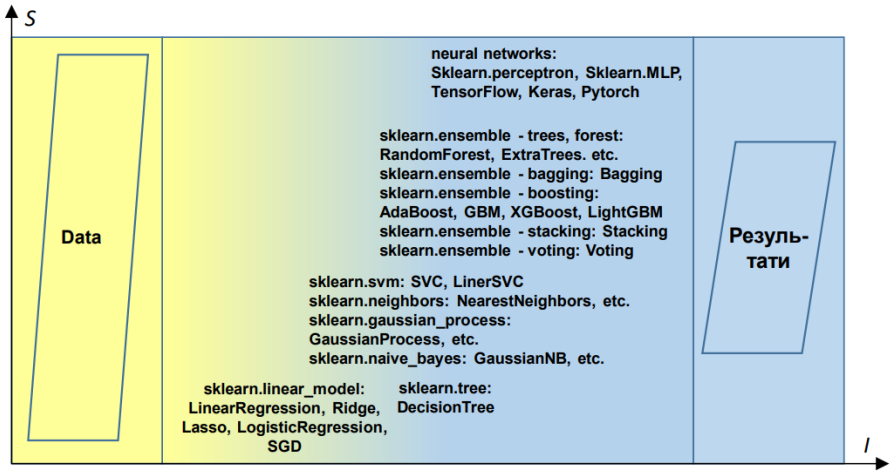


Рисунок 4.32 – Інфографіка моделей машинного навчання та їх ансамблів

4.3 Навчання («tuning») гіперпараметрів моделей та перевірка ефективності їх навчання

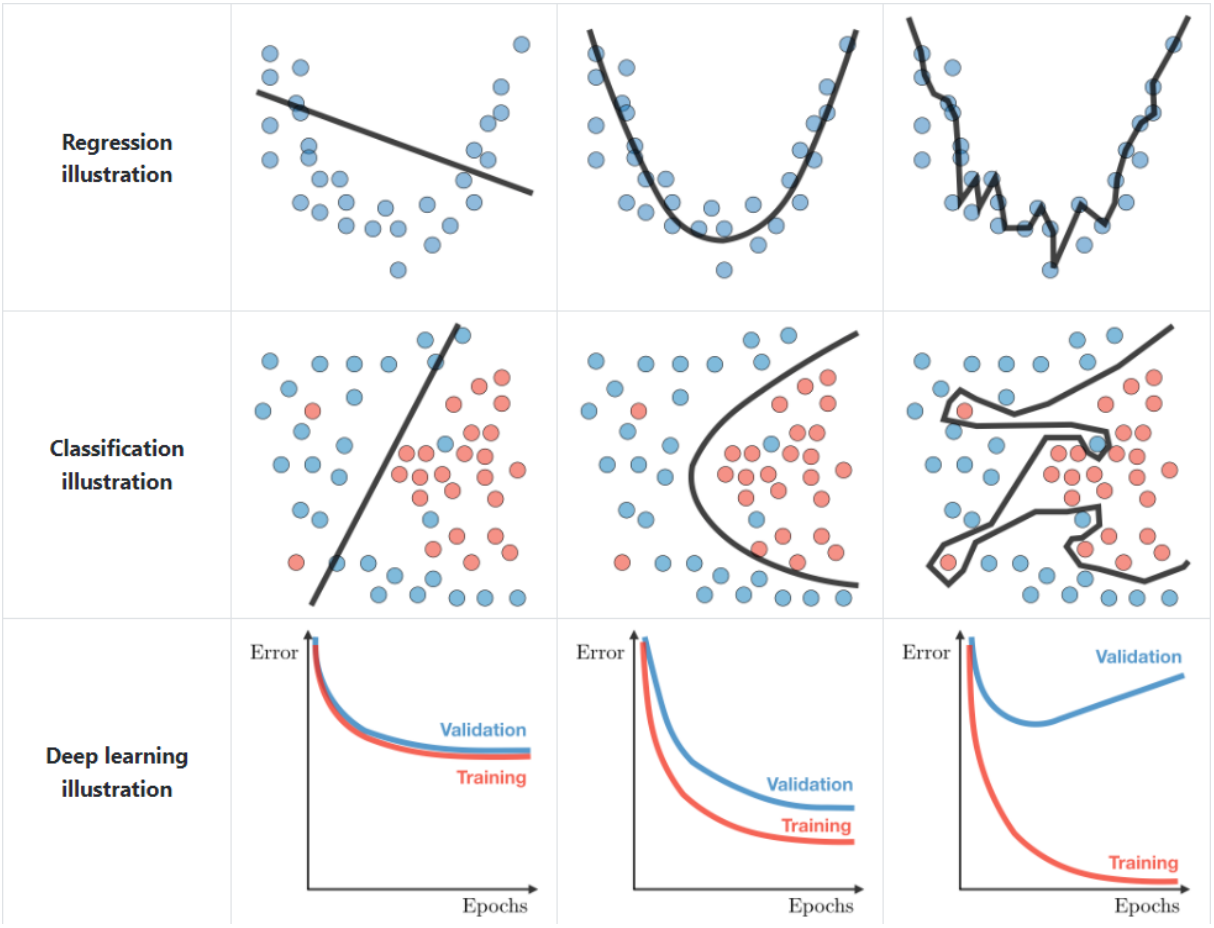
Ключовим етапом машинного навчання є, власне, навчання моделей (звідси – й назва). Метою навчання є досягнення ряду цілей (у напрямку зменшення важливості):

- досягнення кращого значення метрики на валідаційному датасеті val_loss;
- відсутня чи мінімальна різниця між значенням метрики на тренувальному train_loss і валідаційному val_loss датасетах;
- менша тривалість обчислень;
- має меншу вартість обчислень, у т.ч. – менші вимоги щодо необхідних обчислювальних потужностей у вигляді GPU чи TPU.

Тобто серед моделей з однаковими дуже гарними значеннями, наприклад val_loss, слід вибрати ту, в якій train_loss=val_loss (чи майже є таким), а якщо й це збігається, тоді – ту, яка швидше здійснює передбачення (чи й навчання теж) а ще є дешевшою, тобто вимагає менші потужності для обчислень.

Основною метою є досягнення кращого значення метрики на валідаційному датасеті val_loss, але важливо, щоб і значення loss на тренувальних даних теж було гарним і несильно відрізнялось від val_loss (хоча б не більше, ніж на 5-10%). Відповідно до їх співвідношень цих значень, розрізняють 3 види результатів машинного навчання (рис. 4.33):

- *недонавчання* (англ. «Underfitting») (рис. 4.33а): значення loss – погане, тоді значення val_loss особливого значення вже не має, або loss= val_loss;
- *гарне навчання* (англ. «Just Right» або «Appropriate-Fitting») (рис. 4.33б): і val_loss, і loss – гарні чи хоча б задовільні (val_loss має бути гіршим за loss, але незначно – наприклад на 3-10%);
- *перенавчання* (англ. «Overfitting») (рис. 4.33в): loss – гарне, але val_loss – погане або loss є кращим за val_loss більше, ніж на 10%.



а) б) в)

Рис. 4.33 – Види результатів машинного навчання моделі: а) недонавчання; б) гарне навчання; в) перенавчання

Зазвичай, у разі гарної реалізації моделей і порад з підрозд. 4.1 і 4.2, можна добитись гарного loss, а тоді головною проблемою стає саме перенавчання, яке має місце у дуже багатьох випадках. Тому я часто говорять, що основна задача машинного навчання – це уникнути саме перенавчання! Існує ряд прийомів як можна цього добитись.

1. Використання крос-валідації для вибору тренувального і валідаційного датасетів (див. підрозд. 1.2 та пораду «Тір 6.1» в [11]).

2. Використання функції sklearn.model_selection.GridSearchCV, яка дозволяє здійснювати повний перебір варіантів параметрів моделі у заданих списках дискретних значень із заданою крос-валідацією (див. пораду «Тір 6.2» в [11]).

Є ще варіант RandomizedSearchCV – те саме, що – GridSearchCV, але без повного перебору варіантів. Беруться тільки певні випадкові комбінації – не так точно, але працює набагато точніше і можна пробувати більше варіантів, у т.ч. задавати неперервні діапазони значень.

В першу чергу, варто налаштовувати:

- архітектуру моделей, якщо у ній передбачено використання різних складових;

- для дерев рішень: максимальну глибину (max_depth) та/або кількість листя (num_leaves);

- швидкість навчання learning_rate (lr);

Окрім тюнінгу гіперпараметрів самої моделі, ще можуть бути ефективними інші прийому підвищення точності моделей:

- параметри крос-валідації (cv);

- зміна розміру чи кількості партій-батчів (batch);

- спосіб вибору випадкових значень під час тюнінгу моделі (random_state).

Під час тюнінгу слід періодично аналізувати різні комбінації параметрів для пошуку найбільш вдалих та формування плану їх подальшої зміни. В авторських ноутбуках [GRU & LSTM mix & custom loss - tuning by 3D visual, Stock Embedding - FFNN - upgrade & 3D, MoA: Pytorch-RankGauss-PCA-NN upgrade & 3D visual] є гарні 5-вимірні (3D-координати + форма + колір) візуалізації того, як різні параметри впливають на точність. За тими графіками вдалось добре покращити цю точність.

Для автоматизації процесу налаштування існують спеціальні методи. Найбільш популярними серед них є такі:

- GridSearchCV з sklearn.model_selection

- HyperOpt з однойменної бібліотеки hyperopt

- Optuna з однойменної бібліотеки optuna

Їх порівняльний аналіз наведено у таблиці 4.3.

Таблиця 4.3 – Переваги та недоліки методів автоматизації тюнінгу параметрів моделей машинного навчання

Метод	**GridSearchCV**	**HyperOpt**	**Optuna**
Принцип оптимізації	Сітка заздалегідь визначених значень параметрів	Баєсівська оптимізація	Адаптивна оптимізація
Спосіб вибору оптимальної моделі	Повний перебір усіх комбінацій значень	Обчислюють задану кількість (10, 20 чи ін.) найкращих моделей і з них вибирають оптимальну	Обчислюють задану кількість (10, 20 чи ін.) найкращих моделей і з них вибирають оптимальну
Як задаються параметри?	Як правило, як списки варіантів значень, іноді – константи, коли одразу треба задати якесь значення	Як списки варіантів значень чи константи, але можуть задаватись і як діапазон значень з певним кроком зміни	Як списки варіантів значень чи константи, але можуть задаватись і як діапазон значень з певним кроком зміни

Можлива кількість варіантів значень параметрів	Мала кількість	Велика кількість	Велика кількість
Глобальність оптимуму	Гарантовано глобальний оптимум, оскільки перебираються усі комбінації допустимих значень	Є ризик, що глобальний оптимум не буде знайдено, але використовується ряд прийомів, які дозволяють зменшити цей ризик	Є ризик, що глобальний оптимум не буде знайдено, але використовується ряд прийомів, які дозволяють зменшити цей ризик – вважається, що у цього методу цей ризик є меншим, у порівнянні з методом HyperOpt
Переваги	1. Простий у використанні і розумінні 2. Працює добре з невеликою кількістю гіперпараметрів	1. Робить менше ітерацій (більш швидкий) для отримання гіперпараметрів, ніж інші методи 2. Дозволяє задавати неперервні гіперпараметри 3. Враховує неоднакову важливість гіперпараметрів	1. Підтримує розподілені обчислення 2. Доволі швидкий, якщо порівнювати з GridSearchCV
		4. Може розглядати неперервні гіперпараметри	
Недоліки	1. Найтриваліший серед усіх методів, через перебір усіх значень і відсутність паралельних обчислень. 2. Не ефективний для великої	1. Вимагає додаткового часу для налаштування моделі. 2. Складніший для налаштування.	1. Вимагає додаткового часу для налаштування моделі. 2. Складніший для налаштування.

	кількості гіпер-параметрів чи – для декількох параметрів, але багатьма значен-нями		

Гарні приклади застосування GridSearchCV та HyperOpt (рис. 4.34) є у авторському ноутбуку для передбачення тих, хто вижив на «Титаніку» (у конкурсі Kaggle) – див. у змісті підрозділи:

- k-Nearest Neighbors algorithm with GridSearchCV,
- Random Forests with GridSearchCV,
- XGB Classifier with HyperOpt,LGBM Classifier with HyperOpt,
- GradientBoostingClassifier with HyperOpt,
- ExtraTreesClassifier with HyperOpt,
- VotingClassifier (soft voting) with GridSearchCV).

```
0.9338137951554497
{'booster': 'gbtree', 'colsample_bytree': 0.65, 'eval_metric': 'auc', 'gamma': 0.78, 'learning_r
ate': 0.0349, 'max_depth': 7, 'min_child_weight': 4.4, 'missing': None, 'n_estimators': 600, 'ob
jective': 'binary:logistic', 'silent': 1, 'subsample': 0.53, 'tree_method': 'exact'}
0.9371656795353692
{'booster': 'gbtree', 'colsample_bytree': 0.9500000000000001, 'eval_metric': 'auc', 'gamma': 0.6
900000000000001, 'learning_rate': 0.0325, 'max_depth': 5, 'min_child_weight': 4.475000000000000
5, 'missing': None, 'n_estimators': 815, 'objective': 'binary:logistic', 'silent': 1, 'subsampl
e': 0.685, 'tree_method': 'exact'}
100%|██████████| 10/10 [00:20<00:00,  2.07s/it, best loss: 0.9260240223818323]
best:
{'colsample_bytree': 0.8, 'gamma': 0.595, 'learning_rate': 0.0015, 'max_depth': 2, 'min_child_we
ight': 8.575000000000001, 'n_estimators': 870, 'subsample': 0.5700000000000001}
CPU times: user 20.8 s, sys: 350 ms, total: 21.1 s
Wall time: 21.1 s
```

Рисунок 4.34 – Приклад тюнінгу параметрів моделі XGBClassifier методом HyperOpt: найкращі комбінації параметрів (наведено 2 останні з 10) та вибрана оптимальна комбінація параметрів моделі (див. останній словник {}) авторського ноутбуку

Гарні приклади застосування методу Optuna з детальними поясненнями та інфографікою є у ноутбуку. Зазвичай, алгоритм його застосування такий:

1. За результатами розвідувального аналізу вибирають можливі діапазони значень.
2. Застосовують метод HyperOpt чи Optuna і суттєво звужують можливі діапазони значень та кількість гіперпараметрів, які варто змінювати і які найбільше впливають на точність.
3. Застосовують GridSearchCV для уточнення глобального оптимуму для відібраних у п. 2 варіантів значень і гіперпараметрів.

Хоча, іноді, алгоритм завершується на п.2, а іноді, за малої кількості можливих варіантів значень, п.2 опускають і використовують тільки GridSearchCV.

Криві навчання та їх аналіз.

Під час аналізу точності навчання моделей важливо не тільки рахувати основну метрику, а й - аналізувати криву навчання та конфузійну матрицю.

Крива навчання (англ. «Learning Curve») відображає залежність метрики від обсягу тренувальних даних або від кількості ітерацій навчання. По ній видно: як краще вибирати параметри крос-валідації? Чи рівномірно розподілені дані між партіями? Чи є перенавчання чи недонавчання і які етапи крос-валідації збільшують цей ризик? Крива навчання допомагає оцінити, як модель вчиться і як змінюється її ефективність з часом чи від за різних даних. Зазвичай, будуються 2 криві навчання: тренувальна крива (англ. «Training Curve») та валідаційна крива (англ. «Validation Curve»). Цінним є аналіз не тільки їх самих, а й - їх співставлення. Оптимальним варіантом є ситуація, коли валідаційна крива в кінці навчання наближається до тренувальної, але має трохи гіршу за неї точність, і, при цьому, вони обидві досягають гарних значень щодо точності.

Конфузійна матриця (англ. «Confuse Matrix») або «матриця помилок» (англ. «Error Matrix») – це квадратна матриця, розмір якої дорівнює кількості класів цільової ознаки, тому вона будується тільки в класифікаційних задачах. Кожен із рядків цієї матриці відповідає класам, які передбачаються, а кожен стовпець – справжнім класам. У кожній комірці відображається відносна кількість правильно передбачених відповідних класів (може бути ще й абсолютне значення цієї кількості та скільки даних слід було передбачити усього). Ідеальна конфузійна матриця – це одинична діагональна матриця, тобто матриця, в якій в діагоналі – усі 1, а в інших комірках – 0. Для бінарного таргета конфузійна матриця наведена на рис. 4.35.

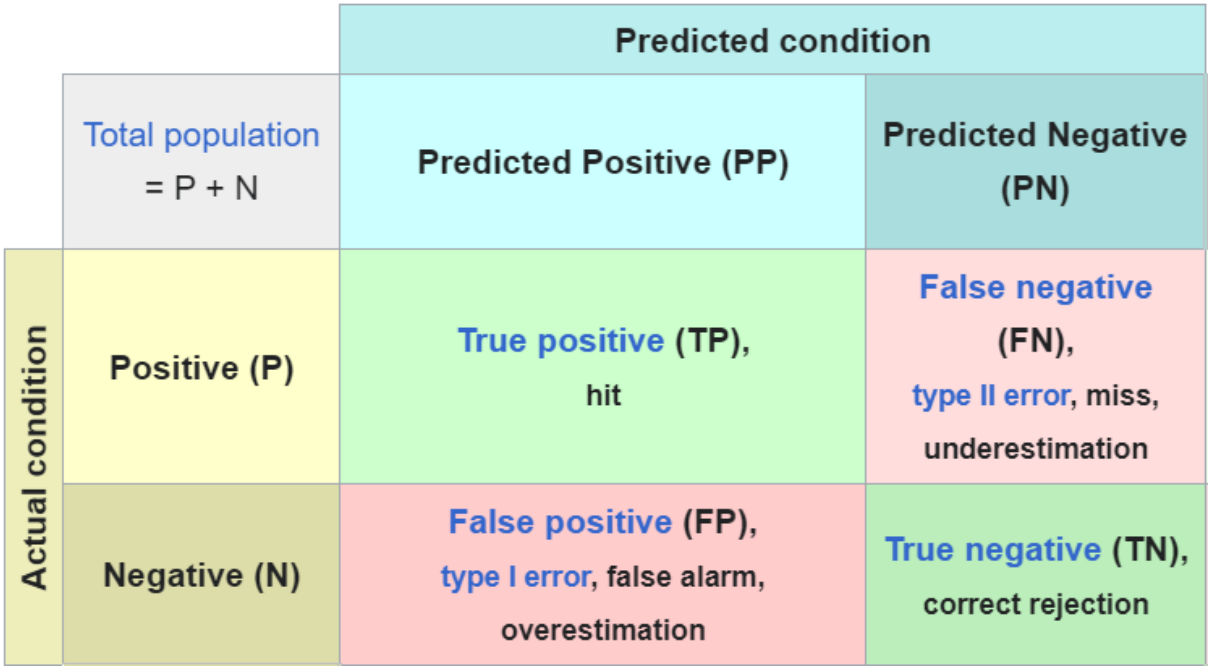


Рисунок 4.35 – Структура конфузійної матриці

Похибка «False Positive» (FP) ще називається «похибка I роду» чи «хибна тривога», а похибка «False Negative» (FN) – «похибка II роду» чи «пропуск цілі». Велике значення FP свідчить про перенавчання, а велике FN – про недонавчання.

За цими похибками визначаються ще 2 важливі метрики Precision і Recall, які використовуються для визначення ефективності моделей для бінарної класифікації:

- *Точність* (англ. «*Precision*» або «True Positive Rate» (TPR)) – визначає, наскільки точними є позитивні (таргет = 1) прогнози моделі:

$$TPR = \frac{TP}{TP+FN}. \quad (4.1)$$

- *Повнота* (англ «*Recall*» або «Positive Predictive Value» (PPV)) – визначає, яку частину всіх позитивних (таргет = 1) екземплярів модель визначила правильно.

$$TPR = \frac{TP}{TP+FP}. \quad (4.2)$$

Якщо важливо вибрати модель, яка має збалансовані значення обох цих метрик, тоді використовують метрику F_1 (гармонійне середнє):

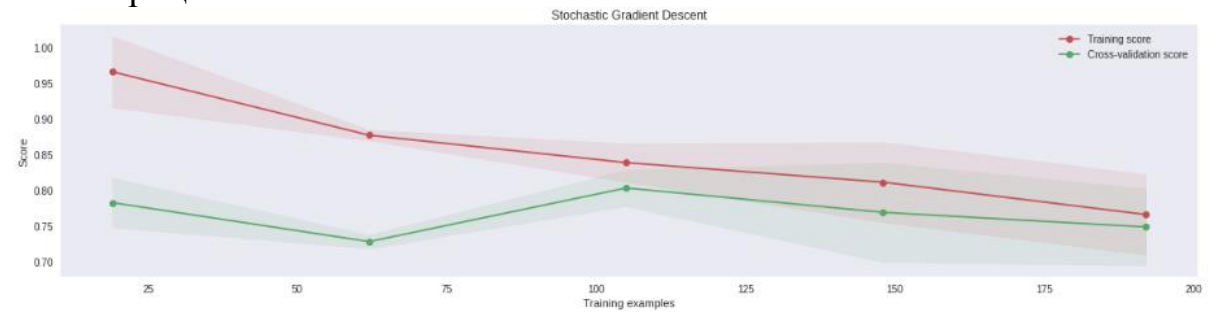
$$F_1 = 2\frac{PPV*TPR}{PPV+TPR} = \frac{2TP}{2TP+FP+FN}. \quad (4.3)$$

У конкурсах Kaggle ще популярними є метрики:

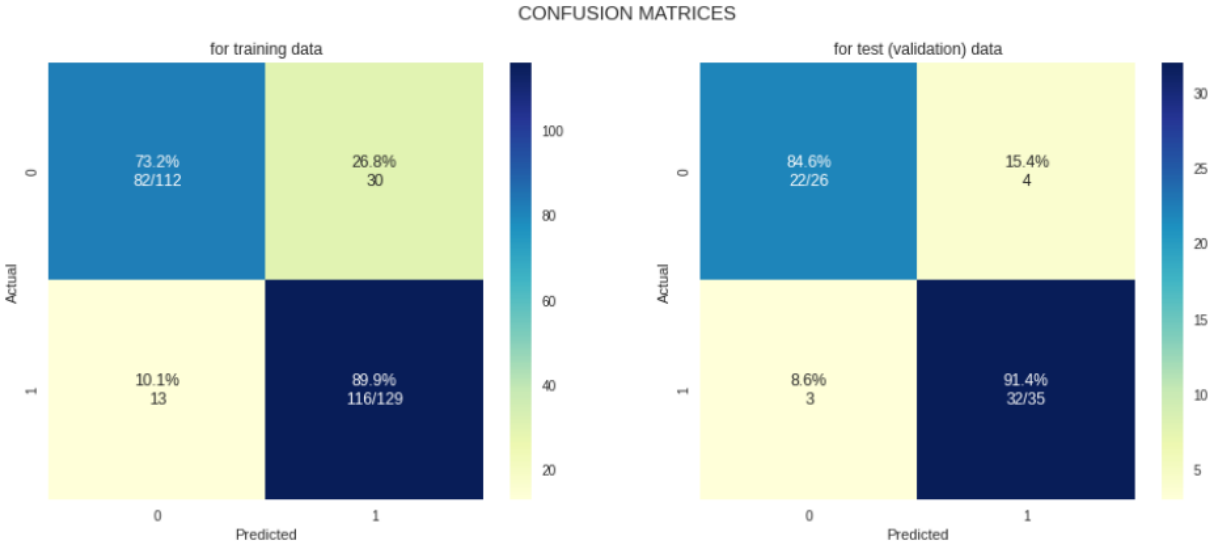
- ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) – площа під кривою ROC, яка графічно відображає залежність між долею TP і FPR при різних значеннях порогу у класифікаційній моделі; чим більше значення AUC (від 0 до 1), тим – краща модель, де 1 вказує на ідеальну модель, а 0,5 вказує на модель, яка класифікує із випадковим вибором;

- F_2 – для випадків, коли важливіше надавати велику вагу повноті, особливо в задачах, де важливо якнайменше уникати хибно-негативних результатів.

На рис. 4.36 та 4.37 наведено приклади кривих навчання та конфузійних матриць.

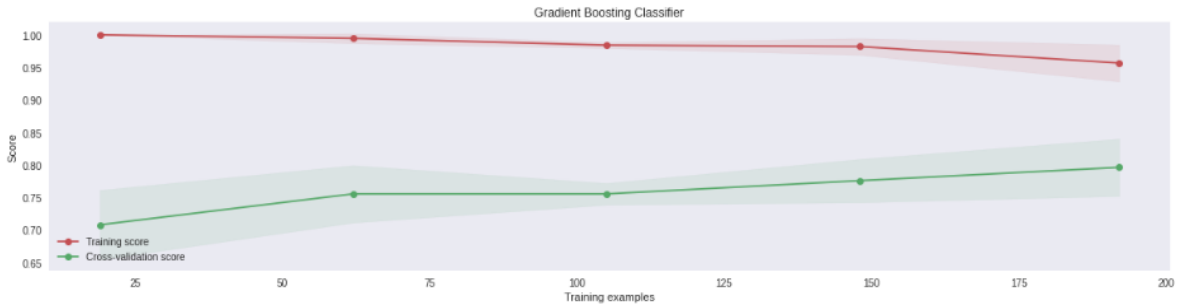


а)

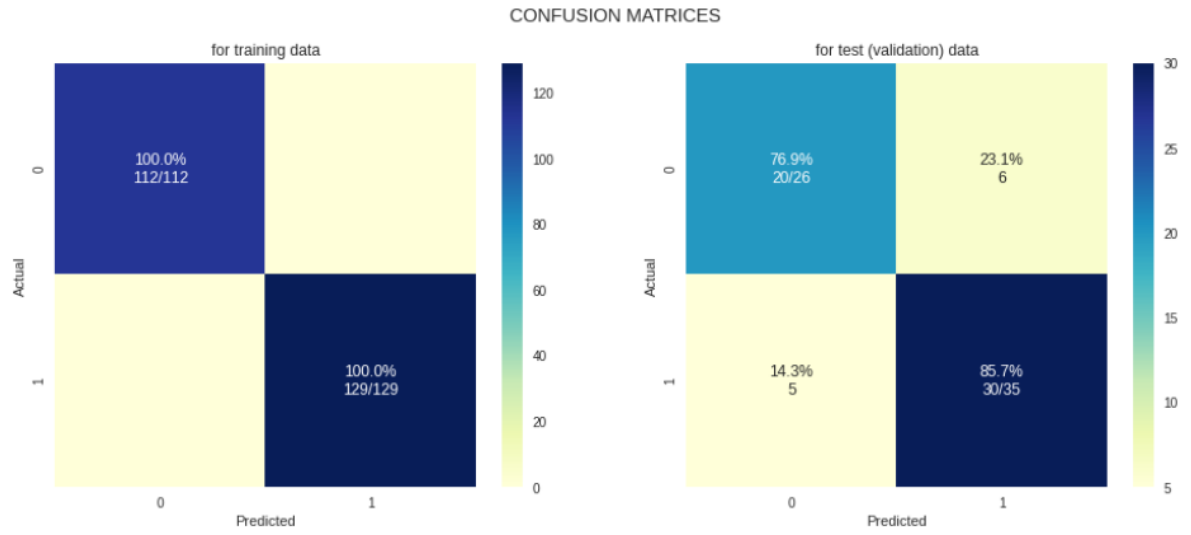


б)

Рисунок 4.36 – Аналіз результатів навчання моделі «Stochastic Gradient Descent» з авторського ноутбуку: а) криві навчання; б) конфузійні матриці



а)



б)

Рисунок 4.37 – Аналіз результатів навчання моделі «GradientBoostingClassifier» з авторського ноутбуку:
а) криві навчання; б) конфузійні матриці

Аналіз рис. 4.36 показує, що модель нормально навчилась, але додавання партій погіршувало її точність, тому варто переглянути параметри крос-валідації. А на рис. 4.37 конфузійна матриця свідчить про явний оверфітінг. Причому, гірше передбачаються значення target=0. Для зменшення оверфітінгу слід оптимізувати параметри моделі. Крива навчання свідчить, що додавання партій дозволяє підвищити точність передбачення валідаційних даних, але недостатньо, тобто і параметри крос-валідації варто оптимізувати теж.

Варто зауважити, що конфузійна матриця може будуватись не тільки для задач бінарної класифікації. На рис. 4.38 наведена конфузійна матриця для задачі розпізнавання і класифікації арабських цифр.

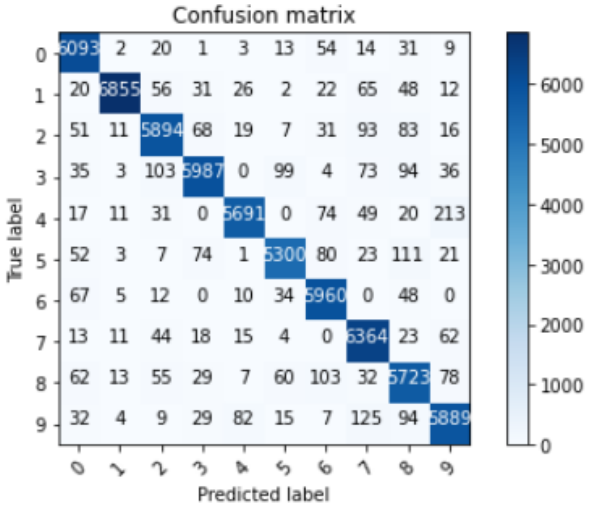


Рисунок 4.38 – Конфузійна матриця для задачі розпізнавання і класифікації арабських цифр з авторського ноутбуку

Метрики Precision, Recall, F_1 теж мабуть свої аналоги для багатокласових задач.

4.4 Інтелектуальний аналіз та передбачення даних, проведені призерами змагань Kaggle

У підрозд. 2.4 і 3.4 наведено приклади ефективного інтелектуального розвідувального аналізу даних та аналізу й оброблення ознак, які дозволили їх авторам отримати призові місця у конкурсах Kaggle. Але, у більшості конкурсів таких вмінь недостатньо і слід вміти не тільки добре виконувати й завдання інших етапів:

– доповнення даних іншими, у т.ч. передбачення тестових даних з використанням псевдо-міток, аугментація даних та ін.;

– створення ансамблів моделей машинного навчання;

– розробка алгоритмів зменшення розмірності для вибору найбільш інформативних ознак та зменшення обсягу обчислень;

– Використання технік збалансування класів для поліпшення моделей у випадку незбалансованих даних;

– Вирішення проблеми надмірного навчання (overfitting) за допомогою регуляризації та інших методів контролю складності моделей;

– Використання технік інтерпретації моделей для розуміння причинно-наслідкових зв'язків та забезпечення довіри до результатів;

Наведені у цьому розділі моделі можуть застосовуватись до розв’язання різних прикладних задач на основі таблиць даних, які є популярними серед призових конкурсів Kaggle:

– взаємодія молекул, атомів у матеріалах для задач хімії, фізики, техніки;

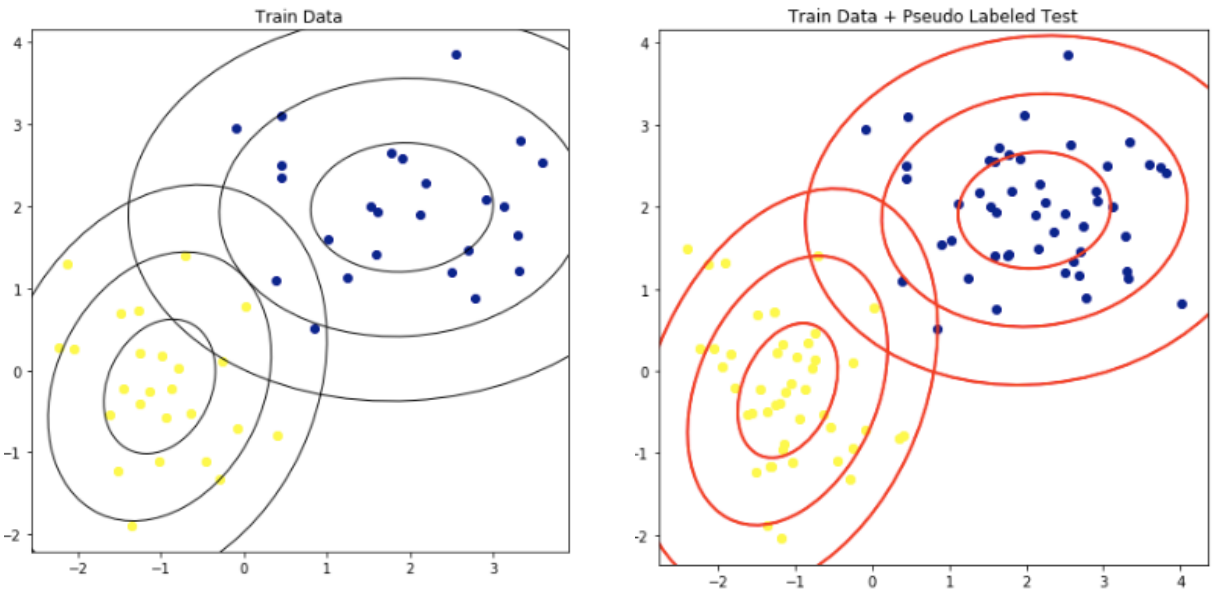
– взаємодія клітин або генів у живих організмах для задач біології, хімії, медицини;

– передбачення рішень на основі аналізу банківської інформації та пояснення прийнятих рішень;

– передбачення продаж в Інтернет-магазинах;

– рекомендаційні системи у сфері торгівлі на основі аналізу історії продаж та у сфері медицини на основі історії хвороби пацієнта.

У підрозд. 3.4 був згаданий конкурс. У ньому 1 місце посіло рішення, де автор застосував цікавий прийом з доповнення тренувальних даних передбаченням тестових даних з використанням *псевдо-міток* (рис. 4.39).



а) б)

Рисунок 4.39 – Дані без а) та з б) псевдо-мітками

Цей же прийом псевдо-міток використав автор ноутбуку «3 Clusters Per Class - [0.975]», який посів 7-ме місце (золота медаль, але без грошового призу) в іншому конкурсі Kaggle – «Instant Gratification».

Звичайно, його застосування могло б обмежитись тільки хитрим прийомом як перемогти в конкурсі Kaggle, «вгадавши відповіді», але така реалізація може бути й корисною в реальних задачах, коли є певна апріорна інформація про можливі тестові дані або з використанням цього підходу можна гарантувати певну точність моделі, за рахунок внесення певних обмежень на тестові дані, які б підходили тіж ті песвдо-мітки, які враховує це рішення.

Конкурс Elo Merchant Category Recommendation полягає в розробці алгоритмів машинного навчання, які допоможуть ідентифікувати та надавати найбільш відповідні пропозиції для клієнтів на основі їхньої лояльності. Завдання полягає в тому, щоб зрозуміти, які пропозиції працюють краще для клієнтів та торговців, та створити персоналізований досвід для клієнтів. Це включає в себе розробку моделей, які аналізують дані про покупки та поведінку клієнтів, щоб забезпечити їм найкращі пропозиції та зменшити кількість непотрібних рекламних кампаній.

Перше рішення включає в себе декілька моделей: регресію для прогнозування, бінарну класифікацію для виявлення викидів, а також моделі з низькою та високою ймовірністю. Моделі були розроблені з метою передбачення транзакцій та виявлення аномалій, таких як карти без повторних транзакцій у тестовому періоді. Для різних груп карт були створені окремі моделі регресії з різними підходами до обробки аномальних викидів, що допомогло уникнути етапу післяоброблення, який виявився головним джере-

лом "shakeup". У кінцевому результаті прогнози з обох моделей були поєднані та змішані з результатами регресії для підготовки остаточного набору прогнозів (рис. 4.40).

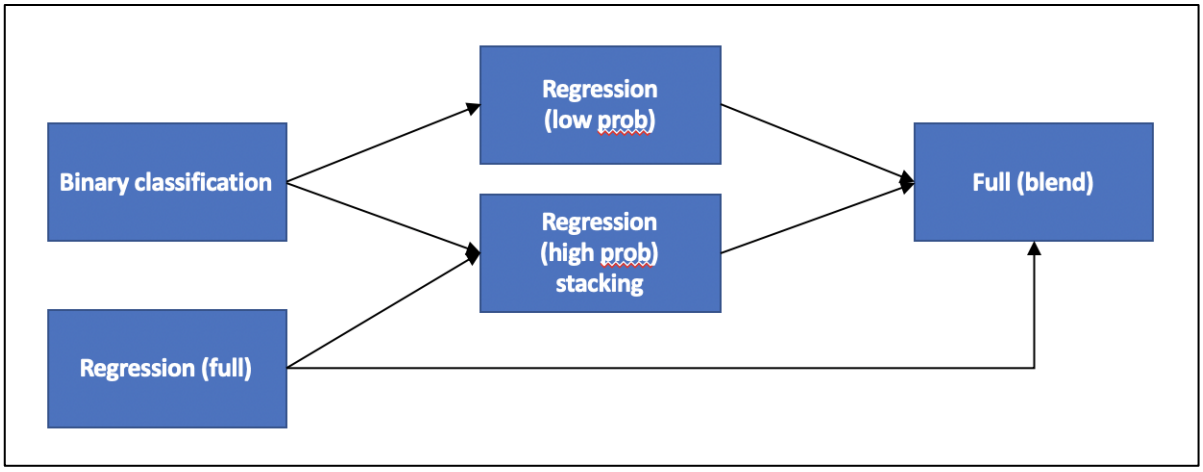


Рисунок 4.40 – Накопичення та змішування регресійних моделей

Друге рішення включає в себе використання декількох моделей для прогнозування, аналізу та оброблення даних про транзакції. Попередні оброблення включали сортування даних за часом та створення різних наборів даних з різними характеристиками. Велика увага була приділена побудові нових ознак, з урахуванням агрегації, аналізу інтервалів та взаємозв'язку між різними факторами, а також – використанню різних моделей, у т.ч. LightGBM та нейронних мереж. Після цього застосовувалися ансамблі та післяоброблення результатів для підвищення точності моделей, що зображено на рисунку 4.41.

$$target(card\_id) = \frac{\sum_m sum\_future\_purchase\_amount(m, card\_id)/2}{\sum_m last\_historic\_transaction\_purchase\_amount(m, card\_id)}$$

Рисунок 4.41 – Ядро симуляції непомічених даних - використання Magic FE, включаючи комбінаторику

Рішення 3 включало в себе ретельну передобробку даних і створення широкого набору ознак за допомогою різних методів, таких як об'єднання даних, використання Count Vectorizer, PCA, SVD, розрахунок статистик за місяцями, а також розрахунок різних відношень і середніх значень. Відбір ознак виконувався за допомогою Boruta, а також моделей класифікації, таких як LGBM, XGB та інші з використанням адверсарних перевірок для вибору кращих ознак.

Це рішення, яке зайняло 11 місце, використовувало широкий спектр методів для побудови моделі прогнозування. Основні складові роботи включали передобробку даних та створення багатьох різноманітних ознак, таких

як агрегаційні функції та взаємодії між різними факторами. Після цього використовувалися різні моделі для навчання, такі як LightGBM, Xgboost, H2oRF і H2oGBM, а також стекінг близько 32 моделей. Додатково було проведено постпроцесінг для виявлення та видалення викидів. Хоча використано багато методів, не всі з них принесли очікуваний результат, але команда визнає важливість коректної крос-валідації.

Це рішення, яке зайняло 16-те місце, складалося з кількох ключових етапів, таких як відбір ознак та моделювання. Для відбору ознак використовувалися підходи, що базувалися на порівнянні розподілів ознак у навчальному та тестовому наборах даних, а також – на техніці перемішування тесту для оцінювання важливості ознак. Після відбору ознак була використана модифікована модель, яка враховувала виявлені «викиди». На підставі цих етапів було побудовано кілька моделей, включаючи найкращу одиночну модель та комбіновану модель, яка використовувала інформацію про виявлені «викиди» , що зображено на рисунку 4.42.

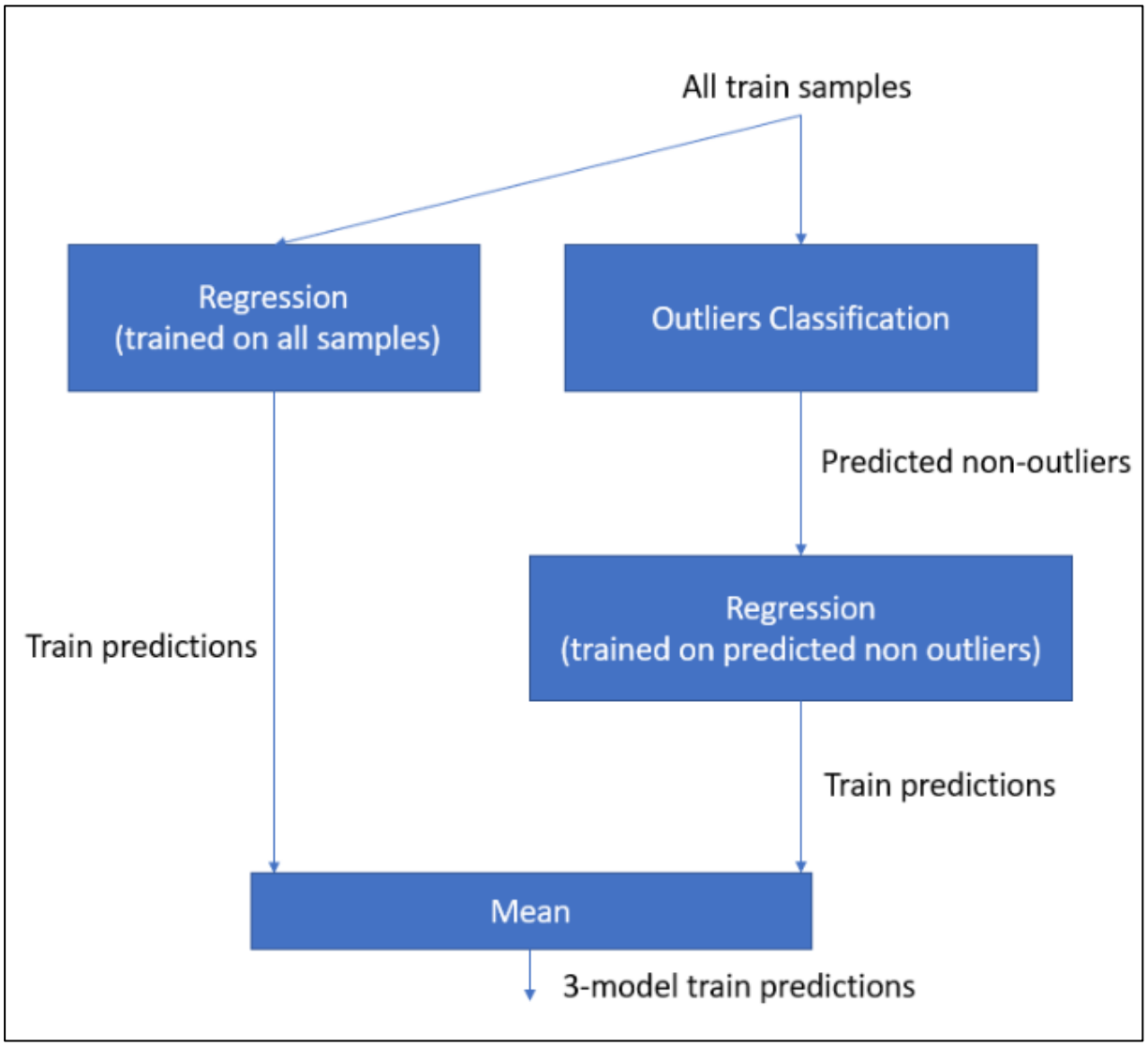


Рисунок 4.42 – Модифікована 3-модельна модель

У конкурсі AMP®-Parkinson's Disease Progression Prediction треба було передбачити появу хвороби Паркінсона. Перше місце зайняв автор рішення, яке полягає у використанні двох моделей - LGB та NN - для прогнозування хвороби Паркінсона на основі даних з медичних візитів пацієнтів. Рішення засноване на простому усередненні прогнозів цих моделей. Основні ознаки для прогнозування включають місяць візиту, прогнозований горизонт, місяць прогнозування цілей, показник взяття крові під час візиту та інші. Також були спроби використати дані з обстеження крові, але вони не принесли покращення результатів.

Переможці конкурсу у сфері передбачення термостабільності варіантів ферментів (білків, які діють як каталізатори хімічних реакцій живих організмів) «Novozymes Enzyme Stability Prediction» використовували ансамблі моделей. Зокрема автори 2-го місця – різні моделі, передусім – XGBoost, LightGBM, RandomForest, автори 3-го місця – XGBoost. Варто зауважити, що автор рішення, яке посіло 2-ге місце, вчинив оригінально. Він створив 2 різних ансамблі і передбачення одного відняв від передбачення іншого. Найбільш складним для нього було підібрати ваги для усіх моделей в ансамблі та їх параметри, але це дало гарний ефект. Автор рішення, яке посіло 1 місце, використовував різні нейронні мережі, але, цікаво, що його рішення не було найкращим, як він сам зазначає. Невдале балансування тестових даних у конкурсі призвело до сильних змін у рейтинговій таблиці і багато команд з найкращими рішеннями просто не вибрали їх як фінальні. Наприклад, цікавим є порівняльний пост автора рішення, яке посідало 1 місце на публічній частині Leaderboard, потім впало на майже 1000 місць на приватній (прихованій – фінальній) частині тестових даних, за якими й визначався переможець. Автор стверджує, що одне з його рішень є кращим за рішення переможця, але він його не вибрав, сподіваючись, що тестові дані збалансовані… марно сподівався. Його найкраще рішення, яке йому варто було б вибрати як фінальне, основано на моделі RandomForest. Його пост є цікавим в тому плані, що він у ньому аналізу і порівнює рішення переможців конкурсу та наводить свої цікаві ідеї і висновки.

Конкурс «Microsoft Malware Prediction», організований компанією Microsoft разом з ВНЗ та дослідницькими центрами, спрямований на розробку методів передбачення атаки вірусів на комп'ютери. Учасники отримують доступ до великого набору даних з метою розробки ефективних методів прогнозування випадків зараження комп'ютерів шкідливим програмним забезпеченням. Головна мета полягає в запобіганні збитків, які можуть бути заподіяні споживачам та підприємствам через шкідливе ПЗ. Учасники використовують розроблені методи, щоб передбачити можливі атаки вірусів та допомогти забезпечити безпеку понад одного мільярда комп'ютерів.

У конкурсі «IEEE-CIS Fraud Detection», учасники отримують можливість покращити системи виявлення шахрайства у фінансових транзакціях, використовуючи дані реальних транзакцій електронної комерції від компа-

нії Vesta. Завдяки новим методам машинного навчання та аналізу даних учасники можуть підвищити точність виявлення фальшивих транзакцій, що допоможе мільйонам користувачів уникнути фінансових втрат.

На рис. 4.38 наведена інфографіка інструментарію, згаданого у розділі 4 в цілому, у системі координат *S(I)*.

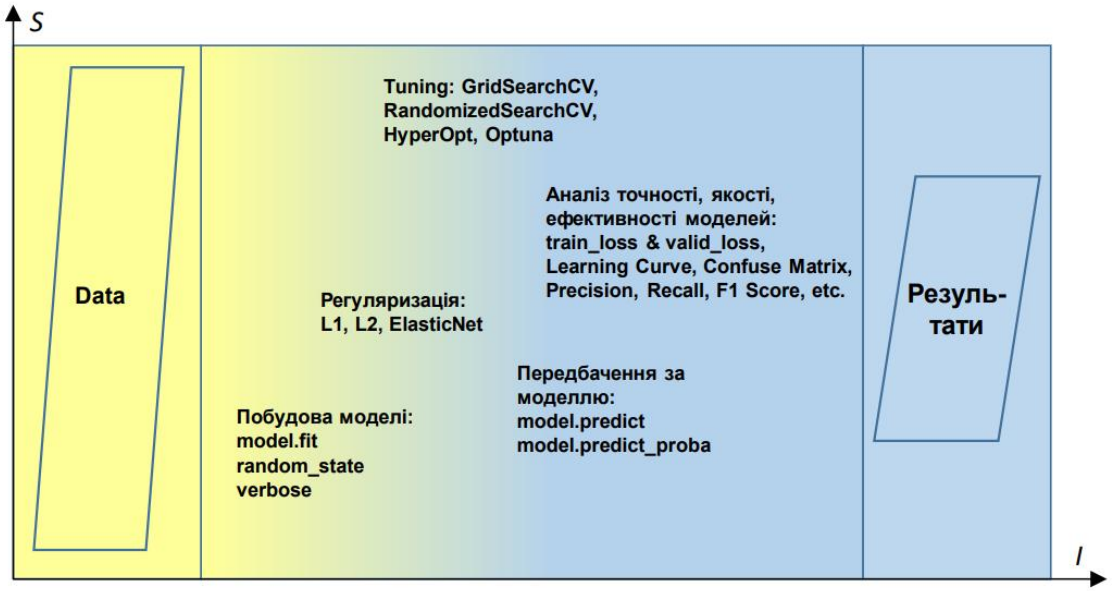


Рисунок 4.38 – Інфографіка тюнінгу моделей машинного навчання

Можливі теми практичних і лабораторних завдань

<u>Тема № 1</u>. Вибір методів та налаштування моделей машинного навчання для розв'язання задач аналізу та передбачення (або «**Побудова інтелектуальної моделі для передбачення даних про стан складної системи та аналіз вхідних даних за її допомогою**» або «**Вибір методів та ідентифікація моделей для розв'язання задач системного аналізу**»).

Метою заняття є вивчення інформаційних технологій і Python-бібліотек для налаштування моделей машинного навчання для розв'язання задач аналізу та передбачення й опанування практичних навичок застосування деяких із них на прикладі одного із датасетів Kaggle чи за даними, завантаженими через API.

План заняття:

1. Знайти датасет.

2. Описати точну постановку задачі і вказати яка це задача – класифікаційна чи регресійна.

3. Вибрати Python-бібліотеки, які будуть використані для побудови моделей машинного навчання та їх використання для передбачення значень, наприклад: Sklearn, Xgboost, Lightgbm тощо і зазначити для чого саме, для яких моделей із них.

4. Побудувати мінімум 3 різні моделі.

5. Зробити ансамбль із моделей чи виконати один з видів узагальнення їх рішень.

6. Навести таблицю (для класифікаційної задачі – ще й конфузійні матриці) з досягнутою точністю моделей та їх ансамблів та/або узагальнених рішень на навчальних та валідаційних даних, навести графік передбачених тестових даних. Зробити висновок про найкраще рішення та обґрунтувати це, на основі порівняння та аналізу її точності на навчальних та валідаційних даних.

Новачкам у цій сфері рекомендується взяти за основу заготовку ноутбука: AI-ML-DS Training. L2T: NH4 - Tree Regress models.

Варто послухати відео з коментарями:

- Моделі-класифікатори табличних даних на прикладі конкурсу по Титаніку - Курс AI-ML-DS Training
- Моделі-регресори табличних даних на прикладі моделювання якості води - Курс AI-ML-DS Training
- Моделі AI: передоброблення даних, тюнінг та оцінювання точності моделей - Курс AI-ML-DS Training
- Дерева рішень-регресори на прикладі моделювання якості води - Курс AI-ML-DS Training
- Приклад реальної задачі - моделювання якості води - Курс AI-ML-DS Training

Приклади ноутбуків:

- WQ SB river : EDA and Forecasting
- AI-ML-DS Training. L1T: Titanic - Decision Tree
- AI-ML-DS Training. L2T: NH4 - Tree Regress models
- AI-ML-DS Training. L4AT: Heart Disease prediction
- Heart Disease - Automatic AdvEDA & FE & 20 models
- Autoselection from 20 classifier models & L_curves
- Biomechanical features - 20 popular models
- Suspended substances prediction in river
- Merging FE & Prediction - xgb, lgb, logr, linr
- BOD prediction in river - 15 regression models

Тема № 2. Машинне навчання та застосування інтелектуальної моделі у змаганні Kaggle

Метою заняття є вивчення інформаційних технологій і Python-бібліотек для налаштування моделей машинного навчання для розв’язання задач аналізу та передбачення й опанування практичних навичок застосування деяких із них на прикладі одного із датасетів Kaggle чи за даними, завантаженими через API.

План заняття:

1. Знайти датасет.

2. Розібратись у постановці задачі та усіх її аспектах (цільова ознака, вид задачі, метрика, надані дані, чи можна використовувати зовнішні дані і з якою ліцензією тощо).

3. Сформувати команду. Організувати її роботу (варто скористатись порадами посібника [https://iq.vntu.edu.ua/repository/getfile.php/5171.pdf]).

4. Вивчити наявні рішення та поради у розділах «Code» і «Discussion».

5. Провести розвідувальний аналіз даних, у т.ч. FE-етап, та розробити власні гіпотези щодо рішень.

6. Побудувати моделі.

7. Подати серію рішень, щоразу їх удосконалюючи.

8. Після завершення конкурсу вивчити рішення переможців.

9. Написати статтю з оглядом задачі, її вирішення, власними гіпотезами. Зробити висновки. Що спрацювало, що – ні. Опублікувати свої вдалі і цікаві рішення. Зробити посилання у статті на ноутбуки і пости інших та на свої.

Контрольні питання

1) Які основні види моделей машинного навчання і їх переваги?
2) Що таке навчання моделі та гіперпараметрів? Як вони впливають на процес побудови моделі?
3) Що таке регуляризація, і яку роль вона відіграє у мінімізації ризику перенавчання?
4) В чому полягає лінійна регресія, і які її особливості?
5) Як працює логістична регресія, і в яких областях вона застосовується?
6) Що таке стохастичний градієнтний спуск, і як він використовується для навчання моделей?
7) Як працюють методи опорних векторів, і коли вони застосовуються?
8) Як працює метод k-найближчих сусідів, і які його особливості?
9) Що таке ансамблі моделей, і як вони допомагають у покращенні результатів прогнозування?
10) Як використовується навчання гіперпараметрів для покращення ефективності моделей машинного навчання, і як відбувається перевірка їх ефективності?

5.1 Основні поняття нейронних мереж (NN) та глибокого навчання (DL)

Нейромережа в машинному навчанні це – шари з нейронів і зв'язки між ними. *Нейрон* – це функція з багатьма входами і одним виходом, значення якого формується, в залежності від *функції активації* нейрону. Як тільки вихід цієї функції долає певний поріг, тоді на виході формується 1, інакше – 0 (більше детально про них буде нижче). *Зв'язки* – це канали, через які нейрони шлють один одному значення. Кожен зв'язок має свою *вагу* – параметр, на який помножується значення в каналі (рис. 5.1). Саме налаштування цих ваг (але не тільки їх) здійснюється під час налаштування (тюнінгу) нейронної мережі.

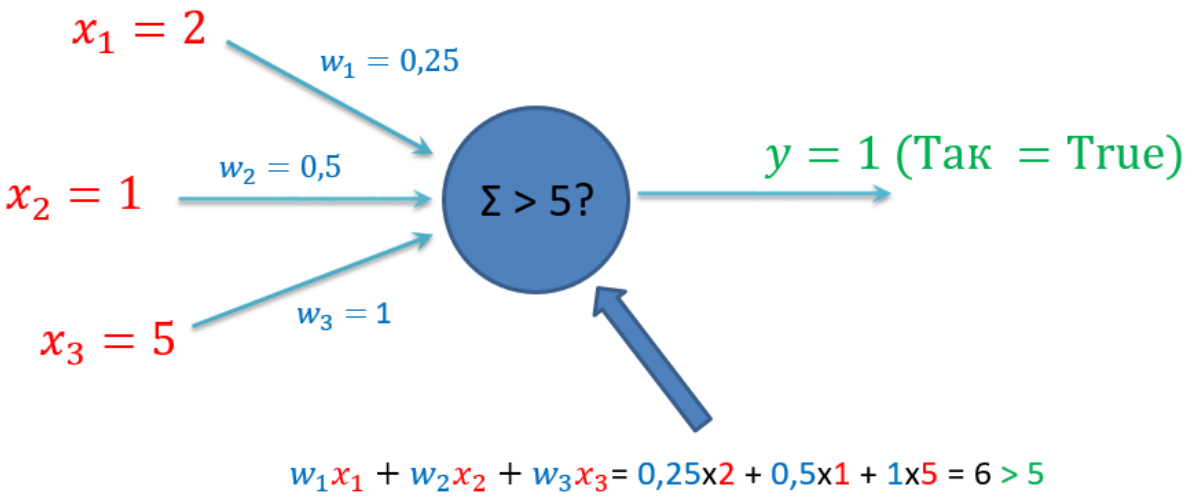


Рисунок 5.1 – Ілюстрація роботи одного нейрону нейронної мережі у машинному навчанні

З нейронів формують *шари*. У межах шару вони не пов’язані, але пов’язані з попередніми і наступним шарами (рис. 5.2).

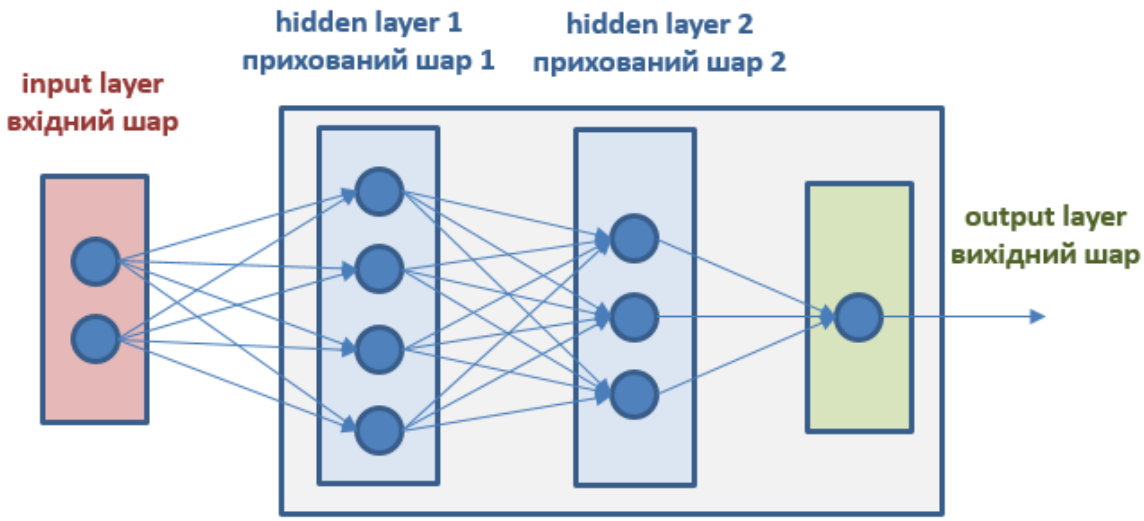


Рисунок 5.2 – Нейронна мережа з 4-ма шарами, у т.ч. з 2-ма прихованими

Вхідна матриця даних (як правило, це – 4-вимірна матриця, яку в машинному навчанні називають «*тензор*» – важливо не плутати цей термін з тензорами в математичній фізиці, вони – геть різні) надходить на перший шар. Дані поширюються зліва направо. Тому одна з найпоширеніших бібліотек для роботи з нейронними шарами називається TensorFlow – потік тензорів. Google придбав TensorFlow та інтегрував в бібліотеку Keras, тому тепер достатньо працювати тільки з останньою. Її основним конкурентом є PyTorch. Професіонали, як правило, використовують PyTorch, а для навчальних цілей краще опановувати Keras, тому ще це простіше і швидше.

Зазвичай, користувач «бачить» (може програмно подавати сигнали чи зчитувати їх) тільки входи і виходи, а значення у вузлах нейронної мережі від нього приховані, тому такі шари називаються *прихованими*. Якщо кількість прихованих шарів нейронної мережі більша за 1, то це – *глибоке* (іноді його ще називають «глибинним») *навчання* (англ. «Deep Learning» – DL).

Функція активації, або *передавальна функція* (англ. «activation function» чи «excitation function», «squashing function», «transfer function») штучного нейрона це – залежність вихідного сигналу штучного нейрона від вхідного. Зазвичай ця функція відображає дійсні числа на інтервал [0, 1] чи [-1, 1]. Найбільш популярні види функцій активації нейронних мереж [30, 31]:

1. Усічена одиниця (ReLU) (у Вікіпедії: «випрямлена одиниця», що змістовно – менш коректно) (рис. 5.3):

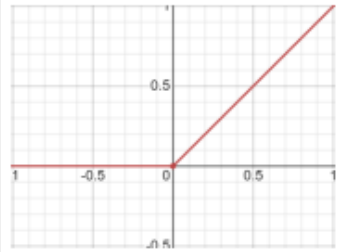
keras.activations.relu(x, alpha=0.0, max_value=None, threshold=0.0)				
Випрямлена лінійна		$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$	$[0, \infty)$

Рисунок 5.3 – Функція активації ReLU «усічена одиниця»

Має ряд підвидів:

- Leaky ReLU & Parametric ReLU (PReLU) (рис. 5.4):

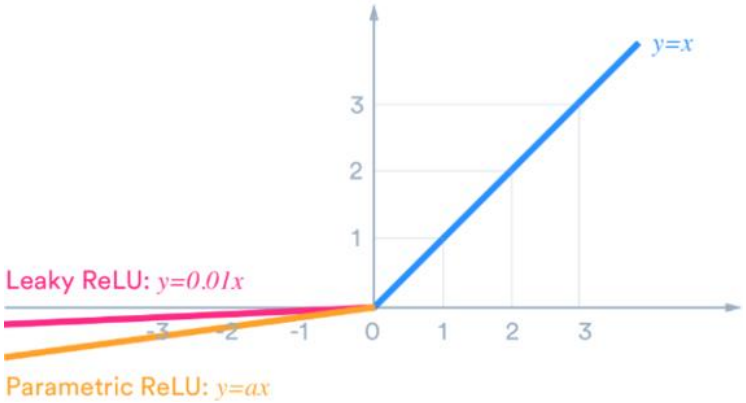


Рисунок 5.4 – Функція активації PReLU «усічена одиниця за параметром»

- ReLU-6 (рис. 5.5):

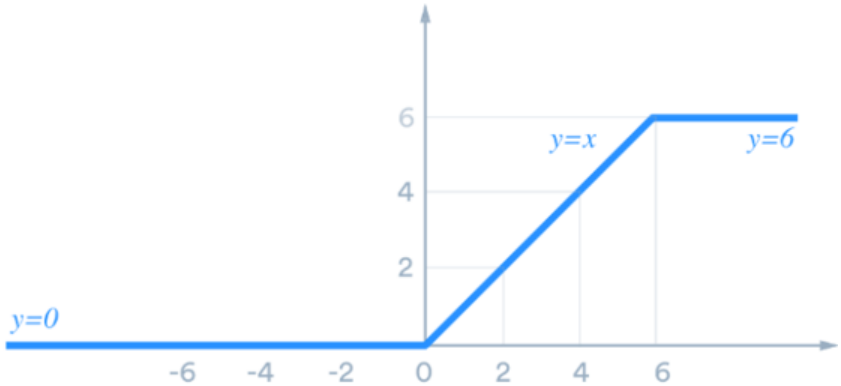


Рисунок 5.5 – Функція активації ReLU-6
«усічена одиниця з насиченням у 6 одиниць»

- Exponential Linear (ELU, SELU) (рис. 5.6):

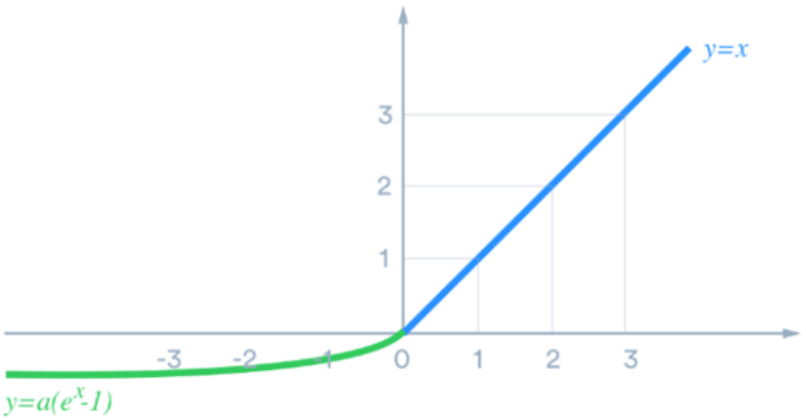


Рисунок 5.6 – Експоненціальні функції активації ELU, SELU

2. Сігмоїда (логістична функція) «sigmoid» (рис. 5.6):

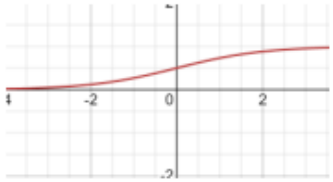
keras.activations.sigmoid(x)				
Логістична або Сігмоїдна		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$	(0,1)

Рисунок 5.7 – Функція активації Сігмоїда (логістична функція) «sigmoid»

3. Гіперболічний тангенс «tanh» (рис. 5.7):

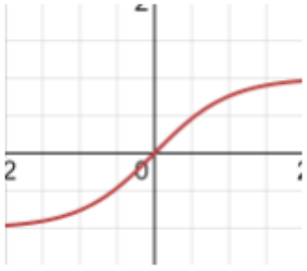
keras.activations.tanh(x)				
Гіперболічний тангенс		$f(x) = tanh(x) = \frac{(e^x - e^{-x})}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	(−1,1)

Рисунок 5.8 – Функція активації Гіперболічний тангенс «tanh»

Історично, першою функцією активації була сігмоїда, але, через багато помножень на дуже малі значення, сигнал часто «губився» між шарами і занулювався. Це було однією з причин повільного розвитку нейронних мереж. Але згодом, було винайдено фунцію ReLU, яка мала потрібні нелінійні властивості, легко автоматизувалась та була позбавлена такого недоліку. Наразі, в усіх прихованих шарах найбільш популярною є саме ReLU, а у вихідному шарі, особливо для задач класифікації використовують сигмоїду

(якщо усі значення є позитивними) чи – гіперболічний тангенс (якщо є від'ємні значення).

Реалізація простої нейронної мережі з рис. 5.2 з використанням бібліотеки Keras (TF) виглядає доволі лаконічно (рис. 5.8).

```
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam

model = Sequential()
model.add(Dense(units=4, activation='relu', input_dim=len(train.columns)))
model.add(Dense(units=3, activation='relu'))
model.add(Dense(units=1, activation='sigmoid'))

optimizer = Adam(lr=0.001)
model.compile(loss='mse', optimizer=optimizer, metrics=['mse'])
```

Рисунок 5.9 – Реалізація нейромережі з рис. 5.2 у Keras (TF)

Аналогічний код в PyTorch виглядатиме значно довшим (рис. 5.9).

```python
import torch
import torch.nn as nn
import torch.optim as optim

class NeuralNetwork(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2, output_size):
        super(NeuralNetwork, self).__init__()
        self.layer1 = nn.Linear(input_size, hidden_size1)
        self.relu1 = nn.ReLU()
        self.layer2 = nn.Linear(hidden_size1, hidden_size2)
        self.relu2 = nn.ReLU()
        self.output_layer = nn.Linear(hidden_size2, output_size)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.layer1(x)
        x = self.relu1(x)
        x = self.layer2(x)
        x = self.relu2(x)
        x = self.output_layer(x)
        x = self.sigmoid(x)
        return x

# Set network parameters
input_size = len(train.columns)
hidden_size1 = 4
hidden_size2 = 3
output_size = 1

# Create an instance of the NeuralNetwork class
model = NeuralNetwork(input_size, hidden_size1, hidden_size2, output_size)

# Determine the losses and the optimizer
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Рисунок 5.10 – Реалізація нейромережі з рис. 5.2 у PyTorch

5.2 Навчання нейронної мережі та аналіз її точності

Як було зазначено вище, навчання нейронних мереж здійснюється по-етапно. Кожен такий етап називається епоха (рис. 5.10).

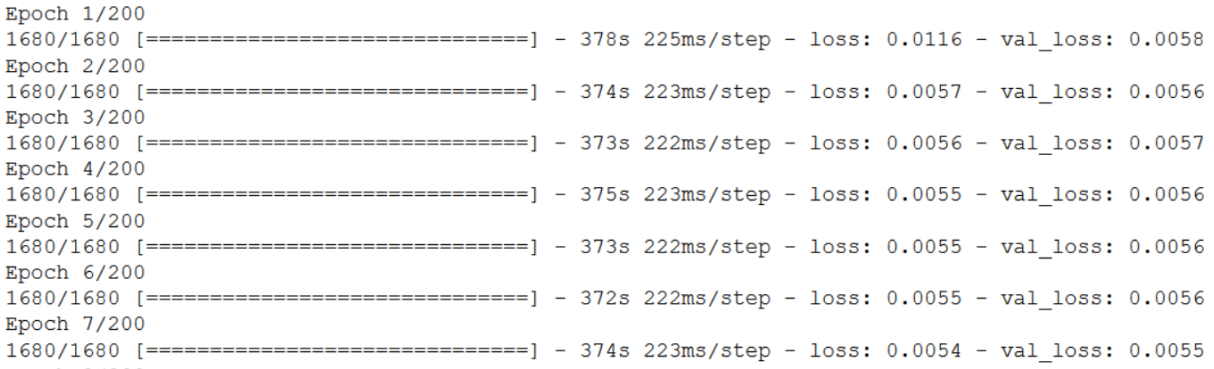


Рисунок 5.11 – Приклад навчання нейронної мережі

Рис. 5.10 ілюструє приклад навчання нейронної мережі (з реального прикладу одного із співавторів посібника): перші 7 епох з 200 максимальних (як правило, навчання завершується, коли точність стане задовільною, або, коли кількість епох досягне заданої максимальної кількості). Число 1680 означає кількість батчів (партій), що означає, що здійснюється оброблення доволі великого датасету, розбитого на ці батчі. На першій епосі оброблення кожного батча займало в середньому 225 мс, а вся епоха тривала 378 с. Головним критерієм навчання, як було зазначено вище, є точність на валідаційному датасеті – val_loss. Користувач сам задає метрику. Теоретично, метрика для точності на тренувальному датасеті (loss) і для val_loss може відрізнятись, особливо, коли модель тренується для конкурсу з якоюсь дуже специфічною метрикою, тоді вона вказується тільки для val_loss. На рис. 5.10 видно, що алгоритм намагається зменшити val_loss, отже метрика передбачає «чим менше, тим краще», наприклад, це – MSE чи RMSE для регресійної задачі. В деяких епохах val_loss є однаковим, але це може бути і не так – скоріше, варто у параметрах вимагати виводити це значення з більшою кількістю знаків після коми, оскільки різниця може бути у 5 чи 6 знаку. Також, видно, що val_loss і loss – корелюють, тобто зменшуються синхронно, що є доброю ознакою, тобто модель навчається на тренувальному датасеті гарно, що підтверджується, щоразу, на валідаційному датасеті. Однак, так буде рідко. Частіше, val_loss перестає змінюватись, а loss починається збільшуватись або перестає змінюватись, тоді застосовують змінний крок learning_rate (скорочено: lr). Код з рис. 5.8 зі змінним кроком наведено на рис. 5.11.

```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers

def build_nn():
    model = Sequential()
    model.add(Dense(units=4, activation='relu', input_shape=(len(train.columns),)))
    model.add(Dense(units=3, activation='relu'))
    model.add(Dense(units=1, activation='sigmoid'))
    model.compile(loss='mse', optimizer='adam', metrics=['mse'])
    learning_rate_reduction = ReduceLROnPlateau(monitor='val_mse',
                                        patience=3,
                                        verbose=1,
                                        factor=0.5,
                                        min_lr=0.0001)
    return model
nn_model = build_nn()
```

Рисунок 5.12 – Реалізація нейромережі з рис. 5.2 у Keras зі змінним кроком

Рис. 5.11 ілюструє спеціальну команду, яка відслідковує зміну val_loss і, якщо протягом patience=3 кроків його значення не зазнає змін, тоді значення lr помножується на factor=0.5, тобто зменшується удвічі. І так буде робитись, аж поки значення lr не зменшиться до min_lr=0.0001, а тоді перестане зменшуватись. На рис. 5.12 наведено приклад кривої навчання зі змінним кроком lr.

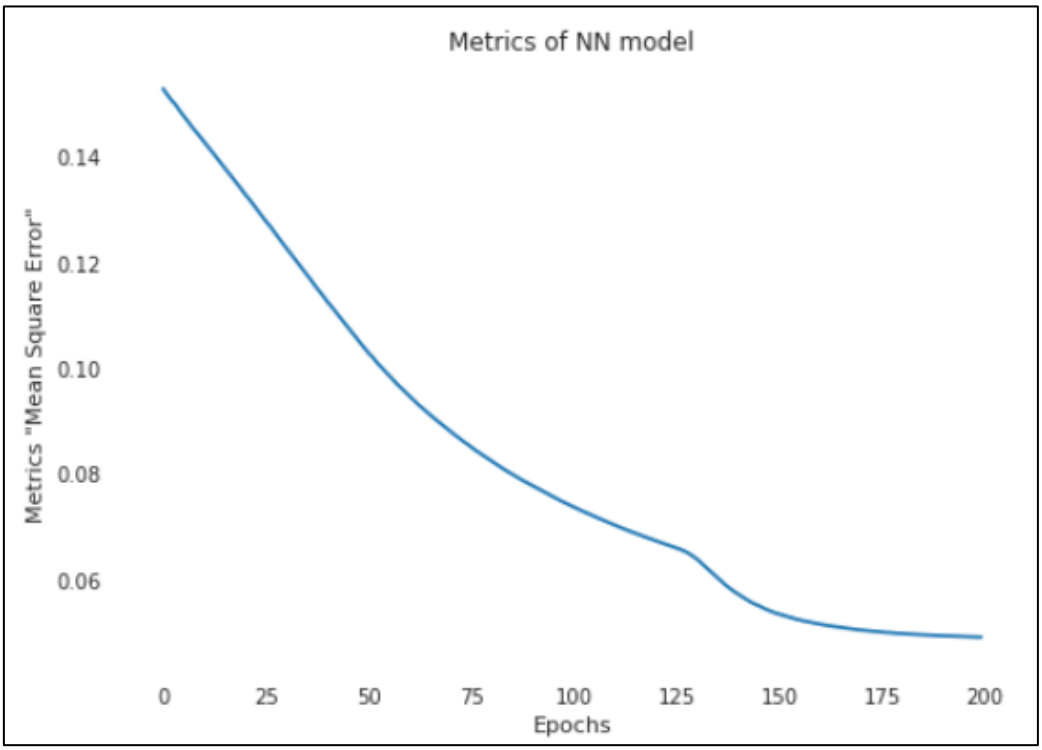


Рисунок 5.13 – Навчання нейронної мережі з архітектурою з рис. 5.2 зі змінним кроком lr

Для збільшення узагальнюючої здатності нейромережі та зменшення ризику перенавчання, між її шарами може застосовуватись операція Dropout з параметром від 0,1 до 0,5. Це означає, що від 10% до 50% нейронів випадково відключаються під час тренування ваг моделі (рис. 5.13). Насправді, 0,4-0,5 рідко використовуються, оскільки можуть призвести до недонавчання моделі.

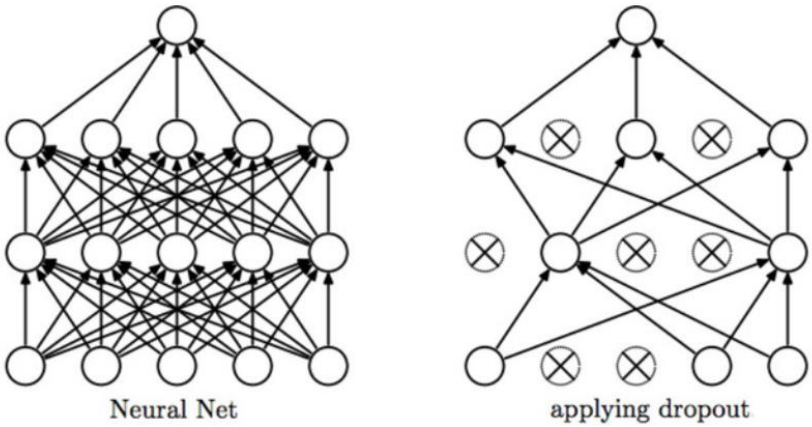


Рисунок 5.14 – Ілюстрація дії операції Dropout з відключення частини нейронів під час навчання нейронної мережі з ноутбука

На рис. 5.15 наведено фрагмент додавання Dropout до коду з рис. 5.11.

```
model = Sequential()
model.add(Dense(units=4, activation='relu', input_shape=(len(train.columns),)))
model.add(Dropout(0.2))
model.add(Dense(units=3, activation='relu'))
```

На рис. 5.15 Додавання Dropout до коду з рис. 5.11 (див. ноутбук)

Крива навчання нейромережі з Dropout з рис. 5.14 наведено на рис. 5.15.

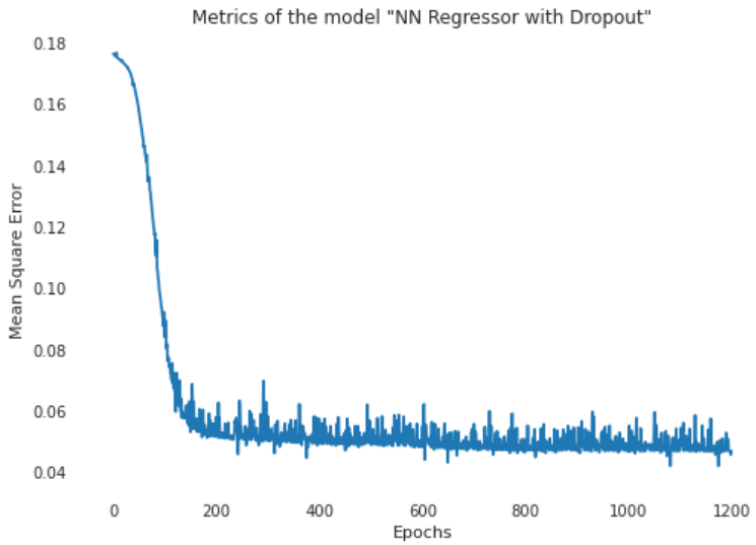


Рисунок 5.16 – Навчання нейронної мережі з Dropout у коді з рис. 5.14, який додано до коду з рис. 5.12 (див. ноутбук)

Як видно на рис. 5.15, Dropout вносить суттєве зашумлення у процес навчання. Можливо, варто було його додати не на початку, а в кінці процесу навчання. Хоча, часто, його додають після кожного шару. Але він є більш ефективним за великої кількості даних та у нейронних мережах складної архітектури, які мають високий ризик перенавчання.

Як правило, в серйозних задачах нейронні мережі навчаються багато годин і, навіть – діб. При цьому, може використовуватись паралелізація з використанням платних сервісів на кшталт платного Google Colab чи Amazon. Тому такі навчання називають «дорогими». А тому важливо мати можливість постійно дивитись на криву з рис. 5.12 чи 5.15 і весь час аналізувати чи навчання триває успішно, тобто чи дійсно val_loss поступово зменшується. У разі, якщо він доволі тривало перестає зменшуватись, є сенс зупинити навчання. Що саме вважати «доволі тривалим» підкаже досвід датасайнтиста. Іноді краще почекати, оскільки повторювати цикл навчання теж дорого.

Як правило, у коді реалізовується механізм автоматичного збереження усіх параметрів нейронної мережі у форматі pkl у певну папку, коли новий val_loss є меншим за val_loss на попередній епосі. Іноді так зберігають тільки кожен 10-й val_loss. Є й інші способи. А тоді, після аварійного переривання навчання, датасайнтист просто бере останній pkl-файл і це і є відповідь на задачу, тобто – параметри оптимальної моделі.

Але як же дивитись криву навчання. Технологічно, запуск коду відбувається у ноутбуці, який виводить результати вже після завершення. Для усунення цієї проблеми існує спеціальний сервіс TensorBoard. В код додається callbacks (https://keras.io/callbacks/) і задається що саме слід відслідковувати. Інформація щоразу зчитується і візуалізується в режимі «он-лайн» – її можна переглядати у браузері (як правило, через канал № 6006) (рис. 5.16).

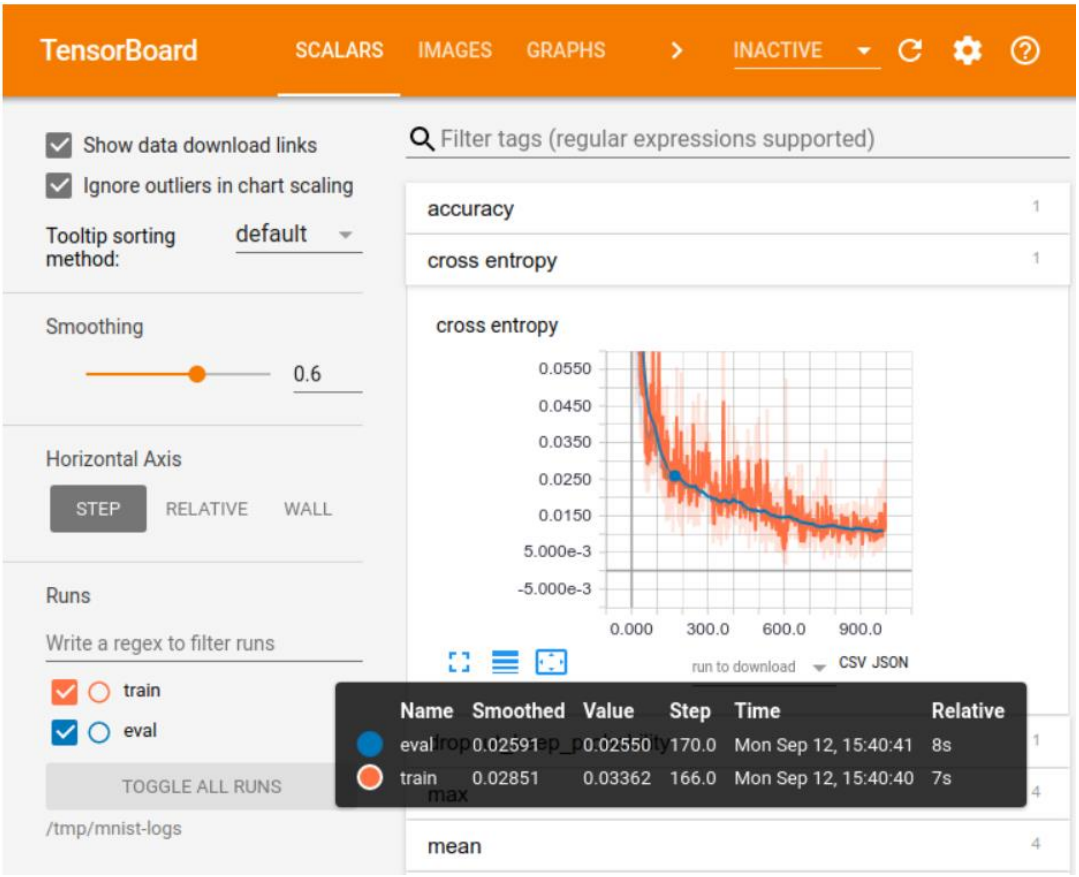


Рисунок 5.17 – Приклад візуалізації поточних результатів навчання нейронної мережі у браузері в режимі «онлайн»

Іноді доволі ефективною є використання моделі багатошарового перцептрона (нейронної мережі) з бібліотеки Sklearn: MLP («Multi-layer Perceptron»):

name_package = neural_network
name_model(model_params):
- класифікатор:
  - MLPClassifier(*hidden_layer_sizes=(100,), activation='relu', *, solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False, validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000*);

- регресор:
  - MLPRegressor(*hidden_layer_sizes=(100,), activation='relu', *, solver='adam', alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001, power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False, warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False,*

validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10, max_fun=15000);

- LinearSVR(*, *epsilon=0.0, tol=0.0001, C=1.0, loss='epsilon_insensitive', fit_intercept=True, intercept_scaling=1.0, dual='warn', verbose=0, random_state=None, max_iter=1000*).

Головними параметрами моделі MLP, які варіюють для підвищення точності, окрім типових для моделей Sklearn та описаних у розд. 4, є кількість прихованих шарів *hidden_layer_sizes* (можна задати у вигляді списку з кількістю нейронів у кожному шарі окремо) та тип функції активації *activation* («identity», «logistic», «tanh», «relu»).

Згадуваний у цьому підрозділі авторський ноутбук ілюструє застосування MLP для цієї ж задачі. На рис. 5.17 наведено результати навчання 4-х описаних у цьому підрозділі моделей.

model	train_score	train_mse	valid_score	valid_mse
MLP Regressor	0.70	0.057061	0.71	0.070000
NN Regressor with Dropout	0.73	0.050000	0.67	0.076795
NN Regressor with changed lr	0.72	0.048084	0.66	0.080000
NN Regressor	0.67	0.057061	0.66	0.080000

Рис. 5.18 Результат навчання 4-х моделей у Kaggle ноутбуці

Як видно на рис. 5.17, найкращою на валідаційному датасеті є модель Sklearn MLPRegressor, хоча нейромережа, побудова на основі бібліотеки TensorFlow з Keras, яка використовує Dropout, досягла вищої точності на тренувальних даних. Зауважимо, що ноутбук не є репрезентативним, оскільки моделі будуються на дуже малій як для регресійної задачі вибірці (сотні даних). Для гарних висновків і високої точності нейронних мереж потрібні мільйони даних!

На рис. 5.18 наведена інфографіка інструментарію, згаданого у підрозділах 5.1 та 5.2 у нотації рис. 1.2.

Рисунок 5.19 – Інфографіка побудови і навчання нейромережевих моделей

5.3 Сучасні архітектури нейронних мереж та їх ансамблів

У 2016 році Інститут Айзека Азимова (США) опублікував інфографіку основних архітектур сучасних на той момент нейронних мереж (рис. 5.20).

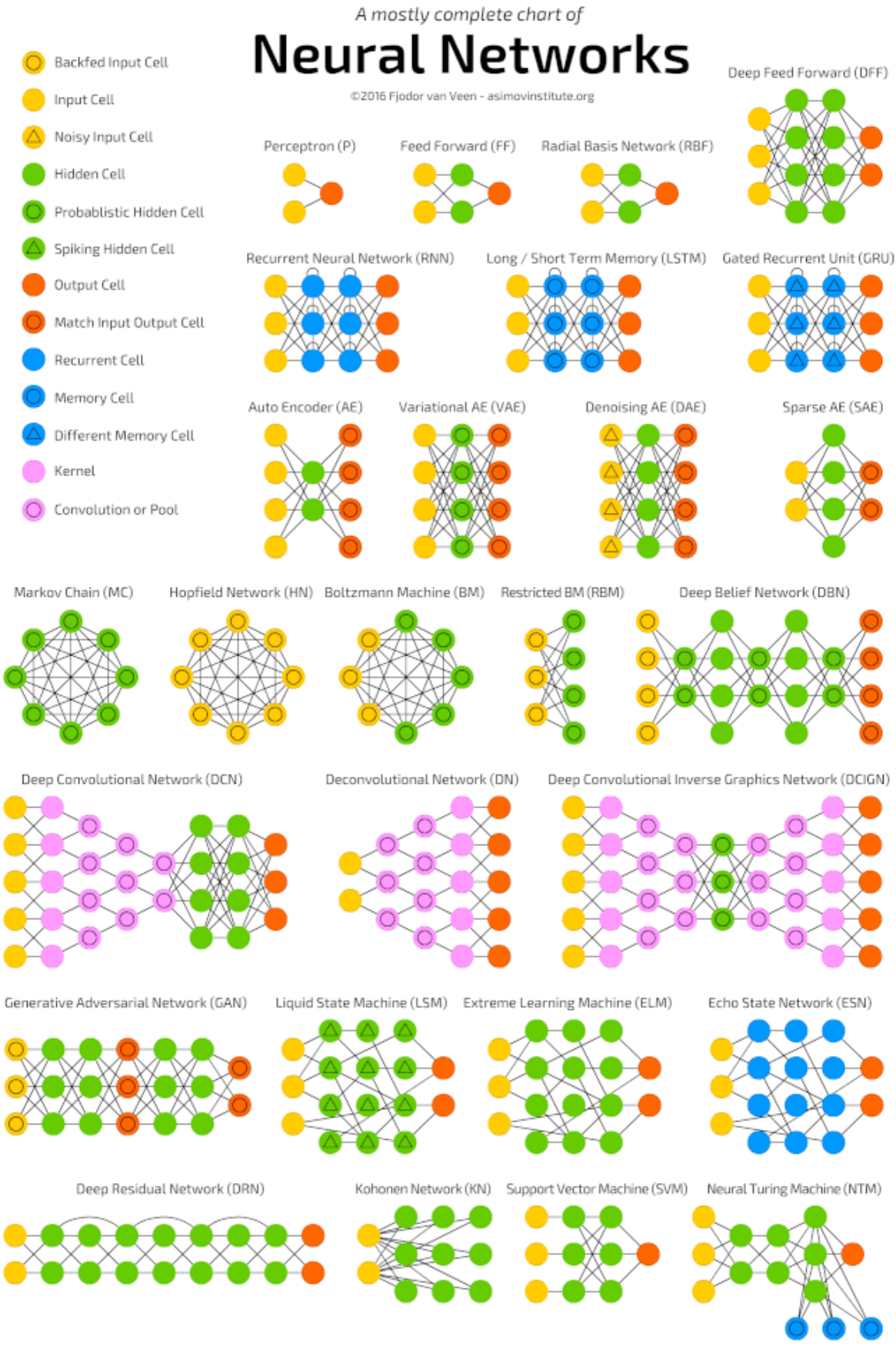


Рисунок 5.21 – Архітектури нейронних мереж станом на 2016 р.

Звичайні мережі Кохонена - це тип нейронних мереж, які використовуються для навчання без учителя і використовуються для виявлення шаблонів або кластеризації даних. Суть даної архітектури полягає у тому, що мережа вчиться відтворювати структуру вхідних даних у вигляді карт, які називаються картами ознак або картами Кохонена. Кожен нейрон у вихідному шарі мережі відповідає певному центру кластера в просторі вхідних даних, а ваги нейронів налаштовуються під час навчання таким чином, щоб найбільш важливі особливості даних були відображені у вигляді цих карт.

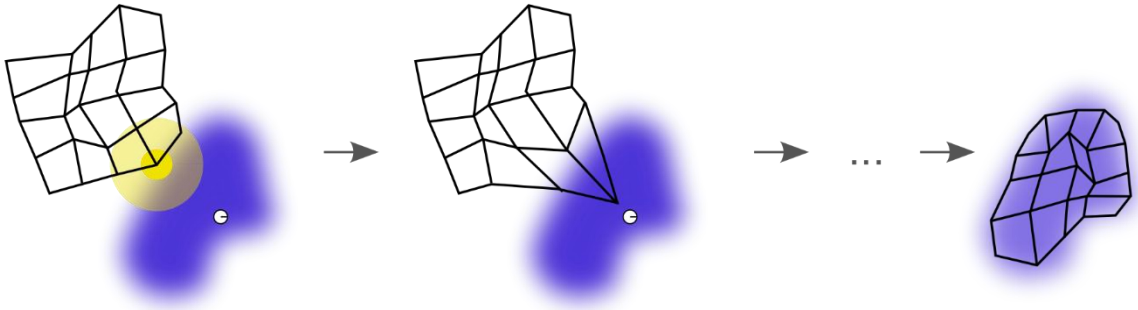


Рисунок 5.22 – Процес навчання мережі Кохонена

Ілюстрація навчання самоорганізаційної карти. Синя область — це розподіл тренувальних даних, а маленький білий диск — поточна навчальна вибірка, отримана з цих даних. Спочатку (ліворуч) вузли карти довільно розташовані в просторі даних. Вибирається вузол, найближчий до тренувального вузла (виділений жовтим кольором), та переміщується у напрямку тренувальних даних, оскільки є (меншою мірою) її сусідами на сітці. Після багатьох ітерацій сітка має тенденцію наближатися до розподілу даних (праворуч).

Машини Больцмана – це тип штучних нейронних мереж, які використовуються для навчання без учителя та для моделювання ймовірнісних розподілів. Суть даної архітектури полягає у використанні випадкового методу для знаходження глобального мінімуму або максимуму функції. У машин Больцмана нейрони організовані у дві шари: видимий шар і прихований шар. Вони взаємодіють між собою через зважені зв'язки, і процес навчання полягає у зміні цих зв'язків таким чином, щоб максимізувати ймовірність спостережуваних даних. Ця архітектура часто використовується для розв'язання завдань у галузі рекомендаційних систем, розпізнавання образів та інших задач машинного навчання.

RBF (Radial Basis Function) – це тип штучної нейронної мережі, яка використовується для наближення складних функцій. Вона складається зі шару вхідних нейронів, одного або декількох шарів прихованих нейронів з радіальною базисною функцією та шару вихідних нейронів. Суть даної архітектури полягає в тому, що прихований шар мережі використовує радіальні базисні функції для перетворення вхідних даних у просторовий проміжок, у якому можливо ефективно проводити класифікацію або регресію.

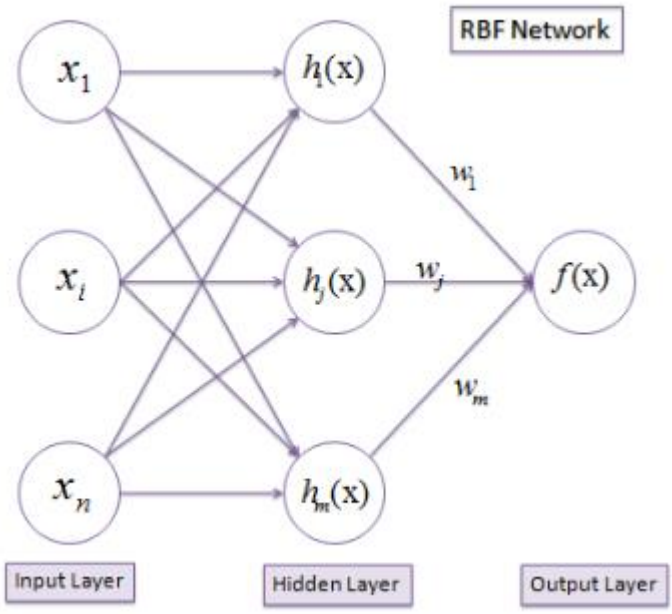


Рисунок 5.23 – Архітектура RBF

RNN (Recurrent Neural Network) – це тип нейронних мереж, які мають здатність враховувати інформацію з попередніх кроків в часі. Основна їх особливість - це здатність працювати з послідовними даними, такими як текст, звук, часові ряди тощо.

Суть архітектури RNN полягає в наявності зв'язків, які утворюють цикли між нейронами, що дозволяє передавати інформацію з одного кроку в часі до наступного. Кожен часовий крок рекурентної нейронної мережі обробляє вхідні дані та виводить вихід, який стає вхідним сигналом для наступного кроку. Це дозволяє моделі враховувати контекст і залежності між даними в часі.

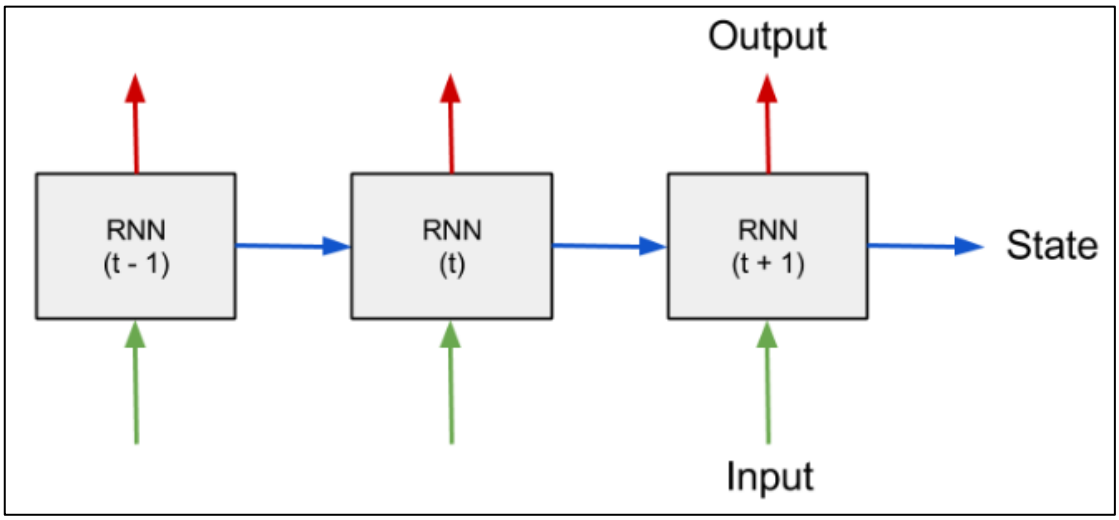


Рисунок 5.24 – Архітектура RNN

Long Short-Term Memory (LSTM) – це спеціальний тип рекурентних нейронних мереж, розроблений для розв'язання проблеми зникнення або вибування градієнту під час тренування, яка часто виникає у звичайних RNN.

Суть архітектури LSTM полягає у використанні спеціальних блоків пам'яті, які здатні зберігати інформацію на тривалий час та контролювати потік інформації через мережу. Основні складові LSTM включають:

1. Клітинний стан (Cell State): Це – основний канал для передачі інформації уздовж мережі. Клітинний стан може зберігати інформацію протягом багатьох часових кроків та контролюється за допомогою воріт (гейтів), що дозволяє відфільтровувати та оновлювати інформацію.
2. Ворота (Gates): LSTM використовує три типи воріт - ворота забування (Forget Gate), ворота входу (Input Gate) та ворота виходу (Output Gate). Кожне з цих воріт має власну функцію активації та допомагає контролювати потік інформації в мережі, регулюючи, яка частина інформації буде передана далі.
3. Пам'ять (Memory): LSTM зберігає та оновлює інформацію в клітинному стані за допомогою воріт, що дозволяє зберігати довгострокові залежності та уникнути зникнення або вибування градієнту.

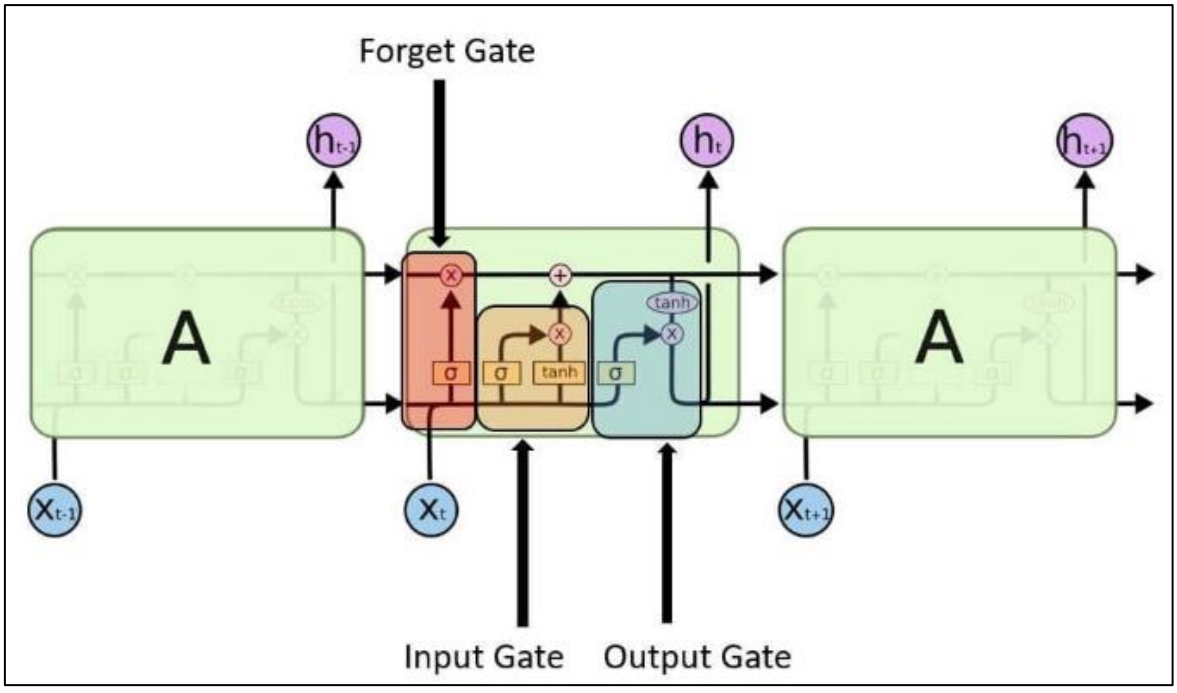


Рисунок 5.25 – Структура LSTM

Gated Recurrent Unit (GRU) - це інший варіант архітектури рекурентних нейронних мереж, схожа на LSTM, але з меншою складністю. Суть GRU полягає в використанні воріт (гейтів), які дозволяють мережі регулювати потік інформації, але вона має меншу кількість параметрів, що робить її більш ефективною для тренування.

Основні складові GRU включають:

1. Оновлення (Update Gate): Відповідає за визначення того, наскільки поточна інформація повинна бути оновлена.
2. Ворота забування (Reset Gate): Відповідає за вирішення, яку інформацію потрібно забути з минулого кроку часу.
3. Схований стан (Hidden State): Виходить із GRU блоку та містить інформацію, яка буде передана наступному кроку часу або в іншу частину мережі.
4. Оновлений стан (Updated State): Представляє оновлений вміст блоку GRU, що враховує інформацію про ворота оновлення та схований стан.

Про автоенкодери та згорткові нейронні мережі більш детально буде написано у наступних розділах.

На рис. 5.26 наведена інфографіка побудови нейромережевих моделей, згаданих у цьому підрозділі у нотації рис. 1.2.

Рисунок 5.26 – Інфографіка побудови складних нейромережевих моделей

5.4 Машинне навчання з підкріпленням (Reinforcement Learning)

Машинне навчання з підкріпленням (Reinforcement Learning, RL) — це галузь машинного навчання, яка зосереджена на вивченні того, як агент може взаємодіяти з невизначеним середовищем для досягнення максимальної винагороди. RL використовується для вирішення проблем прийняття рішень та навчання агентів оптимальному поведінці.

Використовуватися для вирішення широкого кола задач, включаючи:

- Ігри, такі як шахи, Go та Dota 2 (Google DeepMind);
- Робототехніка, роботи збирання та пакування;
- Автопілот Tesla;
- Фінанси, автоматизація торгівлі на ринку (трейдинг) (Alphabet).

Задача машинного навчання з підкріпленням формалізується через марківські процеси прийняття рішень (MDP), в якому агент приймає дії в певних станах, отримує нагороди від середовища та старається максимізувати кумулятивну нагороду. Метою є знайти оптимальну стратегію, яка вказує, які дії потрібно виконати в кожному стані.

До основних понять RL відносять: агента, середовище, винагороду та вартість кроку (рис. 5.27).

• Агент — у контексті навчання з підкріпленням (RL) представляє собою суб'єкт, який приймає рішення та взаємодіє із зовнішнім середовищем. Це може бути програмний агент, робот, або будь-який інший об'єкт, здатний взаємодіяти з оточуючим його світом. Агент має власні цілі та завдання, і його мета - максимізувати отриману винагороду.

•Середовище в RL - це контекст, у якому діє агент. Це може бути фізичне середовище, віртуальний світ, база даних чи будь-яке інше оточення, з яким агент взаємодіє. Середовище може бути стохастичним та змінюватися від часу до часу. Агент взаємодіє із середовищем, приймаючи дії та отримуючи відповідь у вигляді винагороди або покарання, що дозволяє йому вдосконалювати свою стратегію для досягнення своїх цілей.

•Винагорода у контексті навчання з підкріпленням (RL) є числовим сигналом, який агент отримує від середовища в результаті виконаних дій. Вона служить орієнтиром для агента, допомагаючи йому оцінити, наскільки ефективні його рішення в досягненні поставлених цілей. Винагорода може бути позитивною, якщо агент вчинив щось, що допомагає в його завданні, або негативною, якщо його дії призвели до невдачі. Мета агента - максимізувати загальну винагороду протягом взаємодії з середовищем.

•Вартість в контексті навчання з підкріпленням включає в себе не лише отриману винагороду, а й можливі витрати, пов'язані з прийняттям рішень та взаємодією агента з середовищем. Це може охоплювати енергетичні витрати, час, ресурси та інші фактори, які впливають на агента. Максимізація корисної дії агента повинна враховувати не лише величину отриманої винагороди, а й вартість, яка пов'язана з вибором певних стратегій та дій.

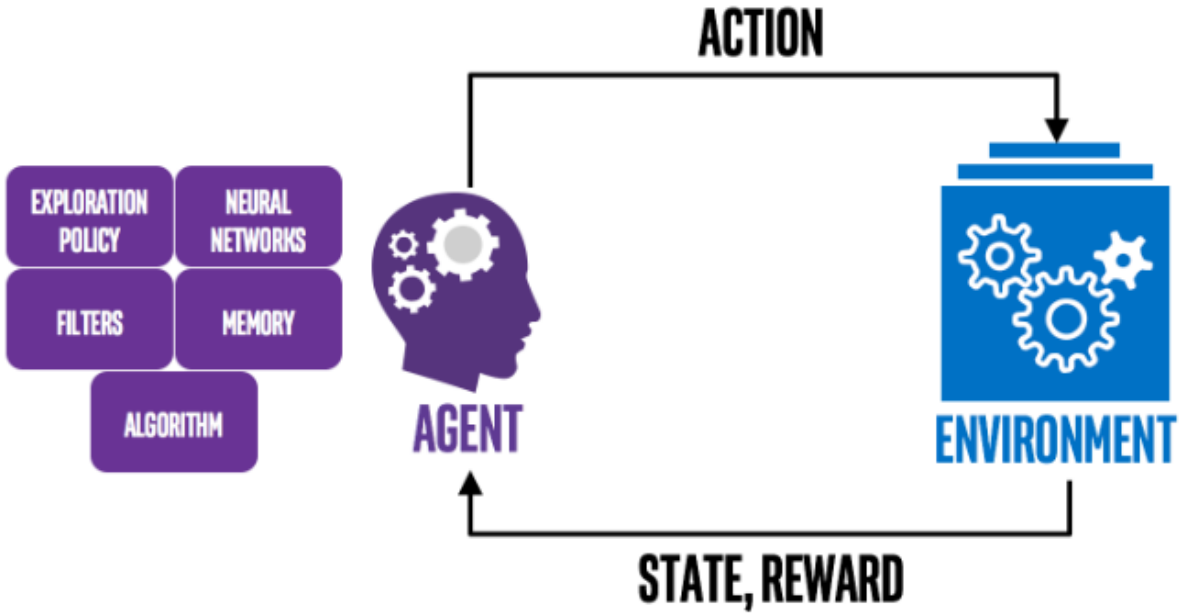


Рисунок 5.27 – Інфографіка побудови і використання моделей машинного навчання з підкріпленням

Існує більше 10 різновидів архітектур навчання з підкріпленням. Ось декілька основних архітектур:

1. Q-Learning, Deep Q Network (DQN) – це метод, оснований на ідентифікації RL-політики у вигляді глибокої нейромережі з використанням принципів методу звичайного Q-навчання. Оскільки агент Q-навчання не може оцінити ті стани, які не бачив, двовимірну сітку виграшів для кожної комбінації станів та дій замінили на нейронну мережу. Тобто метод DQN використовує нейронну мережу для оцінювання значень Q-функції. На вхід мережі подаються дані, а на виході отримуємо значення Q для кожної можливої дії. Основним недоліком такого методу реалізації навчання з підкріпленням є те, що дискретний простір дій, навіть за невеликого розміру простору дій, збільшується експоненціально, а тому надзвичайно складно отримати збіжність до оптимального рішення за обмежений час, що зображено на рисунку 5.28.

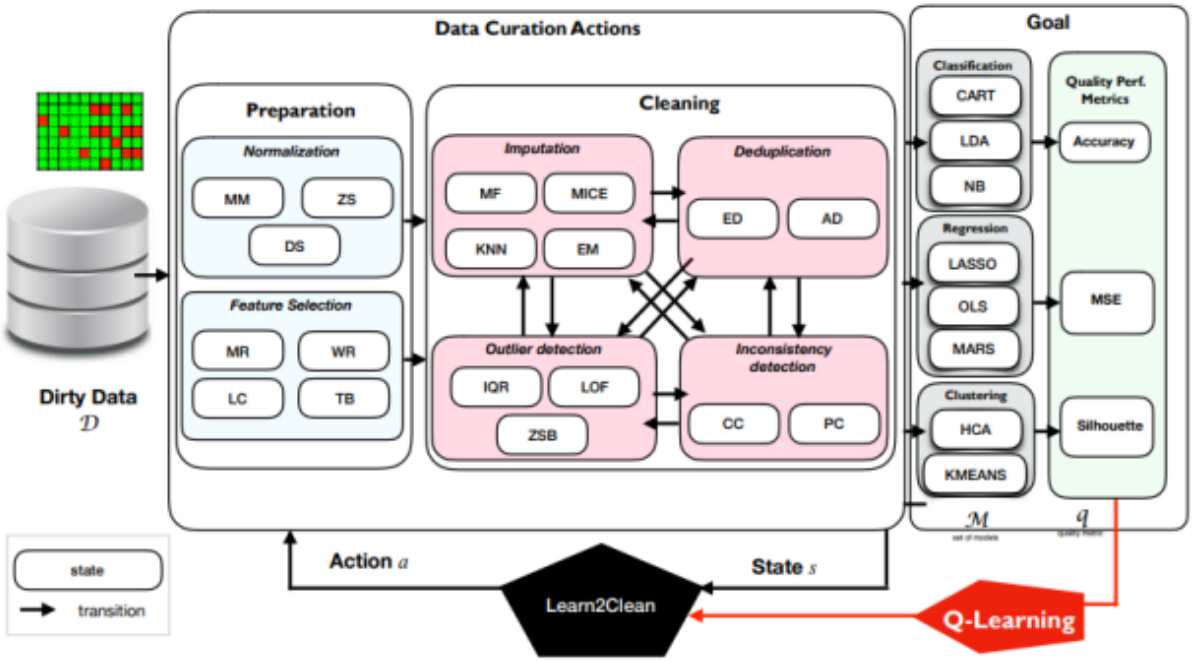


Рисунок 5.28 – Алгоритм роботи методу з використанням Q-навчання

2.Advantage Actor-Critic (A2C) – це метод «актор-критик», який має гібридну архітектуру, що поєднує методи, засновані і на діях, і на зміні політики, зокрема, що зображено на рисунку 5.29:

– «актор» контролює поведінку агента (метод на основі політики);

– «критик» визначає наскільки була ефективною дія чи операція (метод на основі дій).

Під час навчання «актор» вивчає найкращу дію, використовуючи відгуки «критиків» (замість використання винагороди безпосередньо). У той же час, «критик» вивчає функцію цінності з винагород, щоб він міг належним чином критикувати «актора». Загалом, ці агенти можуть обробляти як дискретні, так і безперервні простори дій. Як правило, ідея методу полягає у накопиченні вибраних траєкторій (послідовностей, конвеєрів, комбінацій)

можливо оптимальних дій і періодичному їх аналізі з метою корегування політики з метою покращення правил вибору цих оптимальних дій (рис 5.9).

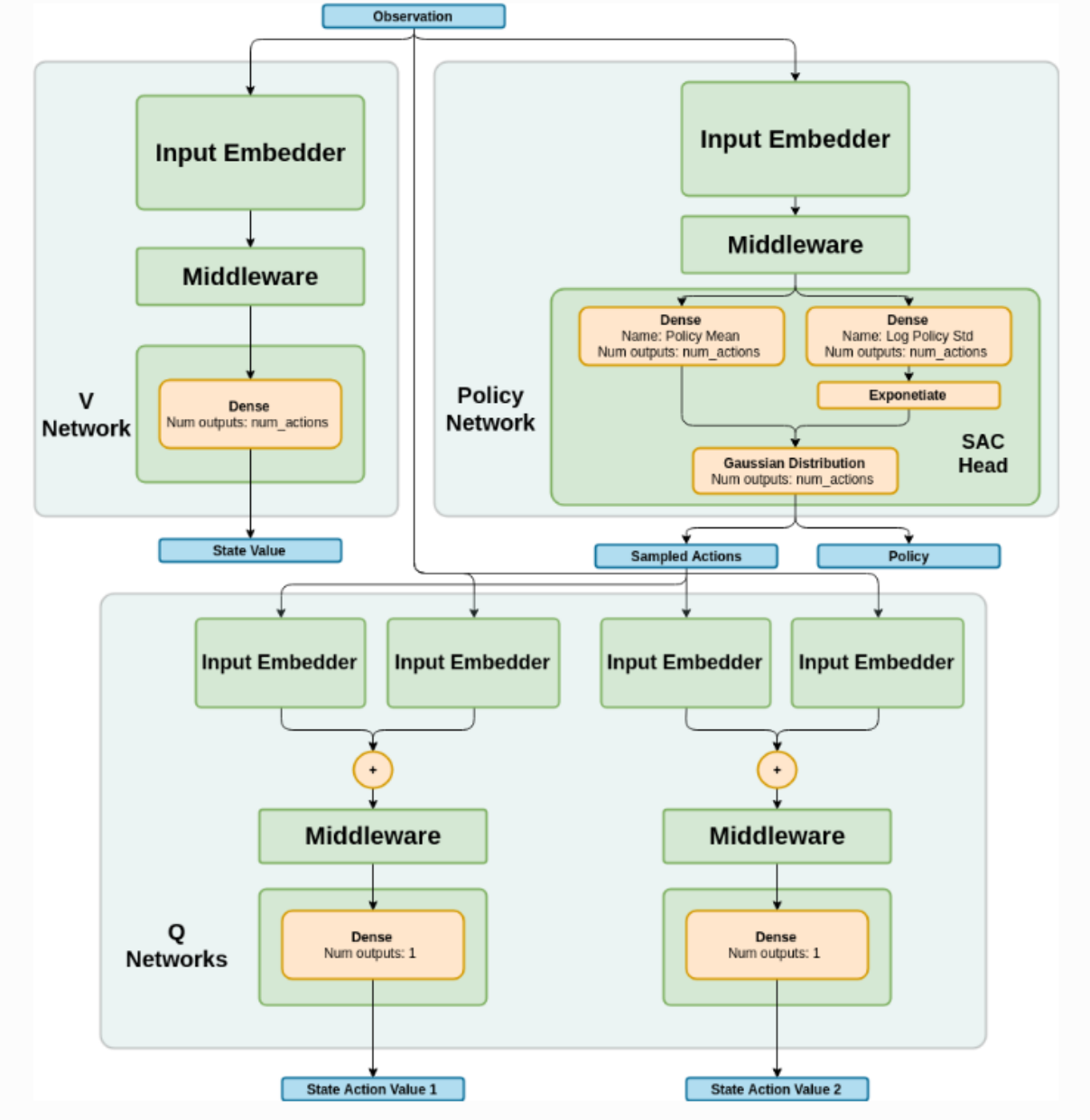


Рисунок 5.29 – Алгоритм роботи методу з використанням A2C

3. Proximal Policy Optimization (PPO) – це метод з квазіоптимальною політикою, який полягає в тому, щоб покращити стабільність політики навчання, обмеживши зміни, які відбуваються під час кожної епохи навчання, шляхом уникнення надто різких оновлень політики (рис. 5.30) [32, 33].

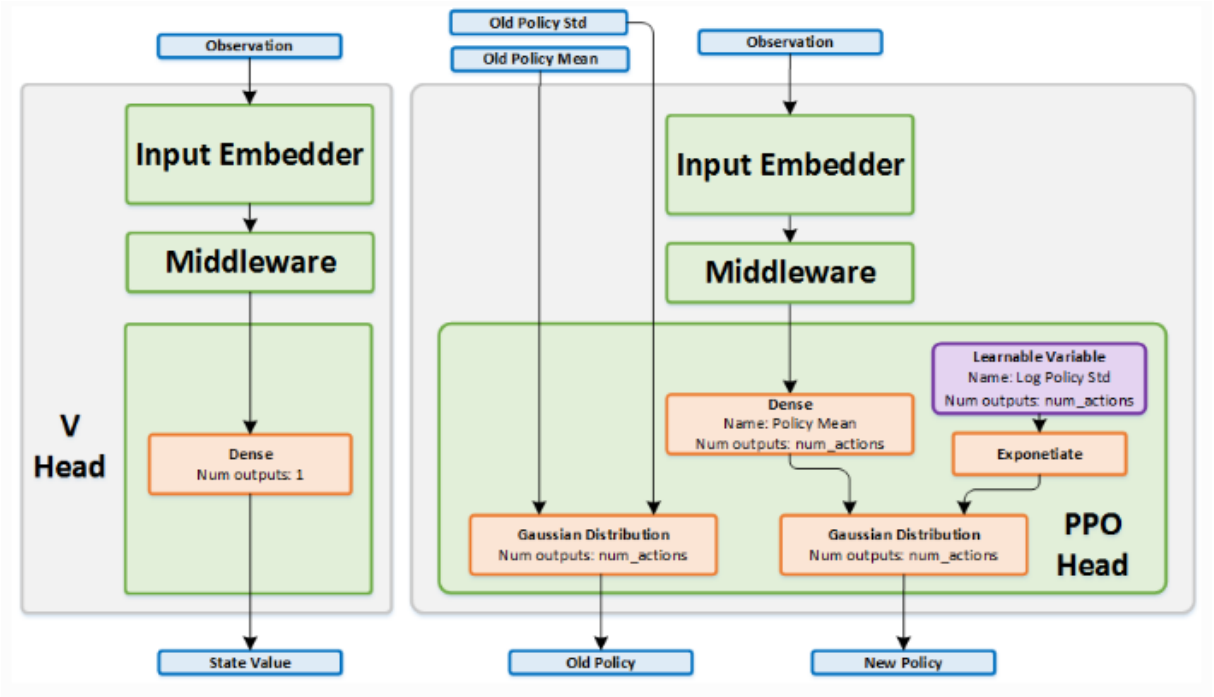


Рисунок 5.30 – Алгоритм роботи методу з використанням PPO

Існує ряд бібліотек, які можна використовувати для реалізації RL-алгоритмів. Ось деякі з найпопулярніших:

- OpenAI Gym – бібліотека, яка містить набори даних для різних RL-задач, таких як ігри, робота та фінансові ринки.
- Kaggle Environments – бібліотека, яка містить набори даних для RL-задач, які використовуються в конкурсах Kaggle.
- PyTorch – фреймворк машинного навчання, який включає в себе підтримку RL-алгоритмів.
- TensorFlow – фреймворк машинного навчання, який включає в себе підтримку RL-алгоритмів.
- Unity ML-Agents – це бібліотека, яка дозволяє створювати ігрові середовища для RL-досліджень. Ця бібліотека дозволяє розробникам створювати ігрові середовища з різними властивостями, такими як складність, розмір і кількість агентів.

Reinforcement Learning Coach – це структура Python, яка моделює взаємодію між агентом і середовищем модульним способом. За допомогою Coach можна моделювати агента, поєднуючи різні блоки та навчаючи агента в кількох середовищах. Доступні середовища дозволяють тестувати агента в різних областях, таких як робототехніка, автономне водіння, ігри тощо. Він надає набір простих у використанні API для експериментів з новими алгоритмами RL і дозволяє легко інтегрувати нові середовища для вирішення поставлених задач. Coach збирає статистику процесу навчання та підтримує розширені методи візуалізації для налагодження агента, який навчається.

Застосування навчання з підкріпленням має широкий спектр застосування, на приклад за темою НДР «Розробка інформаційних технологій для оптимізації роботи зернового елеватора з використанням нейромережевих

моделей та методів навчання з підкріпленням» на основі архітектури A2C розроблена технологія оптимізації енергоспоживання елеватора. Результати дослідження опубліковані на конференції в Мюнхені у тезах [32], також результати представлені у вигляді відео-звіту для європейських партнерів.

Можливі теми практичних і лабораторних завдань

Тема № 1. Ідентифікація параметрів нейронної мережі в задачах аналізу даних (або «**Побудова нейромережевих інтелектуальних моделей в задачах аналізу даних**»).

Метою роботи є вивчення підходів та прийомів до ідентифікації та оптимізації параметрів нейронної мережі з використанням Python-бібліотек в задачах аналізу даних й опанування практичних навичок з їх реалізації на прикладі одного із датасетів Kaggle чи за даними, завантаженими через API.

План заняття:

1. Вибрати датасет (див. розд. 1).

2. Побудувати 2-3 варіанти багатошарових нейронних мереж з використанням Keras (TF) чи PyTorch і порівняти їх передбачувальні можливості з багатошаровим перцептроном MLP бібліотеки Sklearn.

3. Побудувати нейронну мережу зі складною архітектурою.

4. Поекспериментувати з параметрами нейронних мереж у п.2, 3, щоб підвищити їх точність на валідаційних даних. Наприклад:

- кількість прихованих шарів мережі;
- розмір партії «batch size»;
- кількість нейронів у шарах «units in layers»;
- види функцій активації в шарах «activation functions»;
- кількість епох тюнінга моделі «epochs number»;
- параметри ReduceLROnPlateau («patience» та «factor»);
- відсоток даних, який буде приховуватись від одного шару до іншого «Dropout parameter» тощо).

5. Зробити висновки які прийоми були більш ефективними і яка модель виявилась оптимальною.

Рекомендується використати заготовку ноутбука: AI-ML-DS Training. L3AT: NH4 - NN models (тюнінг параметрів слід здійснювати у межах, відповідно до варіанту) і виконувати інструкції, написані в тексті програми після слів «TASK» та, для тих, хто вибрав спосіб 2 для виконання роботи (тобто, з суттєвим доопрацюванням чужого коду – див. вище) – «ADDITIONAL TASK».

Можна послухати відео з коментарями:

- Основи роботи з нейронними мережами. Ч.1. Теорія - Курс AI-ML-DS Training від GM Мокіна В.Б.
- Основи роботи з нейронними мережами. Ч.2. Практика - Курс AI-ML-DS Training від GM Мокіна В.Б.

- Основи роботи з нейронними мережами. Ч.3. Playground.TF - Курс AI-ML-DS Training від GM Мокіна В.Б.

Приклади ноутбуків:

- AI-ML-DS Training. L3AT: NH4 - NN models
- Heart Disease - Automatic AdvEDA & FE & 20 models
- Biomechanical features - 20 popular models
- Autoselection from 20 classifier models & L_curves

Контрольні питання

1) Які основні поняття нейронних мереж і глибокого навчання?
2) Як відбувається навчання нейронної мережі і як оцінюється її точність?
3) Які сучасні архітектури нейронних мереж і ансамблів ви знаєте?
4) Що таке машинне навчання з підкріпленням і яка його роль у штучному інтелекті?
5) Які переваги має використання ансамблів нейронних мереж у порівнянні з одиночними мережами?
6) Які методи використовуються для зниження перенавчання нейронних мереж?
7) Які інструменти і бібліотеки можна використовувати для створення нейронних мереж?
8) Які приклади задач можна розв'язувати за допомогою нейронних мереж?
9) Які переваги і недоліки має використання машинного навчання з підкріпленням?
10) Які алгоритми навчання використовуються для машинного навчання з підкріпленням, і як вони працюють?

6 ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ЗОБРАЖЕНЬ ТА ВІДЕО

6.1 Основні поняття, види задач та технології передоброблення

6.1.1 Основні поняття, кодування кольорів, тензори

Зазвичай, усі відео розглядаються як послідовність зображень. Кожне зображення (позначимо його U) розрізається на сукупність фрагментів (патчів – англ. «Chunks»), наприклад квадратів певного розміру (див. наприклад, статтю [34]). В загальному випадку, це може бути не квадрат, а – прямокутник, хоча таке рідко зустрічається. Кожен квадрат (патч) це – матриця пікселів (X та Y задають номер кожного пікселя по висоті та ширині цього квадрату). Для кожного пікселя задається кількість каналів кольорів C. Як правило: C = 3 у форматі RGB («Red – Green – Blue» – з англ. «Червоний – Зелений – Синій»). Або це може бути C = 1, якщо зображення – чорно-біле. У першому випадку, колір представляється 6-розрядним числом у 16-річній системі числення (рис. 6.4). А для чорно-білих зображень колір є бінарним: 0 або 1, чи, як в датасеті MNIST (див. нижче) – 0 або 255.

#FFCCCC	#FFC0C0	#FF9999	#FF8080	#FF6666	#FF4040	#FF3333	**#FF0000**
#FFE5CC	#FFE0C0	#FFCC99	#FFC080	#FFB266	#FFA040	#FF9933	**#FF8000**
#FFFFCC	#FFFFC0	#FFFF99	#FFFF80	#FFFF66	#FFFF40	#FFFF33	**#FFFF00**
#FFFFE5	#FFFFE0	#FFFFCC	#FFFFC0	#FFFFB2	#FFFFA0	#FFFF99	#FFFF80
#E5FFCC	#E0FFC0	#CCFF99	#C0FFA0	#B2FF66	#A0FF40	#99FF33	**#80FF00**
#CCFFCC	#C0FFC0	#99FF99	#80FF80	#66FF66	#40FF40	#33FF33	**#00FF00**
#E5FFE5	#E0FFE0	#CCFFCC	#C0FFC0	#B2FFB2	#A0FFA0	#99FF99	#80FF80
#CCE5CC	#C0E0C0	#99CC99	#80C080	#66B266	#40A040	#339933	**#008000**
#CCFFE5	#C0FFE0	#99FFCC	#80FFC0	#66FFB2	#40FFA0	#33FF99	**#00FF80**
#CCFFFF	#C0FFFF	#99FFFF	#80FFFF	#66FFFF	#40FFFF	#33FFFF	**#00FFFF**
#E5FFFF	#E0FFFF	#CCFFFF	#C0FFFF	#B2FFFF	#A0FFFF	#99FFFF	#80FFFF
#CCE5E5	#C0E0E0	#99CCCC	#80C0C0	#66B2B2	#40A0A0	#339999	**#008080**
#CCE5FF	#C0E0FF	#99CCFF	#80C0FF	#66B2FF	#40A0FF	#3399FF	**#0080FF**
#CCCCFF	#C0C0FF	#9999FF	#8080FF	#6666FF	#4040FF	#3333FF	**#0000FF**

Рис. 6.1 Приклади RGB-кольорів пікселів у 16-річній системі числення

Отже, кожен фрагмент зображення U кодується у вигляді чотиривимірної матриці:

$$U = [<N>, <X>, <Y>, <C>], \quad (6.1)$$

де в кожній комірці вписано число, які відповідає кольору відповідного пікселя.

Така багатовимірна матриця чисел (6.1) називається *тензором* (англ. «tensor» – не плутати з тензорами у математичній фізиці – вони не мають нічого спільного!). Якщо зображення – невеликого розміру, тоді вони можуть не розбиватись на фрагменти і, навпаки, якщо зображення – дуже великі, тоді вони розбиваються не тільки на фрагменти, а ще й – на партії (англ. «batch») цих фрагментів. Тобто це вже – 6-вимірна матриця (масив) даних або – двовимірний тензор.

Наприклад, цифра датасету MNIST є невеликою – 28 на 28 пікселів, чорно-біла. Отже, вона не розбивається ні на партії, ні на фрагменти і є одновимірним тензором. Оскільки вона – чорно-біла, тоді кількість каналів дорівнює C=1, а числа у комірках тензора – це 0 (білий) або 255 (чорний колір) (рис. 6.1): (1, X, Y, 1), де X та Y – це цілі числа від 0 до 27 включно. А колір пікселя зображення якості HD буде 16-річним числом, наприклад, за номером (8, 16, 22, 2), означає 9-й фрагмент (нумерація – з нуля), 16 і 22 це – координати у цьому фрагменті, 2-й канал («зелений») кольору.

Для деяких задач, як альтернатива RGB, використовується HSV-кодування кольорів («Hue – Saturation – Value» – з англ. «Відтінок – Насиченість – Value»). HSV використовується для задання діапазону кольорів, наприклад як маски під час фільтрування чи ін., коли важливо задати діапазон різних кольорів однієї гами, тоді достатньо задати варіацію одного числа в циклі (рис. 6.5). Іноді HSV-кодування подають у циліндричній системі координат.

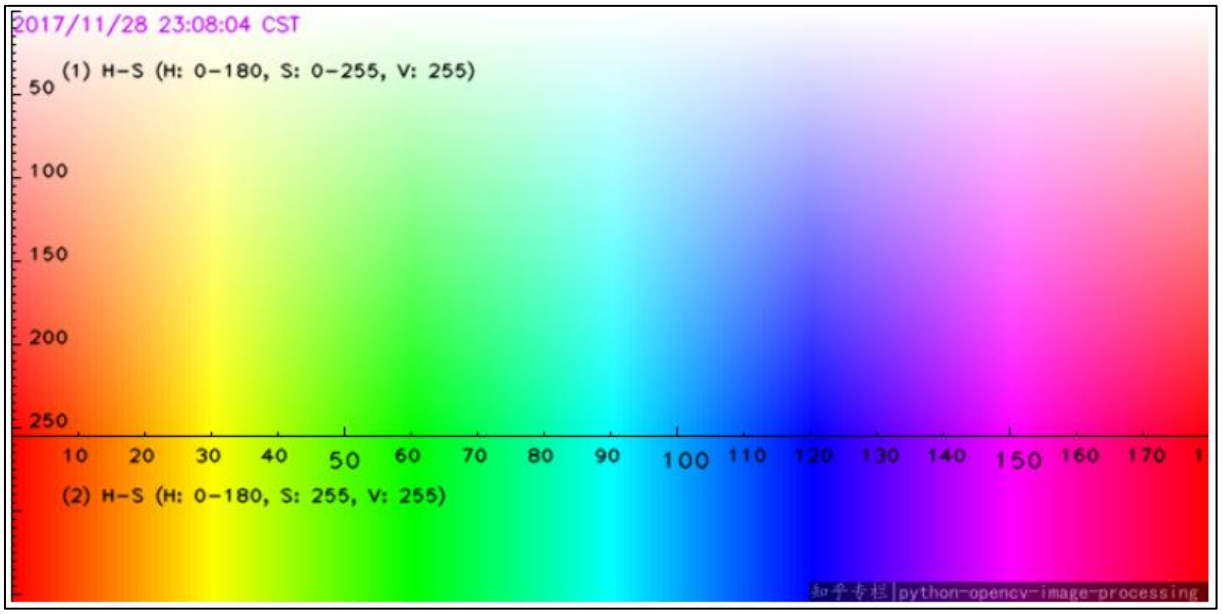


Рис. 6.2 Усі варіанти значень HSV-кодування та їх подання у декартовій системі координат

6.1.2 Стандартні датасети із зображеннями

Існують спеціалізовані розмічені датасети із зображеннями, на яких тестують і порівнюють усі відомі моделі:

1. MNIST («Mixed National Institute of Standards and Technology») — база даних зразків рукописного написання цифр. Є стандартом, запропонованим Національним інститутом стандартів і технологій США з метою калібрування і порівняння методів розпізнавання зображень за допомогою машинного навчання, в першу чергу на основі штучних нейронних мереж. Дані доступні і в Kaggle (рис. 6.1).

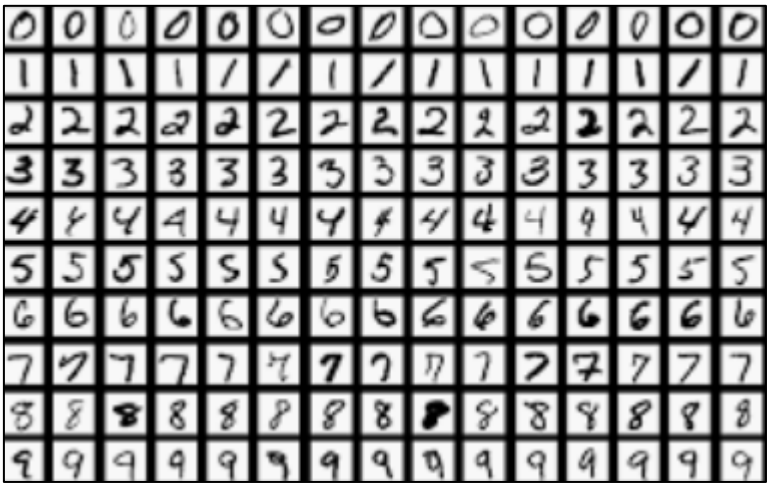


Рисунок 6.3 – Зображення датасету «MNIST» у Kaggle: розмічені, нормалізовані, відцентровані пройшли згладжування та приведення до напівтонового зображення розміром 28x28 пікселів

Рекордні результати машинного розпізнавання на базі MNIST були досягнуті на згорткових нейронних мережах, рівень помилки був доведений до 0,23 %

2. MNIST Fashion – це велика база даних зображень одягу, подібна до класичного MNIST, але замість рукописних цифр містить 10 категорій предметів одягу. Кожне зображення розміром 28x28 пікселів у форматі відтінків сірого, з яскраво визначеними ознаками для полегшення розпізнавання. MNIST Fashion часто використовується для навчання та порівняння моделей машинного навчання, особливо тих, що спеціалізуються на класифікації зображень.

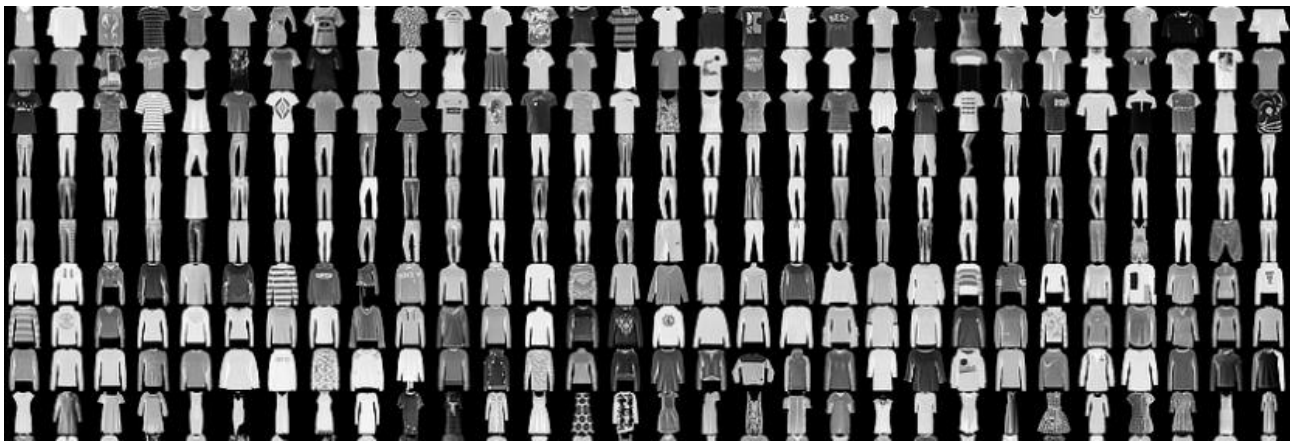


Рисунок 6.4 – Зображення датасету «MNIST Fashion»: розмічені, нормалізовані, відцентровані пройшли згладжування та приведення до напівтонового зображення розміром 28x28 пікселів

3. ImageNet - це великий набір зображень. Існує лексична база слів англійської мови WordNet, де слова групуються в синсети (synsets). Синсет - це множини синонімів, які вказують на один і той же концепт або об'єкт. ImageNet використовує WordNet як основу для класифікації зображень. Тобто кожному синсету WordNet відповідає датасет зображень у ImageNet і це дозволяє більш повно аналізувати і слова, і зображення. Станом на листопад 2023 року в ImageNet: 14,2 млн. зображень та, 21,8 тис. проіндексованих синсетів (рис. 6.3).

Рисунок 6.5 – Зображення датасету «ImageNet»

6.1.3 Типові постановки задач

Класичними постановками задач оброблення та аналізу зображень з використанням Data Science є такі:

1. Класифікація зображень (англ. «Image classification») – визначення класу чи категорії зображень. Це може бути як конкретний вид (обличчя, номер автомобіля, чи є людина в масці, військова техніка за даними супутникової чи аерофотозйомки тощо), так і класифікація певних змін об'єкта (емоції на обличчях, руйнування будинків тощо).

2. Виявлення об'єктів (англ. «Object Detection») – визначення та локалізація об’єктів на зображенні. Виявлення транспортних засобів на вулицях

(підрахунок трафіку, пошук заторів), визначення обличчя людини та її точкових ключових точок, виявлення точок для аналіз відбитків пальців тощо.

3. Семантична сегментація (англ. «Semantic Segmentation») – призначення класу кожному пікселю на зображенні: визначення різних класів на зображенні з високою точністю, таких як визначення контурів дороги та тротуарів. Ще це використовується під час аналізу відео в режимі онлайн з використанням технології YOLO.

4. Генерування зображень (англ. «Image Generation») – створення нових зображень (так званих діпфейків – з англ. «Deep Fake») на основі навчальних наборів даних та спеціальних моделей глибокого навчання. Більш детально це буде описано у підрозд. 6.3, оскільки це зараз один з найбільш потужних сучасних напрямків удосконалення моделей і методів машинного навчання. Зображення можуть створюватись як із декількох зображень (GAN-моделі та ін.), так і за текстовим описом (моделі Stable Diffusion та ін.).

5. Виявлення відмінностей (англ. «Anomaly Detection») – виявлення незвичайних або аномальних паттернів (закономірностей) на зображеннях. Наприклад, є велика кількість зображень сільськогосподарських полів за результатами аерофотозйомки. Необхідно знайти зображення, на яких є пошкоджені рослини чи інші проблемні місця, які потребують швидкого реагування. Якщо йдеться про тисячі гектарів кукурудзи чи інших рослин ростом з людину, то задача не є тривіальною. Саджати агронома на літак чи давати йому 1 квадракоптер, щоб він дивився на ті поля сам – нерентабельно. Більш ефективно, запустити багато квадракоптерів, які в автоматичному режимі обстежать всю територію по заданій програмі, а потім в автоматичному режимі знайти усі аномалії, а вже потім їх показати агроному чи іншому експерту, який класифікує ці аномалії. Згодом, результат такої класифікації можна використати і для навчання штучного інтелекту робити таку класифікацію самостійно та показувати замовнику в динаміці розвитку явища (за даними аерофотозйомки у різний час). Приклад такої задачі описано у статті одного з авторів [34].

6. Розпізнавання облич чи частин тіла людини (Face Recognition) – це підвид одного з попередніх видів. Але, через високу важливість і популярність, часто виділяють як окремий клас задач. Ідентифікація осіб на зображеннях, у т.ч. по повороту обличчя (додаток «Дія»), розпізнавання відбитка пальця на смартфоні – це задачі, з рішеннями яких люди щодня зустрічаються частіше, ніж з іншими.

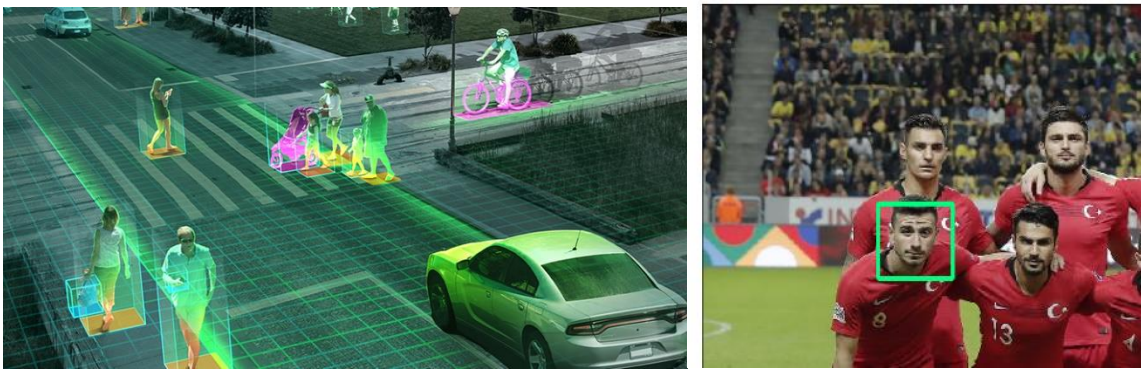
7. Покращення та генерування зображень (Image Enhancement and Generation) шляхом застосування фільтрів та трансформацій для покращення або генерування нових зображень, покращення якості зображень, створення артістичних та креативних зображень. Для розв’язання подібних задач часто використовується бібліотека OpenCV. Розглянемо її детальніше.

6.1.4 Передоброблення зображень. Бібліотека OpenCV

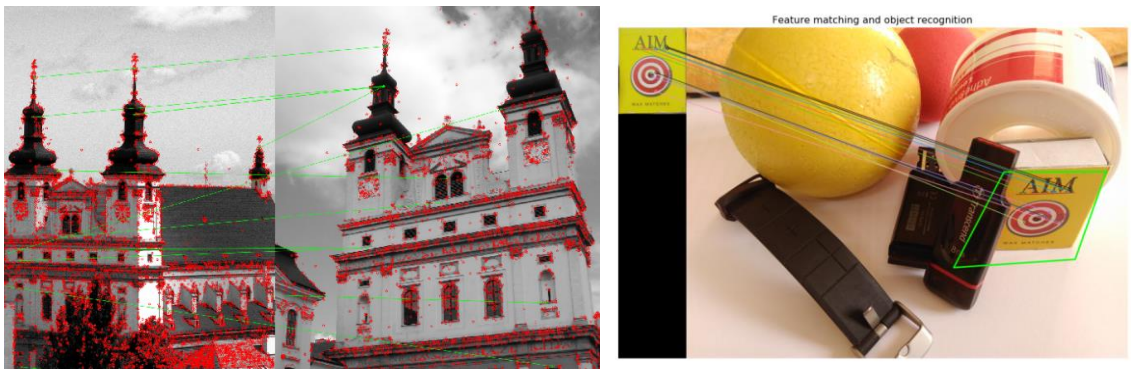
Бібліотека «OpenCV» (скорочено від «Open Source Computer Vision Library») – з англ. «Бібліотека програмного забезпечення з відкритим кодом для машинного навчання»: комп'ютерний зір, оброблення відеозйомки, оброблення зображень тощо. Підтримка різних платформ (Windows, Linux, Android, iOS) і мов програмування (Python, C#, C++, Java, Ruby).

Бібліотека містить понад 2500 оптимізованих за швидкістю і точністю алгоритмів для різних цілей (рис. 6.6):

- виявлення та розпізнавання обличь, ідентифікація об'єктів на зображеннях чи на відео (відео розрізається на окремі зображення і аналізується покадрово);
- класифікація дій людини на відео;
- відстеження рухів камери;
- відстеження рухомих об'єктів (наприклад, під час футбольних матчів камера може автоматично відслідковувати м'яч на полі);
- побудова 3D-моделей, отримання 3D-хмар точок зі стереокамер, що дозволяє одразу будувати та аналізувати модель усього навколишнього середовища в динаміці навколо БПЛА (автомобіль, літак чи квадрокоптер);
- з'єднання зображень разом для отримання високої роздільної здатності зображення цілої сцени;
- знаходження подібних зображень у базі даних зображень;
- видалення «червоних очей» із зображень, зроблених за допомогою спалаху;
- слідкувати за рухами очей;
- розпізнавати краєвиди;
- встановлення маркерів для накладання їх на розширену реальність тощо.



а) б)



в) г)

Рисунок 6.6 – Приклади роботи бібліотеки OpenCV: а) побудова 3D-моделей та ідентифікація зображень; б) розпізнавання обличь; в) знаходження подібних архітектурних форм; г) знаходження подібних зображень, з урахуванням їх геометричних трансформацій

У ноутбуці наведено приклад, який дозволяє краще зрозуміти основні принципи роботи OpenCV з реальним відео. Це відео отримується безпосередньо з відеокамери, підключеної до комп'ютера чи з відеокамери ноутбука (лептопа). Для запуску програми слід скопіювати закоментований текст з ноутбука, прибрати статус коментування і запустити на тому комп'ютері, де є відеокамера. Код у Kaggle закоментовано, оскільки його не можна запустити у хмарі – там немає доступу до периферійних пристроїв.

Програма виконує наступне:

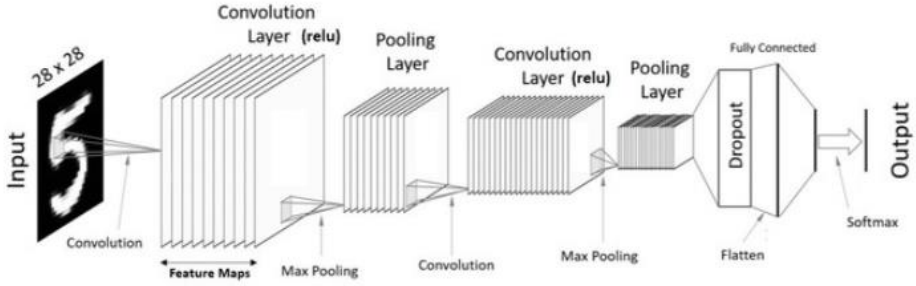
1. Захоплює відео з відеокамери у форматі RGB.
2. Перетворює колір RGB у HSV.
3. Користувач задає маску кольору, користуючись рис. 6.5.
4. Задаються параметри згладжування.
5. Кольори, які відповідають масці, заміняються на інший (наприклад, уся зелена чи синя гама заміняється на білий).
6. Зображення згладжується і виводиться на екран.
7. Зображення виводяться безперервно як послідовність, тобто – як відео, у циклі, поки користувач не натисне клавішу «q».
8. Потім вікно закривається – це обов'язкова команда для роботи з відео.

6.2 Моделі для класифікації зображень та відео: CNN, AE, YOLO та ін.

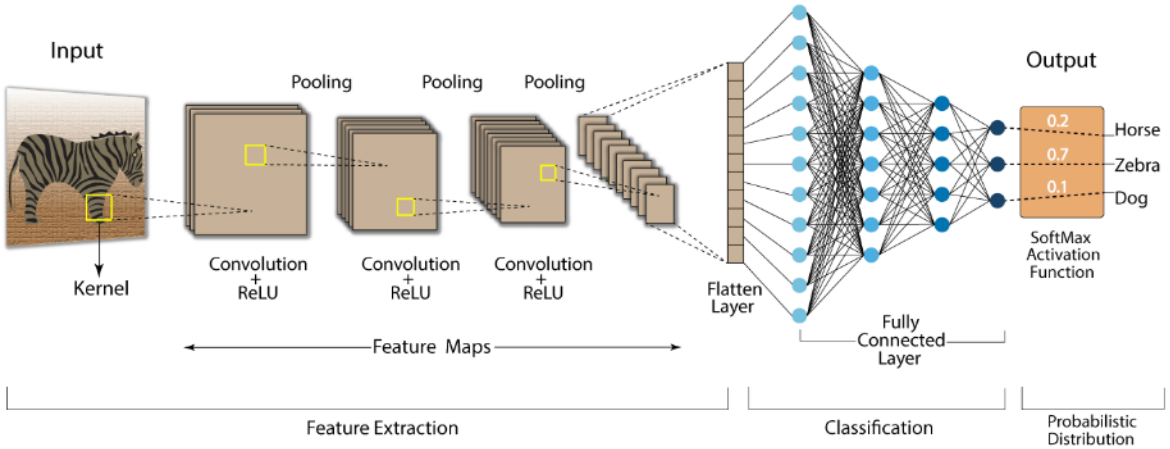
6.2.1 Згорткові нейромережі для аналізу зображень

CNN («Convolutional Neural Network» – з англ. «згорткова нейронна мережа») – в наш час є найбільш поширеною для аналізу зображень та класифікації об'єктів на них. Також, вона часто застосовується і для аналізу інших багатовимірних даних, однак, краще її принцип пояснити саме на прикладі зображень.

Розглянемо застосування CNN до розпізнавання та класифікації цифр датасету MNIST. Як було зазначено вище, він містить чорно-білі зображення арабських цифр у вигляді матриць 28x28 пікселів, відцентрованих і очищених від артефактів. Таке кодування є найбільш спрощеним і часто використовується для пояснення архітектури та принципів роботи елементів CNN (рис. 6.7а,б).



а)



б)

Рис. 6.7. Типова архітектура згорткової нейронної мережі: а) для розпізнавання цифр датасету MNIST [35]; б) для розпізнавання виду тварини [36]

Пояснимо принцип роботи кожного елементу типової архітектури CNN (рис. 6.7). Основним елементом є згортка. Для її застосування потрібна вхідна матриця чисел MxM (в загальному випадку, вона може бути прямокутною, а не квадратною, але розглянемо варіант тільки квадратних матриць), яка характеризує зображення, та ядро згортки WxW (теж може бути прямокутною), яке дозволяє «згорнути» підматрицю такого ж розміру WxW. Ці матриці поелементно перемножаються і результат (одне число) вписується у результуючу матрицю у відповідну комірку (рис. 6.8).

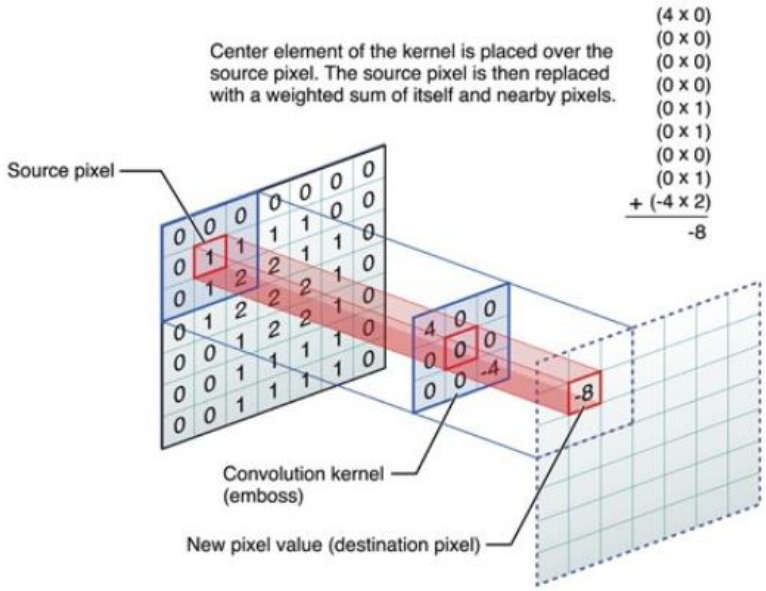


Рис. 6.8. Ілюстрація процесу згортки шляхом перемноження підматриці вхідної матриці (зліва) на матрицю-ядро (по середині), результатом чого є вписування одного числа у комірку результуючої матриці [35]

Операція згортки застосовується в циклі і «згортає» усі елементи вхідної матриці. Вочевидь, якщо перемножити матрицю MxM= = 7x7 на матрицю-ядро (англ. «kernel») WxW = 3x3, то це вдасться зробити тільки $M - W + 1 = 7 - 3 + 1 = 5$ разів. Такий варіант згортки (за це відповідає параметр «padding») називається «value» (у ноутбук2 є гарна gif-ілюстрація, яка в динаміці показує як з використанням ядра у режимі «value» формується результуюча матриця). Але більш популярним є варіант padding = «same», коли розмір результуючої матриці – такий самий, як і у вхідної: MxM. Для такої згортки до вхідної матриці дописують нулі (0) в сусідні комірки, щоб забезпечити можливість застосування операції, скільки треба разів (рис. 6.9).

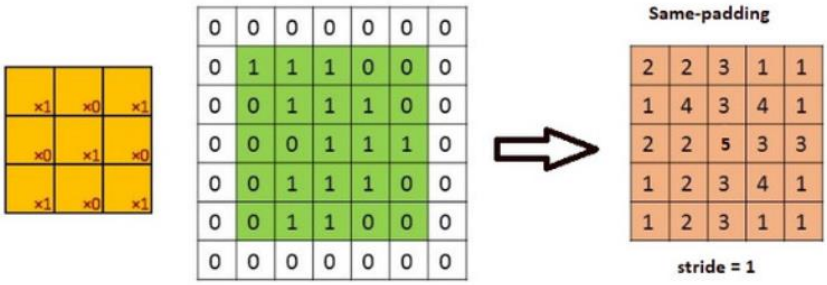


Рис. 6.9. Ілюстрація режиму згортки з параметром padding = «same» шляхом формального додавання нулів до вхідної матриці 5x5, щоб її згортка з ядром 3x3 дала результуючу матрицю теж 5x5 [35]

Важливо, що ядро WxW може застосовуватись не до кожної аналогічної за розміром частини вхідної матриці підряд, а – зі стрибками на S (англ. «stride») кроків вправо і вниз (в загальному випадку, розмір цих стрибків і

розміри вхідної матриці та ядра можуть бути різними по вертикалі і горизонталі). Крім того, згортка застосовується багато разів, але людина тільки вказує архітектуру. Які саме вибирати маски – це вирішує алгоритм навчання цієї нейромережі.

Застосування згортки з використанням фреймфорку Keras (TF) виглядає таким чином:

x = Conv2D(9, (3, 3), activation='relu', stride = (2, 2), padding='same')(x)

що означає, що до вхідного зображення 9 разів застосовується двовимірна згортка Conv2D 3x3 пікселя, функція активації – ReLU (рис. 6.10), ядро стрибає на 2 кроки по вертикалі і 2 по горизонталі, а результат згортки буде того ж розміру, що й вхідне зображення.

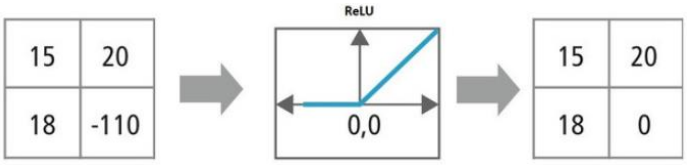


Рис. 6.10. Приклад застосування функції активації ReLU до згорткового шару нейромережі, яка заміняє усі від'ємні значення на нуль «0» [36]

Кожна така згортка дозволяє знайти на вхідному зображенні якісь важливі елементи (англ. «pattern») і зберегти її основні елементи у результуючому зображенні. Наприклад, на рис. 6.11 наведено яким чином згортка дозволяє здійснювати сегментацію зображення (виявлення границь).

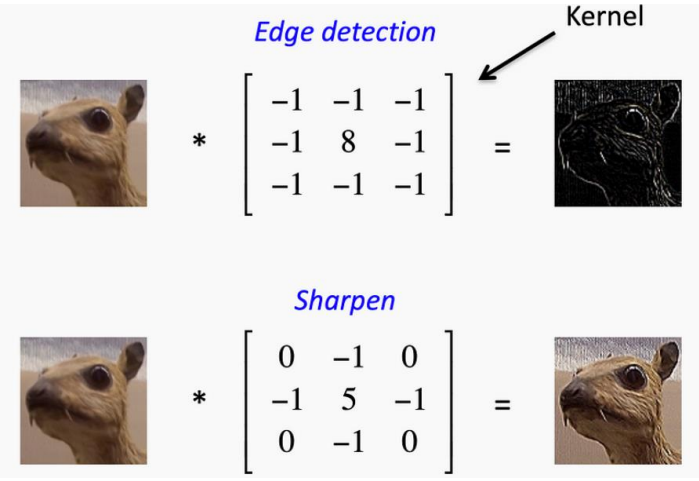


Рис. 6.11. Приклад застосування різних видів згорток до вхідного зображення (з Kaggle ноутбуку)

В результаті застосування згортки вихідне зображення буде доволі різноманітним. А тому в парі з кожною згорткою, як правило, застосовується операція MaxPooling або AveragePooling (див. рис. 6.7). Ця операція має

схожі параметри, як і згортка, але трансформація підматриці вхідної матриці до одного числа здійснюється не шляхом перемноження на іншу матрицю, а – визначенням максимального або середнього значення, відповідно, від усіх елементів цієї підматриці (рис. 6.12):

MaxPooling2D(pool_size=(2, 2), strides=(2,2), padding="valid")
MaxPooling2D(pool_size=(4, 4), strides=None, padding="valid")
AveragePooling2D(pool_size=(3, 3), strides=3, padding="same")

де pool_size – розмір підматриці вхідної матриці, до якої застосовується операція, strides – на скільки робиться стрибок по вертикалі та горизонталі (одне число означає однаковий стрибок уздовж обох осей, None означає відсутність стрибка або що стрибок, формально, дорівнює 1); padding має те саме значення, що й для згортки, але у разі «same» елементи результуючої матриці, в які немає що вписати, заповнюються нулями («обрамляють» результуючу матрицю у режимі «value» по периметру).

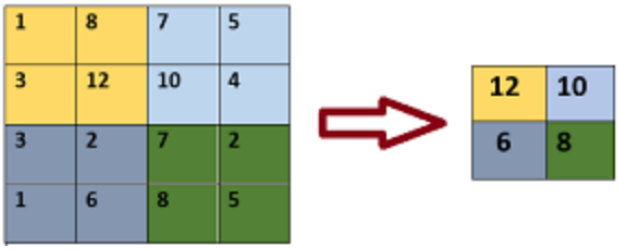


Рис. 6.12. Приклад застосування операції
MaxPooling2D(pool_size=(2, 2), strides=None, padding="valid")
до вхідного зображення [36]

Як видно на рис. 6.12, MaxPooling2D із вказаними параметрами зменшує розмірність кількість елементів матриці у 4 рази, зберігаючи при цьому, основну інформацію.

Формально, вважається, що згорткою є тільки операція Conv, на відміну від MaxPooling, AveragePooling, оскільки у них не використовується ядро згортки. Усі ці 3 види операцій мають варіанти для 1D, 2D, 3D, залежно від розмірності вхідних даних (можуть бути вироджені тензори меншої розмірності). Для випадку 3D параметри pool_size та strides мають вигляд кортежу, відповідно, із 3-х чисел, наприклад, «pool_size = (4, 4, 4)».

Як правило, шари Conv використовуються з параметром padding = «same», а MaxPooling, AveragePooling – padding = «value». Тобто перший здійснює видобування корисної інформації, а другий – зменшення обсягу даних. Ці шари використовуються парами (дис. рис. 6.7) або блоками (2-3 згортки і потім Pooling). На їх виході, іноді, ставлять Dropout, згадуваний у підрозд. 5.2 (див. рис. 5.13). Його можуть ставити і після кожної пари, і – один раз, в кінці, як на рис. 6.7а.

Після пар «Conv-Pooling» з Dropout використовується шар «Flatten», який перетворює усі результуючі матриці в один одновимірний масив даних (рис. 6.13).

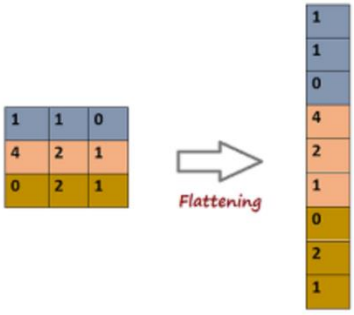


Рис. 6.13. Приклад застосування операції Flatten
до результатів пар «Conv-Pooling» з Dropout [36]

Як видно на архітектурі CNN на рис. 6.7, дані з Flatten надходять на класичну багатошарову повнопов'язану (англ. «Fully Connected») нейромережу. Якщо це – задача класифікації, тоді в останньому шарі цієї нейромережі, як правило, використовують функцію активації Softmax. Ця функція працює шляхом застосування функції експоненційного розподілу до вихідного значення мережі. Вона перетворює ці значення на числа, які представляють ймовірність того, що вхідний зразок належить до кожної з можливих класів. Простіше кажучи, вона дозволяє більш точно визначити який саме клас є результуючим.

Головний принцип і перевага CNN в тому, що дослідник тільки формує архітектуру і задає у вигляді гіперпараметрів, а значення усіх матриць-ядер згортки та інші внутрішні параметри обчислюються автоматично.

В наш час CNN отримали велике поширення і не тільки в задачах розпізнавання зображень. Вони працюють і для аналізу тексту, і для аналізу табличних даних, і для прогнозування часових рядів, оскільки вміють вдало обробляти різну числову інформацію.

6.2.2 Складні архітектури згорткових нейромереж

Для розпізнавання зображень використовуються більш ефективні та складні архітектури, наприклад модель EfficientNet. Її основна ідея полягає в тому, щоб знайти оптимальне співвідношення між глибиною мережі, шириною (кількістю фільтрів у кожному шарі) та роздільною здатністю (розміром вхідних зображень). Замість простого збільшення одного з цих параметрів, EfficientNet використовує коєфіцієнти масштабування для автоматичного налаштування усіх трьох аспектів, знаходячи баланс між точністю та обчислювальною ефективністю. Є багато її варіацій та версій.

Для розпізнавання і класифікації багатьох зображень на одному зображенні використовується CNN-варіація R-CNN (англ. «Region-based Convolutional Neural Network») – це алгоритм для об'єктного виявлення в зображеннях. Він був однією з перших спроб використання глибокого нав-

чання для розв'язання задачі виявлення об'єктів. Основна ідея R-CNN полягає в тому, щоб розбити зображення на регіони і намагатися визначити, чи є в цих регіонах об'єкти, і які саме. Більш сучасним варіантом є модель Faster R-CNN.

Існує веб-портал «Papers with Code», який реєструє і відображає на одному графіку усі відомі моделі та їх точність, там же є код і опис кожної моделі. Звідси – й назва. На рис. 6.14 наведено усі моделі, у т.ч. найкращі у різний час, станом на листопад 2023 року.

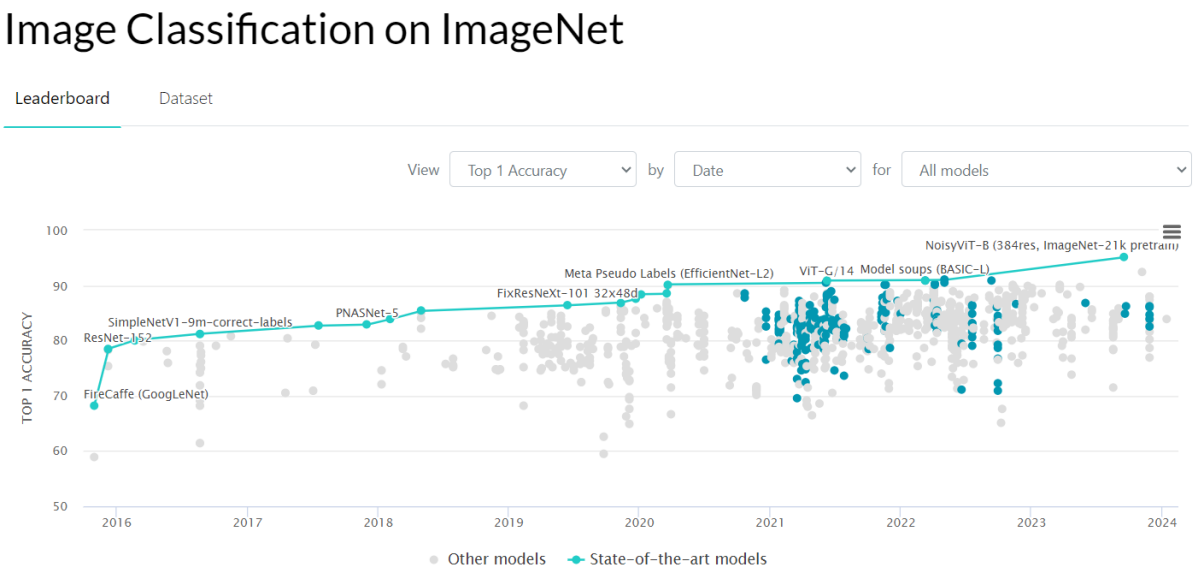


Рисунок 6.14 – Порівняння точності моделей для класифікації зображень датасету ImageNet на порталі «Papers with Code» (синім виділено моделі класу «Transformer»)

Як видно на рис. 6.14, останнім часом найбільш точними є моделі класу «Transformer», у т.ч. моделі ViTs (англ. «Vision Transformers»). Більше детально вони будуть описані далі у розд. 7, оскільки вони є основними і для задач класифікації природномовного тексту. Власне, їх придумали для аналізу тексту, а потім поширили і для аналізу зображень.

Для навчання CNN зі складною архітектурою, часто, не достатньо обсягу вхідних даних чи їх різноманітності. Тоді застосовують так звану аугментацію (англ. «Augmentation»), тобто штучне збільшення датасету.

Якщо датасет містить зображення, тоді, як правило, використовуються методи бібліотеки OpenCV: повертання зображення на різний кут, обертання навколо однієї з осей, деформація (стиснення, розтягування та ін.), додавання чи прибирання шуму, зміна розміру чи роздільної здатності тощо (рис. 6.15).

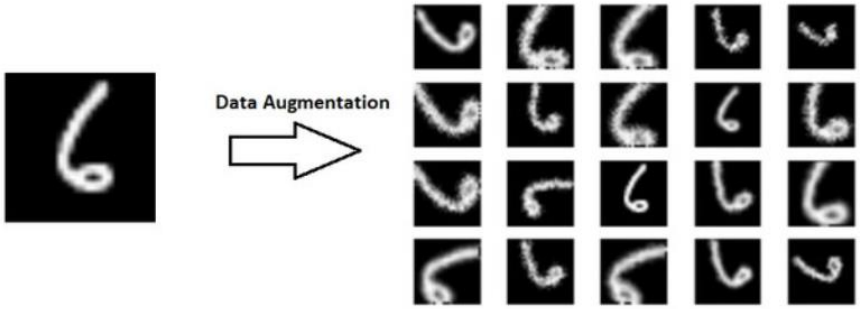


Рисунок 6.15 – Приклад аугментації зображень з ноутбуку

Доволі ефективними для деяких задач є архітектури ResNet (англ. «Residual Networks») і ResNeXt. Однією з головних ідей ResNet є те, що замість того, щоб напряму навчатись між входом і виходом, вона навчається по різниці між ними, намагаючись зменшити її до нуля. Це допомагає уникнути так званої проблеми «зникаючого градієнта» і дозволяє будувати більш глибокі та ефективні нейронні мережі.

Ще більш ефективною є модель ResNeXt. Вона у кожному блоці здійснює декілька паралельних відгалужень, які називаються «кардинальностями» (з англ. «cardinalities»). Кожне відгалуження це – набір згорткових шарів, які витягують різні ознаки з даних. Після цього, виходи з цих відгалужень агрегуються у певний спосіб. Кожне таке відгалуження може мати власні параметри, які навчаються окремо. Це дозволяє ResNeXt одночасно урізноманітнювати структуру і параметри нейромережі та забезпечує паралелізацію обчислень, а отже – підвищує точність і швидкодію одночасно. Нижче буде наведено вдалі приклади застосування цих архітектур CNN.

6.2.3 Автоенкодери у задачах без вчителя

Як було описано вище, існують задачі без вчителя, де треба знайти невідомі паттерни (з англ. «pattern» – якісь закономірності чи аномалії) у великому датасеті. Одним з основних способів вирішення такої задачі є використання згорткового автоенкодера (англ. «Convolution AutoEncoder» – «CAE»).

В літературі іноді зустрічається термін «автокодувальник» (див. наприклад [37]), що є більш правильним з точки правил зору української мови, але на практиці більш поширеним є англіканізм «автоенкодер», тому будемо вживати саме його.

Основний принцип роботи автоенкодера ілюструє рис. 6.16.

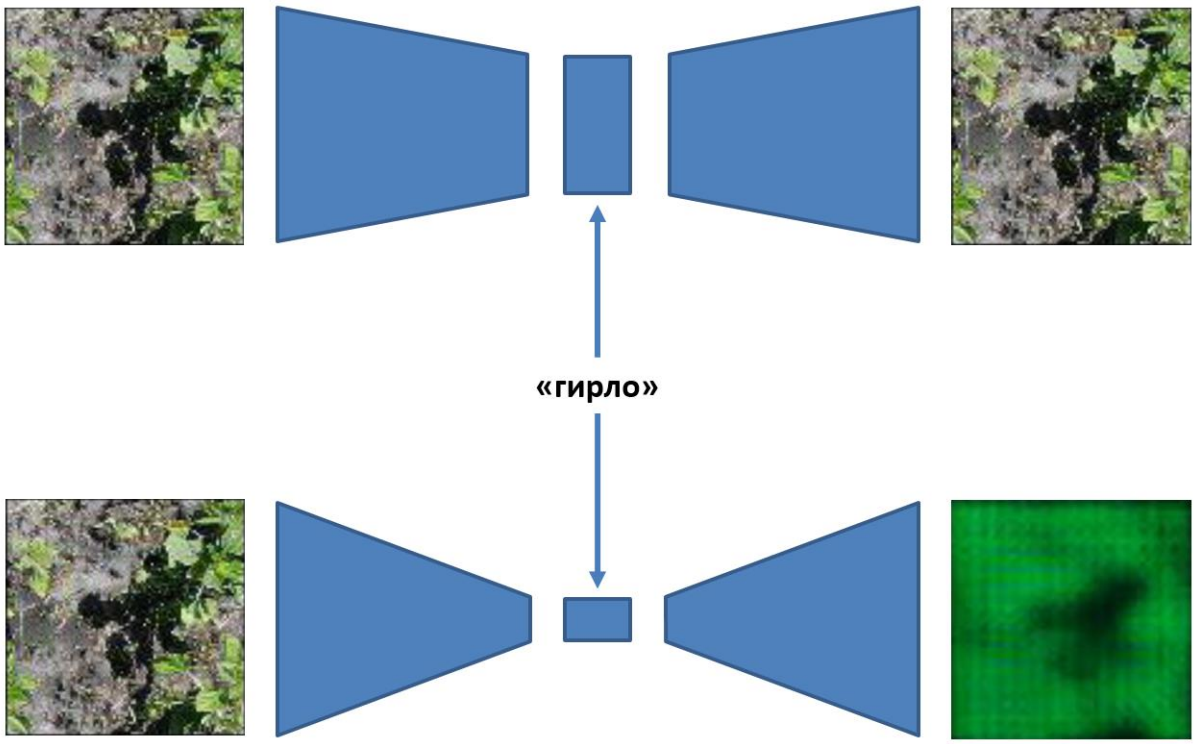


Рисунок 6.16 – Принцип роботи автоенкодера (верхній рисунок – невелике стиснення, нижній – «гирло» є меншим майже у 400 разів)

Вхідне зображення подається на багатошарову згорткову нейронну мережу, яка здійснює перетворення з одночасним зменшенням розмірності до умовного «гирла». А потім нейромережею із віддзеркаленною архітектурою і тими ж самими параметрами здійснює зворотне перетворення із цього ж «гирла» у вихідне зображення. Метрикою є максимальна подібність вхідного і вихідного зображення. Регулюючи розмірність «гирла» можна легко регулювати розмір вихідного зображення. Як видно з нижнього рисунку на рис. 6.6, якщо у гирлі задати дуже малий розмір, тоді якість вихідного зображення буде сильно відрізнятись від вихідного, але воно буде містити якісь важливі паттерни, для виділення яких спеціальним чином будується згорткова нейромережа. Крім того, цікавим є й саме «гирло», адже в ньому будуть міститись основні патерни (особливості) вхідного зображення, але, при цьому, через його малий розмір, його значно легше аналізувати. Наприклад кластеризувати «гирло» для багатьох зображень набагато легше, аніж – вхідні зображення, наприклад кольорові зображення з «Full HD». Отже, такий згортковий автоенкодер має такі цінні властивості:

1) сильне стиснення вихідного зображення до вихідного, яке займає менше пам’яті, наприклад, цей метод використовують деякі сервіси для передавання зображення співрозмовника за умов дуже поганого зв’язку (тренують мережу, а потім передають тільки «гирло» і щоразу відновлюють зображення);

2) виділення ключових патернів, які видно у вихідному зображенні і які зручніше аналізувати;

3) виділення ключових патернів, які видно у «гирлі» і які зручніше аналізувати.

Наприклад, один зі співавторів посібника використовував CAE для пошуку аномальних зон пошкодженої рослинного на полі за даними аерофотозйомки (рис. 6.17) [34].

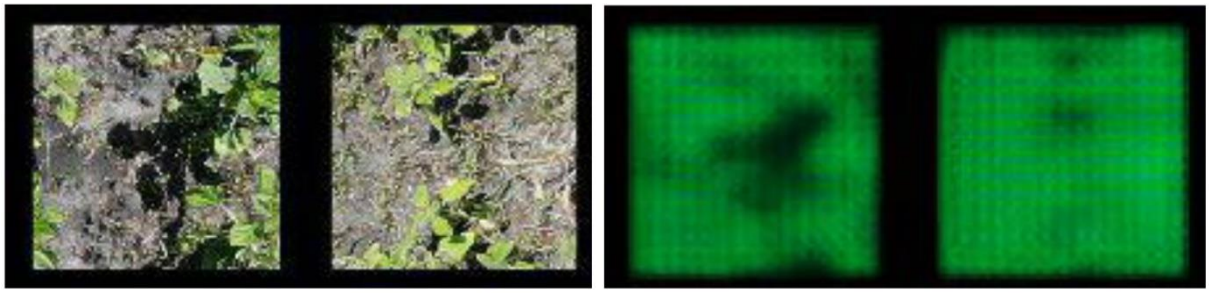


Рисунок 6.17 – Приклади застосування CAE з розміром «гирла» 72 до даних аерофотозйомки 27648 = 96x96x3 (стиснення у 384 разів) [34]

Код для CAE опубліковано у статті [34]. Важливо відмітити, що для віддзеркалення згорткових шарів у другій частині вони заміняються на свої аналоги, які виконують зворотну трансформацію:

Conv2D, MaxPooling2D => Conv2DTranspose, UpSampling2D

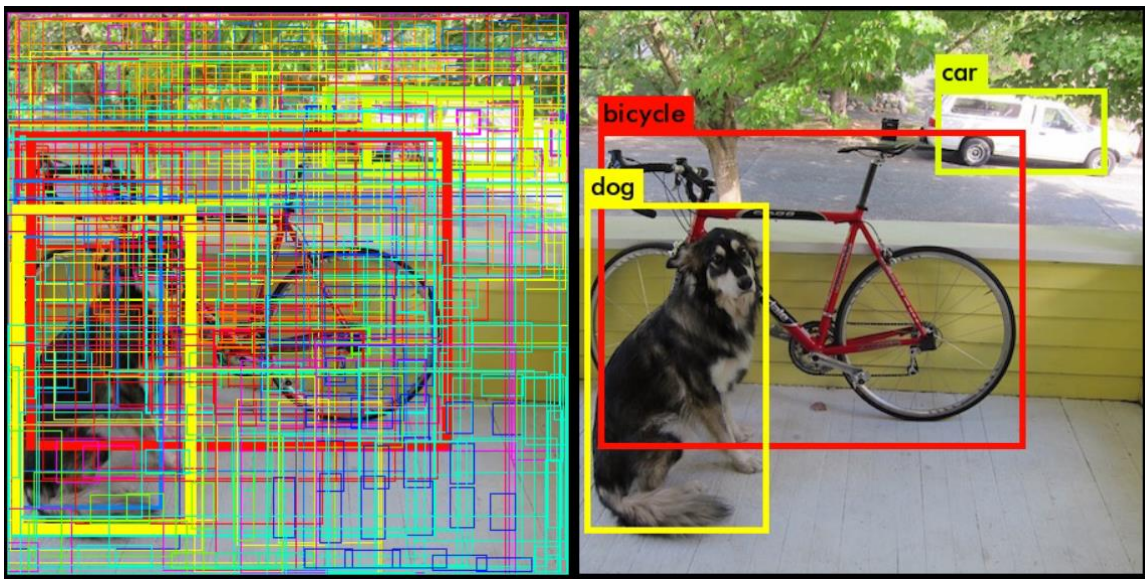
Як видно на рис. 6.17, навіть за дуже сильного стиснення зображення у майже 400 разів на вихідному зображенні можна розпізнати окремі плями, які вже можуть мати певну цінну інформацію. Вочевидь, цю ж інформацію містить і «гирло», але аналізувати матриці з 72 елементами набагато легше, швидше і дешевше, аніж – матриці-тензори 96x96x3. Регулюючи ступінь стиснення можна добитись візуально достатньо гарного відтворення, але воно буде теж меншого розміру, аніж оригінал.

Також, CAE можна використовувати для визначення вмісту вологи на полях за даними дистанційного зондування Землі, пошуку інших патернів тощо.

6.2.4 Аналіз відео. YOLO

YOLO (англ. «You Only Look Once») – це інтелектуальна технологія розпізнавання об'єктів у реальному часі, особливо є ефективним щодо аналізу потокового відео з відеокамер та ін.

Основний принцип YOLO полягає в тому, щоб на кожному зображенні одразу в одній моделі прогнозувати і прямокутники (bounding boxes) з певними об’єктами, і ймовірності віднесення цих об'єктів до певного класу. Ідея YOLO полягає в тому, що мережа розділяє зображення на сітку (grid), і для кожної комірки цієї сітки прогнозує bounding boxes та ймовірності класів, які порівнюються з певним порогом. На рис. 6.18 наведено приклад розпізнавання об’єктів з порогом 0,3.



а) б)

Рисунок 6.18 – Приклад розпізнавання об’єктів з порогом 0,3: а) усі можливі потенційні об’єкти (ширина кольорової рамки пропорційна ймовірності правильної ідентифікації); б) тільки найкращі об’єкти з точністю ідентифікації не нижче 0,3

Основні етапи роботи технології YOLO:

1. Розділення зображення на сітку: зображення розбивається на сітку, і кожна комірка цієї сітки відповідає певній області зображення.

2. Прогнозування bounding boxes та ймовірностей класів: для кожної комірки сітки модель прогнозує bounding boxes та ймовірності класів.

3. Фільтрація та об'єднання bounding boxes: застосовується фільтрація для відсіювання менш важливих bounding boxes. Також може використовуватися метод об'єднання («non-maximum suppression»), який допомагає уникнути дублювання об'єктів у випадку, коли модель виявляє один об'єкт декілька разів.

Наразі, це – один з основних алгоритмів розпізнавання об’єктів у потоковому відео, на відеокамерах тощо. На рис. 6.19 наведено приклад розпізнавання зображення на вулицях міста.

Рисунок 6.19 – Приклад роботи YOLO у місті

Ключовими перевагами YOLO є швидкодія та робота одразу з усім зображенням та багатьма класами об’єктів з використанням передтренованих моделей. Це є й недоліками технології – низька точність у порівнянні з нейромережевими моделями, які більш старанно аналізують кожне зображення, та обмеженість кількості і видів класів, які розпізнаються.

У [38,39,40] є цікавий огляд відмінностей версій YOLO, починаючи з першої версії у 2015 р., які можна звести до такого:

- YOLO2 або YOLO9000 (2016): більше об'єктів (більше 9000 категорій), більша швидкодія і точність архітектура Darknet-19 на основі VGGNet; пакетна нормалізація, навчання моделі на зображеннях у кількох масштабах; нова метрика;

- YOLO3 (2018): архітектура Darknet-53 на основі ResNet; опорні блоки з різними масштабами та пропорціями; концепція «Feature Pyramid Networks (з англ. «мереж пірамід функцій») (FPN), що покращує ефективність виявлення невеликих об’єктів за рахунок використання кількох масштабів; краще виявлення об'єктів з різними розмірами і співвідношеннями сторін; метрика «GHM loss»;

- YOLO4 (червень 2020): архітектура CNN під назвою CSPNet (Cross Stage Partial Network) на основі ResNet; кластеризація kmeans для генерування блоків прив’язки з різними масштабами і пропорціями; метод Mosaic для змішушування чотирьх різних навчальних зображень, щоб надати моделі кращу контекстну інформацію; метрика «GHM loss»;

- YOLO5 (2020): архітектура EfficientDet на основі EfficientNet для більшої кількості категорій об’єктів; новий метод для генерації блоків прив’язки "Dynamic Anchor Boxes" (з англ. «динамічні блоки прив’язки»); "Spatial Pyramid Pooling" (з англ. «об’єднання просторових пірамід») (SPP) для виявлення невеликих об’єктів та об'єктів у різних масштабах; метрика Complete IoU (CIoU);

- YOLO6 (2022): архітектура EfficientNet-L2 з меншою кількістю параметрів і вищою обчислювальною ефективністю одночасно; новий метод для генерації блоків прив’язки "Dense Anchor Boxes" (з англ. «щільні блоки прив’язки»);

- YOLO7 (липень 2022): удосконалений метод генерації блоків прив’язки з різними пропорціями для більш точного виявлення об'єктів у ширшому діапазоні форм і розмірів, розширена ефективна мережа агрегування рівнів (E-ELAN) та ін.;

- YOLO8 (січень 2023)): багато різних алгоритмів із класифікації зображень, виявлення об’єктів без прив’язки, сегментації екземплярів тощо. Зокрема, прогнозується центр об'єкта, замість передбачень обмежувальної рамки, що прискорює подальше оброблення з усуненням дублікатів; метрики CIoU і Distribution Focal Loss (DFL).

Ще у 2021 р. була розроблена модель YOLOX, яка перевищує показники і YOLO4, і YOLO5. Вона використовує архітектуру з увагою, яка дозволяє їй краще фокусуватися на важливих деталях зображення; має більш ефективний алгоритм оптимізації, який дозволяє їй швидше навчатися; підтримує відео та мультикласове виявлення.

Зазвичай, є 5 варіантів YOLO, залежно від кількості параметрів: «нано» (n), малий (s), середній (m), великий (l) і надзвичайно великий (x).

Більше детально переваги YOLO8 перед YOLO5 див. на рис. 6.20.

Performance Comparison of YOLOv8 vs YOLOv5

Model Size	Detection*	Segmentation*	Classification*
Nano	+33.21%	+32.97%	+3.10%
Small	+20.05%	+18.62%	+1.12%
Medium	+10.57%	+10.89%	+0.66%
Large	+7.96%	+6.73%	0.00%
Xtra Large	+6.31%	+5.33%	-0.76%

*Image Size = 640 *Image Size = 224

Рисунок 6.20 – Порівняння ефективності роботи технологій YOLO8 та YOLO5

На рис. 6.21 наведена інфографіка операцій, згаданих у підрозділах 6.1 та 6.2, у нотації рис. 1.2.

Рисунок 6.21 – Інфографіка інтелектуальних технологій з передоброблення та класифікації зображень та відео

6.3 Генерування і детектування діпфейків. GAN, VAE, дифузійні моделі

Одним із найновіших напрямів застосування нейромережевих технологій є генерування та детектування зображень та відео на їх основі.

Зображення та відео, згенеровані з використанням глибоких нейронних мереж, прийнято називати діпфейками (англ. «DeepFake»). Останнім часом, цей термін поширюють вже не тільки на повністю згенеровані зразки, а – й на дещо змінені реальні зображення чи відео: замінено і припасовано обличчя, голос, звуки, фон, будь-який елемент зображення додано, змінено чи видалено.

Поряд із прикладами використання діпфейків з метою шахрайства і дезінформації, є й корисні застосування:

1) у фільмах, анімаційних фільмах, рекламних роліках, музичних кліпах та в іншій розважальній індустрії;

2) для створення освітнього контенту, у т.ч. для різного роду тренажерів;

3) для імітації реалістичних сценаріїв розвитку подій, у т.ч. за участю людей;

4) реконструювання зображень з стародавніх чи пошкоджених документів;

5) генерування заданої інформації у зручній формі для людей з обмеженими можливостями;

6) для генерування віртуальних помічників та у сфері надання різних послуг;

7) для примірювання одягу, зачісок, макіяжу тощо;

8) як аватарки у соцмережах;

9) створення зразків для тестування моделей і технологій, у т.ч. детекторів діпфейків та ін.

Але основною цінністю технологій генерування і детектування діпфейків є не стільки самоціль, тобто вміння згенерувати реалістичний фейк чи вміння його детектувати, а – технології, які розробляються для цього. Це – як змагання «Формула – 1» чи космічні програми. Цінним є не стільки перемогти у змаганні чи висадитись на Місяць або багато разів вивести вантаж на орбіту, скільки створити для цього нові технології, які потім, з часом, трансформуються у звичні для нас у повсякденному житті.

Завдяки зусиллям, спрямованим на розвиток інтелектуальних інформаційних технологій для задачі генерування і детектування діпфейків, постійно удосконалюються:

- інтелектуальні моделі передбачення;
- методи кластеризації;
- технології видалення шумів на зображенні, відео, аудіо-файлах;
- відновлення пошкоджених відео;
- автоматичного розфарбовування чорно-білих відео;

- покращення роздільної здатності зображень і відео;
- архівування файлів;
- трансформація файлів з одного формату в інший;
- генерування зображень і відео за текстовим описом чи, навпаки – генерувати текст по зображеннях та відео (наприклад, див. Kaggle-конкурс «Stable Diffusion - Image to Prompts» якраз з такої тематики) тощо.

На рис. 6.22 наведено ряд зображень, згенерованих у Kaggle одним із авторів посібника з використанням моделі «StyleGAN2-ADA».

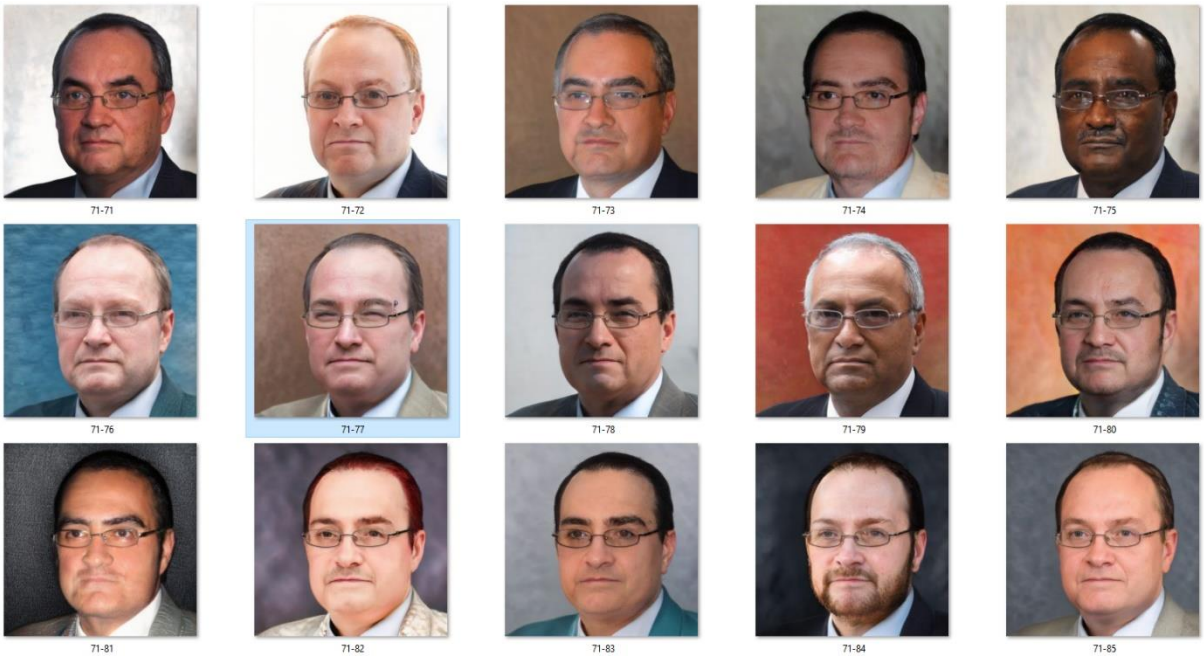


Рис. 6.22. Діпфейки, згенеровані з використанням моделі «StyleGAN2-ADA» (див. «StyleGAN2-ADA» на GitHub та Kaggle ноутбук для застосування цієї моделі), змінюючи різні параметри стилів для заданого фото

Є багато різних способів генерування діпфейків, але серед них варто виділити 3 найкращі види моделей та технологій, принцип роботи яких схожий на роботу автоенкодерів (див. п. 6.2.3):

1. *GAN* (англ. «Generative Adversarial Network») – *генеративні змагальні мережі*. Кожна модель має такі складові: генератор, який з шуму, розподіленого за нормальним законом, генерує діпфейк заданого типу, та дискрімінатор (діпфейк-детектор), який перевіряє чи це – фейк чи ні. Генератор навчається та щоразу удосконалюється таким чином, щоб «перехитрити» дискрімінатор. Результатом є діпфейк, який дискрімінатор не вважає фейком, чи ймовірність того, що це – фейк, є максимально низькою. Це – змагання двох потужних нейронних мереж, звідси – назва. Відповідно: реалістичність фейків – дуже висока, як і необхідні обчислювальні ресурси та тривалість роботи, у порівнянні з іншими підходами.

Однією з головних проблем GAN є низька різноманітність результатів. Якщо добиватись більшої різноманітності, тоді – падає точність (реалістичність). Крім того, не дивлячись на високу візуальну точність, більш прискіпливий аналіз збільшених фрагментів зображення дозволяє виявити патерни («узори») певного виду, якщо знати де і що шукати. Це – один зі способів, який використовують діпфейк-детектори, які, часто, не тільки визначають, що це – діпфейк, а й визначають якою саме GAN-моделлю він був згенерований!

Найбільш вдалою вважається модель StyleGAN2, яка не стільки генерує нове зображення, скільки зосереджується на зміні певних його стилів (емоції на обличчі, фон, освітлення, колір шкіри, елементи обличчя тощо) (див. рис. 6.22). Більш детально див. у статті [37].

2. *VAE* (англ. «Variational Autoencoder») – *варіаційний автоенкодер*. Принцип дій – такий самий, як і в згорткових автоенкодерах (див. п. 6.2.3), але в гирлі – не просто числова багатовимірна матриця, а – прихований ймовірнісний розподіл. Енкодер (кодувальник) VAE з стискає багато зображень до однієї й тієї самої множини багатовимірних ймовірнісних розподілів, які є їх аналогом у певному сенсі. А потім декодер VAE випадково вибирає з цього розподілу ряд параметрів і з них генерує вихідне зображення. При цьому, також, можливі деякі складнощі, але ряд регуляризаційних обмежень дозволяє усунути більшість із них. Більше детально його робота описана у статті [37]. Однією з переваг є висока різноманітність зображень. Алгоритм побудови дозволяє врахувати особливості кожного зображення у тренувальній вибірці і потім використати їх на етапі генерування. В силу архітектури, VAE має тільки одну нейронну мережу, яка навчається швидше, ніж 2 нейронні мережі GAN, а тому моделі на основі VAE працюють швидше, хоча, в загальному випадку, це залежить від багатьох умов.

3. *Дифузійні моделі* (англ. «Diffusion models»). Дифузійні моделі основані на формалізації процесу генерування у вигляді марківського ланцюга (англ. «Markov Chain») – це математична модель стохастичного процесу, де майбутній стан системи залежить тільки від поточного стану, без урахування попередніх її станів. Задане зображення проходить серію поетапних трансформації шляхом додавання на кожній стадії гауссівського шуму аж до тих пір, поки воно не перетвориться на білий шум. А потім оригінальне зображення відтворюється з цього білого шуму у зворотному порядку. На кожному етапі перевіряється збіг і, відповідно, удосконалюється нейромережа яка здійснює це відтворення. Натренована у такий спосіб модель здатна згенерувати доволі реалістичні зображення з простого білого шуму. Як правило, з кожною ітерацією вона додає різні деталі і зображення набуває все більшої чіткості. Забезпечує високу реалістичність, але дещо довго навчається. Хоча, останнім часом, з'явились деякі прийому прискорення навчання, однак, все одно, вважається, що GAN та VAE працюють швидше. Більше детально див. у статті.

Окрім генерування зображень, окремо варто виділити й генерування відео. Це – не просто послідовність зображень. Під час генерування діпфейків слід припасовувати рухи елементів зображень, щоб вони були більш реалістичними. Детектори діпфейк-відео, зазвичай, аналізують динаміку рухів на предмет реалістичності поведінки, у т.ч. з боку законів фізики та анатомічних обмежень. Рухи рук, ніг, особливо людей, мають чимало обмежень. Багато відомих помилок пов'язано з предметами, які перетинають частини тіла людей, наприклад рука, яка проходить крізь мікрофон чи – вглиб поверхні стола, одразу видають діпфейк. Також, аналізується як людина говорить і чи дійсно звук відповідає тому, що людина говорить (найбільш популярний і простий спосіб діпфейка – замінити аудіоряд відео на інший). Відповідно, генератори відео це враховують. Як правило, вдалі діпфейк-відео – це доволі короткі відео, де у справжньому відео заміняється обличчя на іншу людину. При цьому, людина повинна дивитись у камеру, а не стояти боком, бути без окулярів, бороди чи пишної зачіски (припасовування волос на обличчі є значною проблемою), майже не рухатись. Тоді, навіть, з використанням легко доступних моделей з GitHub, можна згенерувати доволі реалістичні діпфейки високої якості.

У 2020 р. у Kaggle проводився конкурс «Deepfake Detection Challenge», де треба було детектувати діпфейк-відео. Рекомендуємо почитати хід розв'язання задачі його переможцями, які посіли 1 місце (моделі EfficientNets), 3 місце (3 моделі EfficientNet-B7 з тривимірною згорткою) та 5 місце (модель SE-ResNeXT50 і різні архітектури 3D CNN).

6.4 Інтелектуальні інформаційні технології аналізу та генерування зображень і відео

Технології інтелектуальних інформаційних технологій аналізу та генерування (синтезу) зображень дозволяють вирішувати багато прикладних задач.

Наведемо короткий огляд конкурсів Kaggle з цієї тематики з реалістичними постановками задач:

- аналіз КТ-томографії, рентгеноскопія, аналіз УЗВ-знімків тощо в медичних цілях;
- розпізнавання зображень та перетворення в інформацію, зручну для сприйняття для сліпих та інших людей з обмеженими можливостями;
- розпізнавання графічної інформації та перетворення її в табличний чи описовий вигляд;
- реконструювання зображення (написів на старовинних сувоях та ін.) за фрагментами зі сканів, отриманих у різний спосіб;

У конкурсі RSNA 2023 Abdominal Trauma Detection слід було розробити моделі штучного інтелекту, які допоможуть у точному та швидкому виявленні та оцінці важкості травм внутрішніх органів черевної порожнини

на зображеннях комп'ютерної томографії (CT). Зокрема, учасники мали розробити алгоритми, які зможуть виявляти пошкодження та внутрішні кровотечі в органах, таких як печінка, нирки, селезінка та кишечник, забезпечивши медичним працівникам важливу інформацію для ефективного та швидкого лікування травмованих пацієнтів. Перше місце посіло рішення на основі моделей GRU та CNN. Дані попередньо оброблялися, включаючи масштабування DICOM-зображень та використання віконного відображення м'якіших тканин. Далі вони застосовували 3D сегментаційну модель для отримання масок для кожного зрізу томограми та вирізали області органів. Використовуючи отримані маски, дані оброблялися в 2.5D форматі для навчання моделей, що включали в себе 2D CNN та RNN, що зображено на рисунку 6.23.

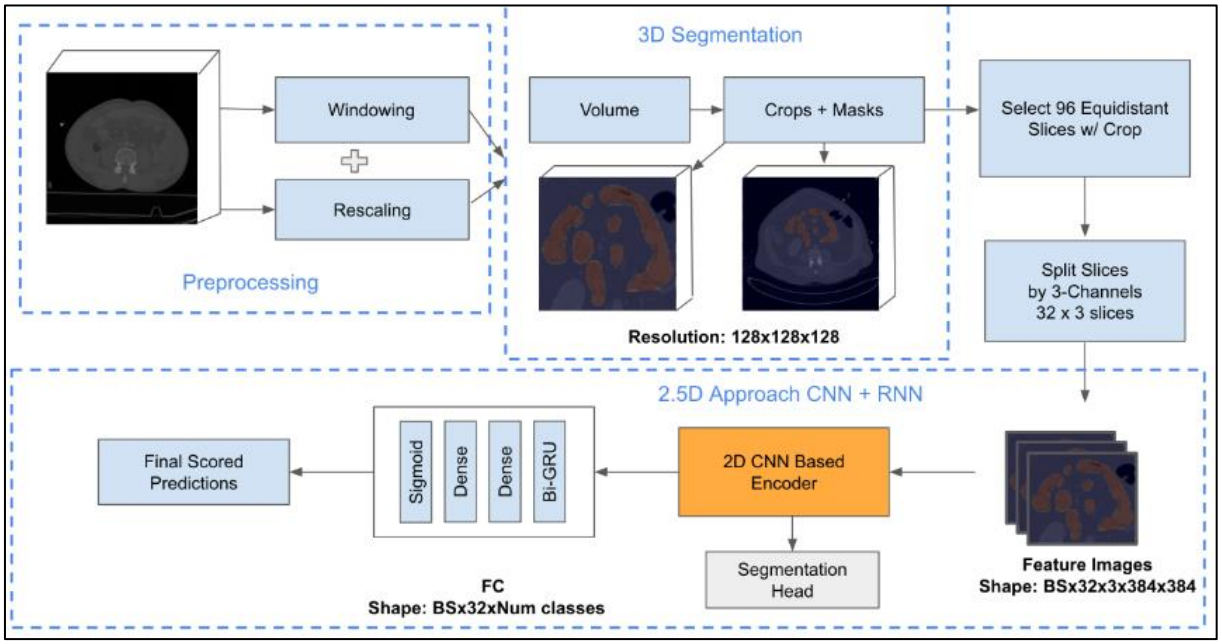


Рисунок 6.23 – Перший етап рішення

У другому етапі, моделі навчалися використовуючи підходи з використанням віконного відображення та масок, згенерованих у попередньому етапі. Для агрегації прогнозів використовувалася максимальна агрегація. Для покращення результатів також використовувалися додаткові втрати на основі сегментації та архітектури нейромереж. Наприкінці використовувалася ансамбль моделей, агрегуючи прогнози на рівні зрізів та використовуючи максимальне значення для кожного пацієнта , що зображено на рисунку 6.24.

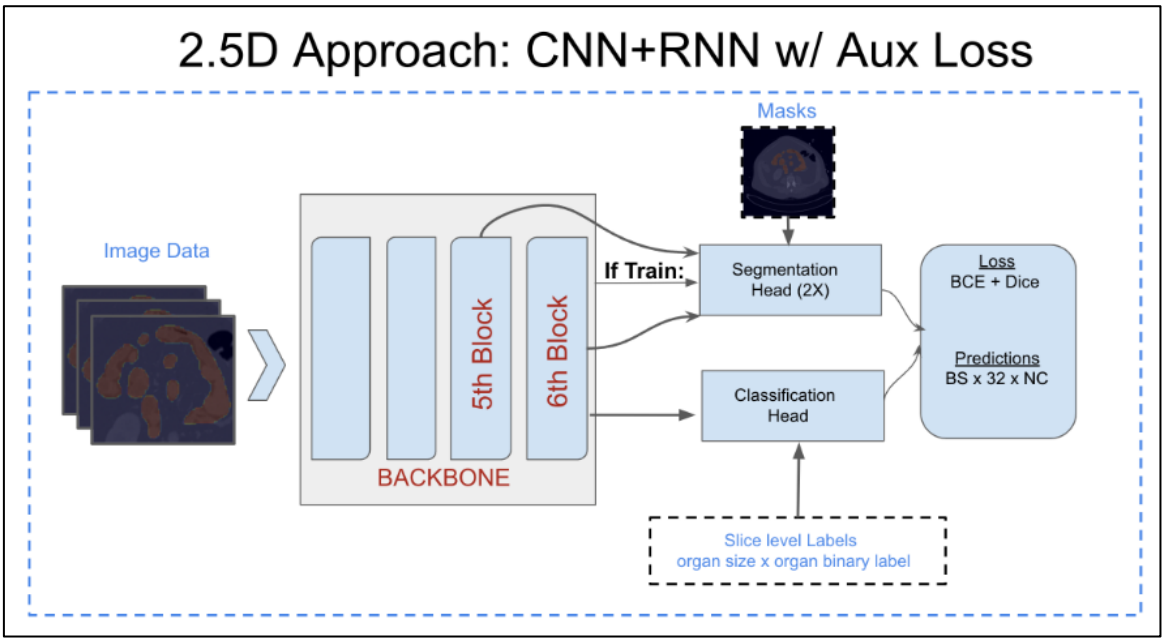


Рисунок 6.24 – Другий етап рішення

У конкурсі Benetech - Making Graphs Accessible слід було автоматично розпізнати вид графіка на зображеннях у сфері STEM. Конкурс орієнтований на розширенні доступу до знань людей з обмеженими можливостями, але така постановка задачі може бути цікавою й в інших сферах, адже часто інформація доступна лише у графічній формі і її слід розпізнати та проаналізувати. Перше місце зайняла команда з доволі цікавим рішенням.

Етап 1: Класифікація типів графіків. Використовуючи дві моделі - convnext_large_384 та swin_large_patch4_window12_384 - вони класифікували графіки індивідуально, а потім об'єднали результати за допомогою зваженого ансамблю, що зображено на рисунку 6.25.

Етап 2: Прогнозування даних серій. Залежно від типу графіка, застосовувалися різні методи. Для Bar, Line та Dot використовувався підхід Deplot, тоді як для Scatter використовувалася об'єктно-орієнтована модель детекції.

Крім того, для підвищення точності моделей використовувалися додаткові набори даних, а також були проведені різні етапи навчання з метою покращення результатів. Наприкінці, команда здійснила ансамблювання моделей для кожного типу графіка для отримання кінцевих прогнозів.

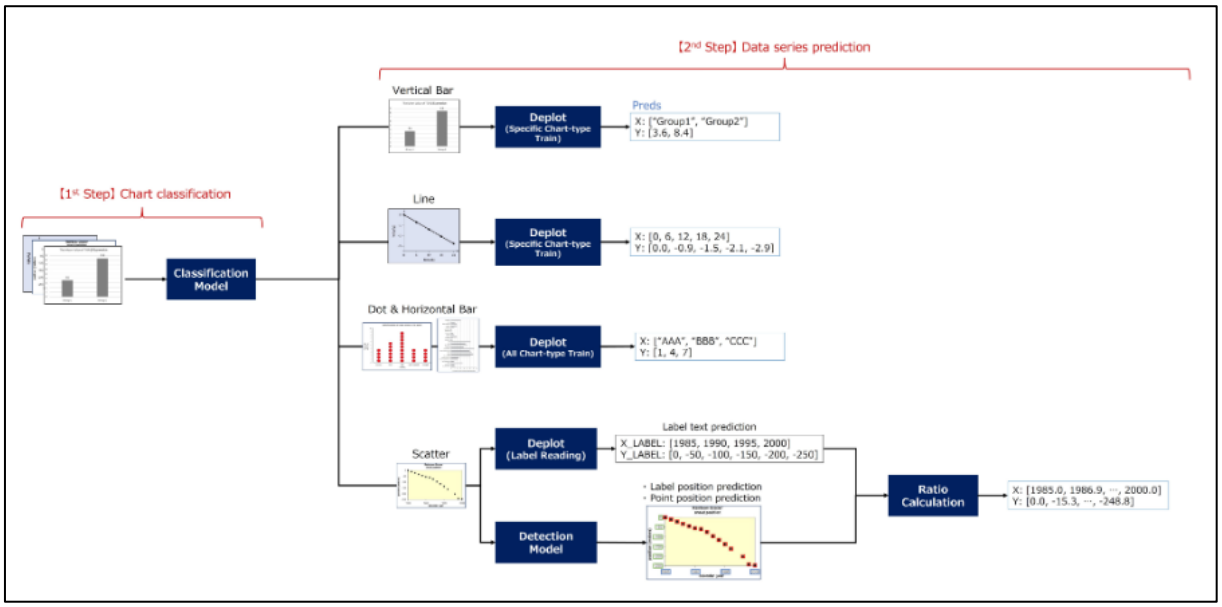


Рисунок 6.25 – Етапи рішенням конкурсу Benetech - Making Graphs Accessible

У конкурсі Vesuvius Challenge - Ink Detection слід було прочитати напис на сувої при тому, що чорнила не видно. Їх наявність можна відновити тільки по фрагментах сканування у різних діапазонах та за даними, зібраними в інший спосіб. Таке дослідження є цінним, оскільки часто буває, коли потрібне зображення не видно повністю, а його можна тільки реконструювати за даними з різних джерел. Перше місце зайняла команда що розробили комплексний підхід до сегментації тексту на зображеннях, поєднуючи використання більших областей зображень з моделями, стійкими до глибини тексту. Вони створили низку моделей, які ефективно працювали з різними глибинами тексту та змогли підвищити точність шляхом ансамблювання прогнозів. Тренуючи моделі на всій доступній інформації та застосовуючи різноманітні методи аугментації, вони досягли стабільного покращення результатів. Крім того, учасники систематично оцінювали та післяобробляли прогнози, щоб забезпечити оптимальну калібрування та точність моделей.

Виявленню раку молочної залози за даними регулярного скринінгу мамографії (по 4 мамограми на пацієнта) був присвячений конкурс «RSNA Screening Mammography Breast Cancer Detection». Усі переможці (принаймні, перші 3 місця) використовували згадану у п. 6.2.4 модель YOLOX на етапі передоброблення. Потім використали ряд прийомів, у т.ч. аугментацію датасету з використанням HorizontalFlip, VerticalFlip, RandomBrightnessContrast, ShiftScaleRotate, MedianBlur, GaussianBlur, GaussNoise, ElasticTransform, GridDistortion, OpticalDistortion, CoarseDropout, Mixup та ін., а потім створили ряд ансамблів. Переможець (1 місце) використовував моделі Eff-B2, Eff-B4, Effv2-si на основі «EfficientNet» («Eff»), згаданій у п. 6.2.2, а також – Convnextv1-small, та досліджував їх різні особливості, результати аналізу з графіками описав у сво-

єму пості. Рішення 2-го місце основано на моделі Faster R-CNN, теж згадувана у п. 6.2.2. Автори рішення 3-го місця використали ансамбль моделей на основі моделі LSTM, про які буде більш детально написано у розд. 8.

Крім того, є чимало інших прикладів розв'язання задач з використанням інтелектуальних інформаційних технологій аналізу та генерування зображень.

Дуже цікавим був конкурс «Stable Diffusion - Image to Prompts», де учасникам пропонувалось створити модель, яка зможе по зображенню відновити промпт, за яким було створено це зображення. Така модель дозволила б покращити і саму технологію генерування зображень, і зробити його більш передбачуваним, і навчитись краще виявляти зображення, згенеровані саме з використанням Stable Diffusion. Переможець досягнув точності лише 0.66. Рекомендуємо почитати його пост про те, як він цього досяг – дуже повчально: і гіпотези, і як він збирав інформацію, у т.ч. за допомогою ChatGPT, як з датасету COYO-700M з 747 млн. картинок з їх промптами відбирав найкращі пари, як тренував різні моделі. Він описав весь процес свого наполегливого дослідження. Не менш цікавими є й пост призера 2-го місця та пост призера 3-го місця учасника – вони використовували дещо різні підходи та датасети.

На рис. 6.26 наведена інфографіка інструментарію, згаданого у розділі 6 в цілому, у нотації рис. 1.2.

Рисунок 6.26 – Інфографіка інтелектуальних технологій з передоброблення, класифікації та генерування зображень і відео

Можливі теми практичних і лабораторних завдань

Тема № 1. Аналіз та класифікація тексту природною мовою з використанням технологій глибокого навчання та NLP (або **«Інтелектуальний аналіз графічних даних про стан складної системи з використанням технологій штучного інтелекту на Python»**).

Метою роботи є вивчення інформаційних технологій і Python-бібліотек для аналізу та класифікації тексту природною мовою (NLP) з використанням технологій глибинного навчання й опанування практичних навичок застосування деяких із них на прикладі одного із датасетів Kaggle чи за даними, завантаженими через API.

План заняття:

1. Вибрати датасет (наприклад датасет «NLP : Reports & News Classification»):

2. Завантажити дані (якщо вони – кирилицею, тоді – правильно налаштувати кодування). Відібрати дані для заданої у варіанті бригади цільової ознаки («target»).

3. Підключити модель одну з передтренованих BERT-моделей з бібліотеки «Hugging Face», наприклад 'distilbert-base-multilingual-cased' («multilingual» - працює саме з кирилицею, як і з іншими 104 мовами світу) та правильно адаптувати до неї дані (токени, маски тощо).

4. Натренувати модель для післяоброблення даних – для класифікації за навчальними даними виходу BERT-моделі, тобто даних ембедингів, на які вхідний текст перетворює модель з п.3. Такою моделлю може бути й дуже складна модель – логістична регресії, дерево рішень чи нейронна мережа з 1 чи 2 шарами. Налаштувати цю модель.

5. Зробити передбачення даних на тестовому датасеті та порівняти із заданими. Оцінити точність за accuracy_score для випадку, коли у тестовому датасеті опиняється 40% від вхідних даних. Спробувати оцінити як змінюється точність за розміру тестових даних у 10%, 20%, 30% від загальної кількості (решта – навчальні дані).

6. Здійснити візуалізацію отриманих результатів.

7. Навести посилання на створений ноутбук, який містить усі результати роботи (п. 2-6 та коментарі англійською мовою щодо того, яка модель - оптимальна).

Приклади ноутбуків із розв'язання задач класифікації текстових даних з використанням NLP-технологій:

- Data Science with DL & NLP: Advanced Techniques
- NLP - EDA, Bag of Words, TF IDF, GloVe, BERT
- NLP with Disaster Tweets - EDA and Cleaning data
- NLP for UA : BERT Classification for Water Report

Контрольні питання

1) Що таке тензори і як вони використовуються в обробці зображень?
2) Які стандартні датасети із зображеннями використовуються для навчання моделей?
3) Які типові постановки задач існують у відношенні до обробки зображень?
4) Які операції передоброблення зображень можна виконати за допомогою бібліотеки OpenCV?
5) Що таке згорткові нейромережі, і як вони використовуються для аналізу зображень?
6) Які складні архітектури згорткових нейромереж ви знаєте і як вони відрізняються від базових моделей?
7) Як використовуються автоенкодери у задачах без вчителя, особливо у відношенні до обробки зображень?
8) Що таке YOLO і як він використовується для аналізу відео?
9) Що таке генеративно-противні мережі (GAN) і варіаційні автоенкодери (VAE), і як вони використовуються для генерації та детектування deepfakes?

10) Які інтелектуальні інформаційні технології використовуються для аналізу і генерації зображень і відео?

7 ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ТА ОБРОБЛЕННЯ ПРИРОДНОМОВНОГО ТЕКСТУ

7.1 Основні поняття, види задач, збирання та передоброблення даних

Оброблення природної мови (англ. «Natural Language Processing» – «NLP») – один із найбільш популярних в наш час вид задач інтелектуального аналізу даних.

Технології NLP стосуються як тексту (будь-якими мовами), так і мови, тобто – звукових сигналів. Але найбільше поширення останнім часом отримало застосування NLP саме для оброблення природномовного тексту.

Корпус – це великий датасет (колекція) текстів з якоїсь теми.

NLP вирішує наступні *задачі*:

- переклад з однієї мови на іншу (як аудіо-, так і текстових даних);
- аналіз настроїв людей (сентиментний аналіз – англ. «Sentiment Analysis»);
- пошук чи видобування інформації, у т.ч. *веб-скрапинг* (завантаження сторінок і вилучення з неї блоків корисного тексту) та *парсинг* (формалізація і структурування тексту за певними критеріями і шаблонами та приведення до заданого формату, наприклад до CSV чи JSON);
- узагальнення даних («анотування» – англ. «Summarization») – автоматичне написання анотації чи стиснення тексту до заданого розміру;
- заповнення пропущеного тексту у реченнях чи продовження набору речення, наприклад в СМС-ках на смартфонах чи в інших діалогових системах;
- класифікація тексту, у т.ч. диференціація тексту, написаного людиною, від тексту, написаного з використанням машинного навчання технологій NLP;
- фільтрування спаму на пошті чи у соцмережах;
- розпізнавання текстового запиту з метою реагування на нього, у т.ч. генерування відповіді (QA-системи, англ. «Quation-Answear») чи генерування зображення за текстовим запитом;
- генерування інформації, у т.ч. в діалоговому режимі, підтримка роботи систем для call-центрів, рекомендаційних систем тощо;
- розпізнавання іменованих сутностей (географічних назв, назв компаній, ПІБ людей) та зв'язків між ними на основі природномовного тексту (англ. «Name Entity Recognition» – скорочено: «NER»);
- пошук і вибір у тексті певних частин речення (іменників, дієслів, прикметників та ін.) чи частин у слові (суфіксів та ін.) тощо.

Більшість із цих задач можна розв'язати з використанням великих мовних моделей та API чат-ботів (наприклад, за допомогою ChatGPT компанії OpenAI чи ін.), яким присвячений підрозділ 7.3. А підрозділ 7.2 буде присвячений більш традиційним моделям, методам і підходам.

Приклад алгоритму NLP для, наприклад, класифікації тексту (див. реалізацію у Kaggle ноутбуці одного з авторів українською та англійською мовами на його ж датасеті, де є результат парсингу у формат CSV тексту його монографії у співавторстві зі шведами, виданої українською та англійською мовами, за фінансування проєкту SIDA (Швеція)):

1. Знайти дані (найкраще – англійською, але можна й іншими 104 мовами, які підтримують більшість великих мовних моделей, у т.ч. українською).

2. Розмітити дані (найпростіше, коли є одна ознака з 0 або 1, але може бути й багато) – NLP найкраще працює тільки для розмічених даних.

3. Очистити дані та здійснити інше передоброблення (про це буде далі).

4. *Токенізувати* текст, розділивши його на індивідуальні слова (чи склади).

5. Вибрати і налаштувати модель та технологію її застосування.

6. Вибрати і налаштувати варіант постоброблення.

7. Проаналізувати «викиди» (англ. «outliers»). У разі виявлення чітких патернів у викидах, тобто певних закономірностей, перейти до пп. 3, 4, 5 або 6. У разі, якщо точність недостатня, а будь-які зміни пп. 3-6 не допомагають, тоді треба або змінити розмітку або збільшити сам датасет.

8. Якщо точність моделі достатня, тоді варто її зберегти, наприклад у формат pickle, що дозволяє згодом її одразу застосовувати як передтреновану, для інтелектуального аналізу інших текстових даних. Наприклад, таку модель можна застосувати для класифікації довільного тексту, наприклад, до тексту сайту Басейнового управління водних ресурсів Південного Бугу. Веб-скрапинг та парсинг у формат CSV однієї сторінки цього сайту виконує інший Kaggle ноутбук автора.

Приклади очищення та передоброблення тексту () (цей етап приведення тексту до стандартної форми ще часто називають *нормалізацією*):

1. Перевести всі символи в нижній регістр (це не варто робити, якщо планується далі застосовувати NER чи інші методи, для яких велика літера несе цінну інформацію).

2. Виправлення орфографічних помилок (наприклад, «будьмо» / «будмо» / «бууууууууудьмо»), хоча сучасні мовні моделі, часто, це вже вміють робьити самі.

3. Видалити знаки пунктуації (не є обов’язковим, оскільки по-перше, це може нести цінну інформацію, по-друге, потужні моделі це вміють ігнорувати, але прості моделі потребують цього етапу).

4. Видалити символи, які не відносяться до природної мови (фрагменти програмного коду, цифри, гіпертекстові посилання, смайлики тощо).

5. Здійснити *лематизацію*, тобто приведення усіх слів до єдиної словникової формі (наприклад, «машина» замість «машиною», «машині», «машинах» тощо).

6. Видалення *стоп-слів* – в англійській мові ними є, наприклад: "the", "is" тощо, які не мають специфічної семантики;

7. Застосувати *стемінг* (англ.: «Stemming»), коли слова зводяться до кореня шляхом видалення змінної частини форми слова, через відкидання непотрібних символів, зазвичай, суфіксів чи закінчень (рис. 7.1).

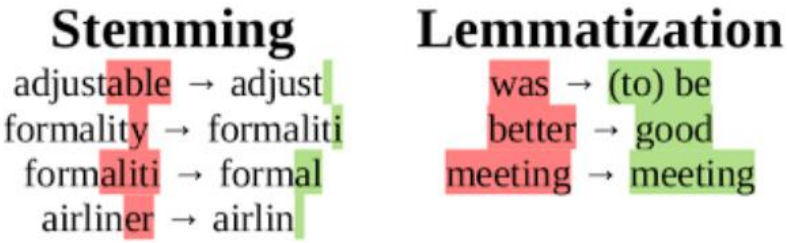


Рис. 7.1 Приклади стемінгу та лематизації з Kaggle ноутбука

8. Застосування *лематизації* – іншого підходу до усунення змінних частин форми слова (чи заміни усього слова на іншу форму) з використанням деталізованої бази даних мови (рис. 7.1).

Приклади коду для застосування багатьох з цих функцій є в авторському Kaggle ноутбуці.

Для автоматизації цих операцій використовуються бібліотеки регулярних виразів re та NLTK (англ. «Natural Language Toolkit»). Більше детально про NLTK див. у роботах [28, 41] та у документації.

7.2 Мовні моделі та їх застосування для розв’язання NLP-задач

Однією з найбільш показових NLP-задач є класифікація природномовного тексту. На ній можна продемонструвати основні технології і підходи, які використовуються для розв’язання й багатьох інших NLP-задач. Тому почнемо з неї. У підрозд. 7.4 н прикладі розв’язання реальних задач та аналізу рішень переможців змагань Kaggle будуть розглянуті особливості розв’язання й деяких інших NLP-задач.

7.2.1 Мішок слів (Bag of Words)

Одним із перших і найбільш популярних методів машинного навчання у сфері NLP був метод класифікації текстів на основі так званого «мішка слів» (англ. «Bag of Words» – скорочено: «BOW» або «BW»). Метод полягає в наступному: робиться список-словник унікальних слів у тексті. Потім кожне речення, абзац чи інша частина тексту подається як вектор, в якому ставиться 0, якщо слово є словнику унікальних слів, і 1 - в протилежному випадку (рис. 7.2). Одним з основних недоліків використання BOW є ігнорування порядку слів та взаємозв'язків між ними, що є дуже важливим для природномовного тексту. Тому цей метод рідко використовується для розв'язання реальних задач.

	good	movie	not	a	did	like
good movie	1	1	0	0	0	0
not a good movie	1	1	1	1	0	0
did not like	0	0	1	0	1	1

Рис. 7.2 Приклад формування векторів чисел замість природномовних речень з використанням методу мішка слів («BOW») з Kaggle ноутбука

7.2.2 TF-IDF

Для збільшення ефективності методу BOW вирішили враховувати частоту появи слів в тексті, щоб фільтрувати загальновживані слова, які є менш цінними для аналізу. Цей метод називається «TF-IDF», оскільки він ранжує слова за показником, який є добутком показників TF і IDF:

- «Частота термінів» (з англ. «Term Frequency» – «TF») – дорівнює відношенню кількості разів появи терміну T у документі до кількості термінів у документі;

- «Інверсія частоти документів» (з англ. «Inverse Document Frequency» – «IDF») – дорівнює log (N/n), де N – кількість документів, а n – кількість документів, в яких з'явився термін T.

Приклад обчислення показників TF і IDF наведено на рис. 7.3.

Sentence A. The car is driven on the road.

Sentence B. The truck is driven on the highway.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

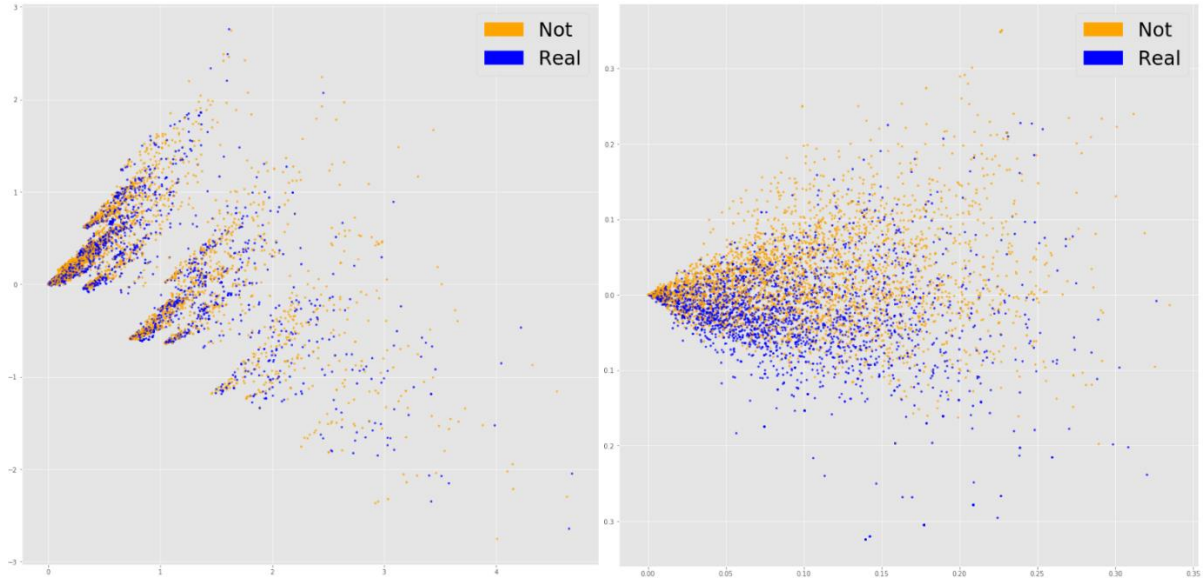
а) б)

Рис. 7.3 Приклад обчислення показників TF і IDF [42]: а) приклади 2-х речень, які аналізуються і утворюють словник з 9 унікальних слів-значень; б) результат розрахунку показників TF і IDF та їх добутку для речень A і B

Як видно на рис. 7.3, за показником «TF*IDF» більш значущими є слова з більшим значенням цього критерію (у малому регістрі): «car», «truck», «road», «highway».

Тренуватись з цієї задачі можна й у Kaggle-конкурсі «Natural Language Processing with Disaster Tweets (Predict which Tweets are about real disasters and which ones are not)», задачею якого є визначення чи стосується заданий твіт якоїсь катастрофи чи ні? Цей конкурс – «вічний», тобто він не має терміну завершення і використовується для тренувань навичок новачків у сфері NLP та для розроблення навчальних матеріалів з цієї тематики. Одним із

авторів цього посібника розроблено ряд таких навчальних ноутбуків. В одному з них (який має більше 500 голосів) [43] наведено приклад застосування різних методів до цієї задачі. Результати трансформовані у двовимірну систему координат з використанням методу зменшення розмірності PCA, що дозволяє візуально порівнювати наскільки добре відрізняються кластер 0 від 1. Варто зауважити, що у багатовимірному просторі, кластери можуть краще розділятись, ніж – у двовимірному. Але певне уявлення ці графіки дають. На рис. 7.4 наведено результат застосування методів BOW та TF-IDF до даних.



а) б)

Рис. 7.4 Приклад 2-вимірної візуалізації результатів передбачення текстових даних Kaggle-конкурсу методами: а) мішок слів (BOW); б) TF-IDF [43]

Метод TF-IDF, не дивлячись на свою простоту, часто використовують, у т.ч. медалісти Kaggle-змагань. Він є ефективним, коли порядок слів має мале значення, а цінним є тільки унікальність слів. Крім того, він є значно швидшим, ніж складніші методи, які використовують нейромережеві моделі зі складною архітектурою.

Як видно на рис. 7.4, метод TF-IDF краще діхотомізує таргет, тобто є більше зон з точками лише одного кольору, але точність все ще є надто низькою. Отже, варто застосовувати кращі методи.

7.2.3 GloVe. Поняття ембеддінгів

В основі більшості більш ефективних NLP методів лежить поняття ембеддінгу (англ. «Embedding» – «Вбудовування»). В NLP *ембеддінги* це – вектори чисел, на які перетворюються токени текстової інформації (як правило, слова чи склади) з урахуванням їхніх семантичного значення та типових взаємозв'язків з іншими токенами у природномовному тексті заданої даної предметної області. На прикладі слів, ембеддінги містять інформацію

про те, в якому контексті воно, зазвичай, вживається. Які слова стоять ближче, які – далі? Які у реченні йдуть до нього? Які – після? Кожною мовою – окремо. Як все це змінюється, в залежності від предметної області? Англійське слово «Bug» може означати і вид (клас, орден) комахи, і помилку у програмному коді, і назву річки і все це буде мати геть різний контекст та типові взаємозв'язки. Так само, слово «Python» – це і вид змій, і мова програмування, і назва шоу («Monty Python's Flying Circus» або «Monty Python»). Як відомо, Гвідо ван Россум – головний розробник мови програмування Python дав їй назву саме на честь цього шоу, а не – змії, хоча у більшості логотипів використовується саме змія, оскільки так зручніше та оригінальніше.

Іноді, вектор методу BOW теж називаються ембеддінгом. В широкому сенсі, це – дійсно вектор чисел, але у вузькому NLP-сенсі, він не є ембеддінгом, оскільки не враховує взаємозв'язки між словами та порядок слів у реченні.

На ембеддінгах основані методи, моделі і технології GloVe, Word2Vec, трансформери, у т.ч. BERT, та ін. Спосіб обчислення ембеддінгів є різних у різних методах.

Найбільш проста для розуміння концепція використовується у методі GloVe (англ. «Global Vectors for Word Representation») – метод глобальних векторів для представлення слів. Береться великих корпус (сотні тисяч, мільйони, навіть, мільярди слів – англомовні тексти Вікіпедії, GitHub та ін.). Слова кодуються номерами та аналізуються попарно. Створюється матриця, де кожен елемент (*i, j*) містить кількість разів, коли слово *j* зустрічається в контексті слова *i*. Потім обчислюється ймовірність цієї появи. А далі формуються вектори, які характеризують ймовірності появи інших слів у контексті заданого. Це – доволі складний оптимізаційний алгоритм, але він демонструє вражаючі результати. Канонічним є приклад операцій зі словами «король» і «королева», наведений, наприклад, у блозі. На рис. 7.5 наведено ембеддінг слова «king» (з англ. «король»), визначений методом GloVe на основі корпусу з 400 тисячами унікальних англомовних слів з Вікіпедії.

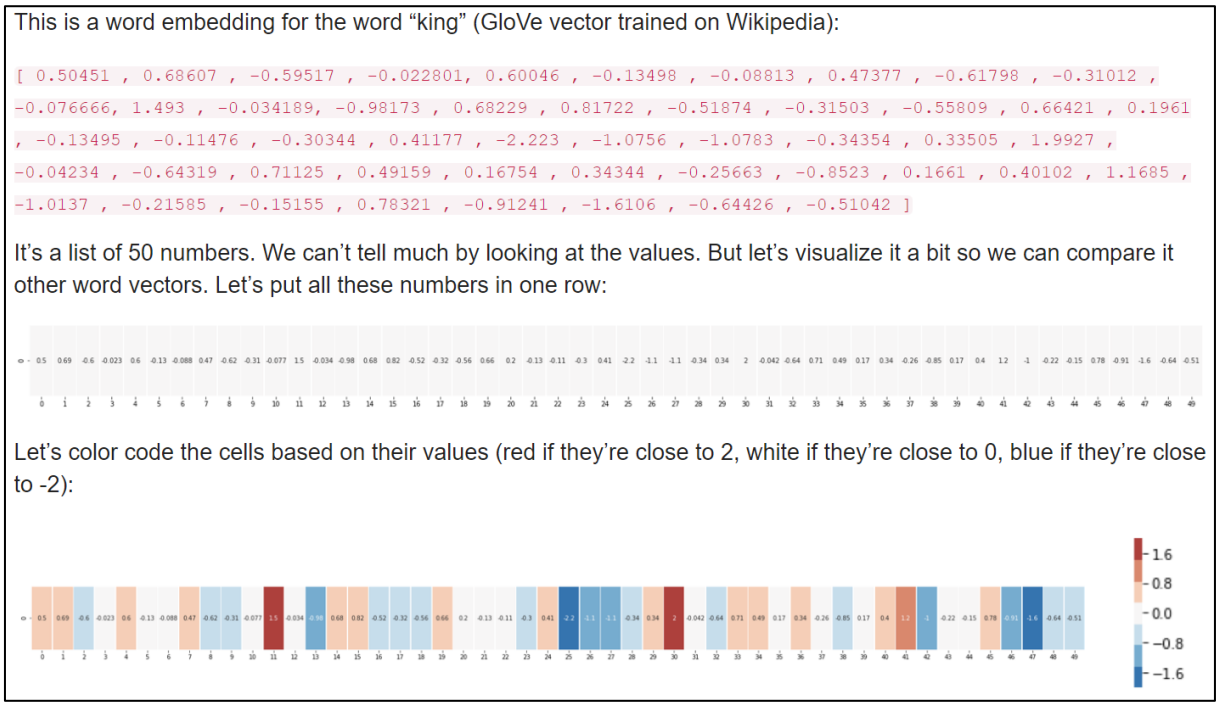


Рис. 7.5 Ембеддінг слова «king» (з англ. «король»), визначений методом GloVe на основі корпусу з 400 тисячами унікальних англомовних слів з Вікіпедії (з блогу)

На рис. 7.6 наведено 8 подібних ембеддінгів.

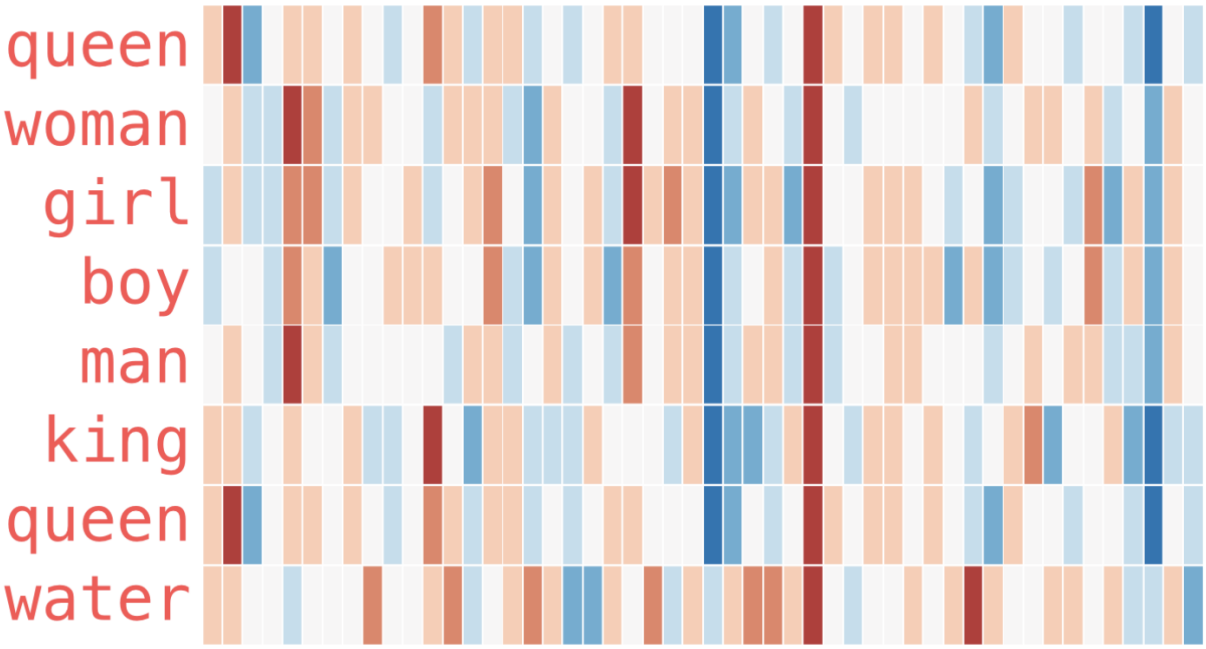


Рис. 7.6 Ембеддінги 8 слів, визначений методом GloVe на основі корпусу з 400 тисячами унікальних англомовних слів з Вікіпедії (з блогу)

Цікаво попарно порівняти на рис. 7.6 людей і не людей («вода»). Осіб чоловічого («король», «чоловік» і «хлопець») і жіночого роду («королева», «жінка» і «дівчина»), «король» і «королева» та ін. Видно, що є певні характерні збіги. Рис. 7.6 ілюструє й важливу особливість ембеддінгів сучасних

NLP-методів: елементи цих ембеддінгів не є інтерпретабельними, тобто не можна вказати які саме значення дали ті чи інші числа, оскільки пораховані у певний спосіб імовірності появи сусідніх слів потім багато разів трансформуються і обробляються. Ця особливість характерна й для більш складних методів. Ембеддінги лише характеризують задані токени (слова чи склади) і дозволяють виконувати з ними певні операції порівняння, додавання, кластеризувати та визначати які більше схожі на інших, аніж – інші. При цьому, не можна порівнювати ембеддінги, визначені різними методами чи ідентифіковані на різних корпусах – це вектори у різних системах координат. Вони не є співставними. Рис. 7.7 демонструє канонічний приклад операцій з ембеддінгами з рис. 7.6.

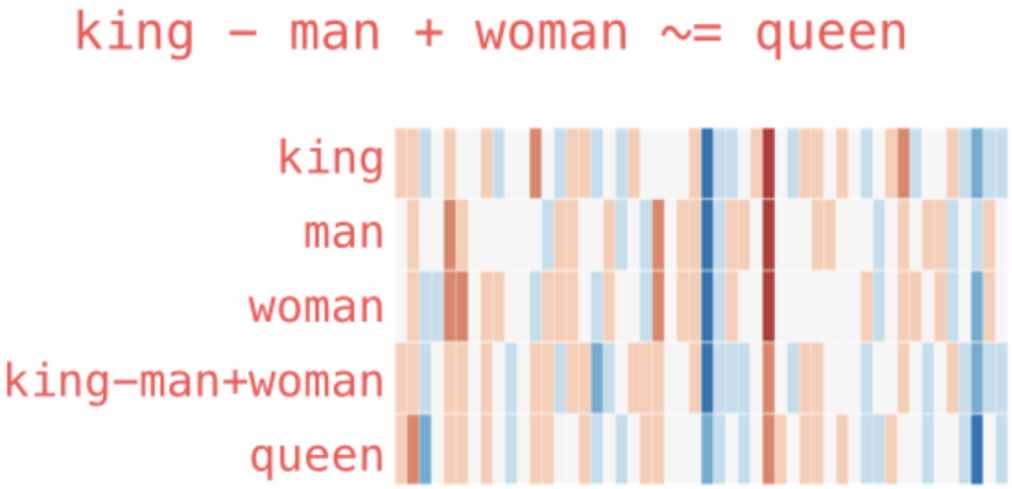


Рис. 7.7 Результат операцій з ембеддінгами з рис. 7.6:
«король» - «чоловік» + «жінка» та порівняння результату
з ембеддінгом слова «королева» (з блогу)

Як видно на рис. 7.7, якщо застосувати поелементне додавання і віднімання векторів з рис. 7.6 за формулою «король» - «чоловік» + «жінка», тоді результат, як стверджується у блозі, буде найбільш близьким до слова «королева» серед усіх 400 тисяч унікальних англомовних слів. І це ще одна особливість усіх методів класифікації машинного навчання – важливо не точно визначити клас – достатньо знайти найбільш схожий з обмеженої вибірки. А отже, в середньостатистичному сенсі результат «король» - «чоловік» + «жінка» можна вважати таким, що приблизно дорівнює (відповідає) слову «королева».

У статті [44] наведено інший приклад: «Paris – France + Italy = Rome».

Метод GloVe є добре масштабується на великі дані, але його ефективність проявляється тільки за дуже великих корпусів, а таке навчання є довготривалим, і це є його одним з основних недоліків.

7.2.4 Word2Vec

Більш оригінальним і швидкодіючим є метод Word2Vec. Він поєднує 2 концепції: передбачення контексту по слову і – слова по контексту (рис. 7.8):

- метод «неперервного» мішка слів (CBOW – з англ. «Continuous Bag-of-Words») передбачає слово по його контексту: нейронна мережа бере кожне слово як таргет та аналізує слова у кожному реченні навколо нього (контекст), намагаючись передбачити це цільове слово;

- скіп-грами (з англ. Skip-grams) передбачають контекст слова за самим словом з використанням його векторного представлення.

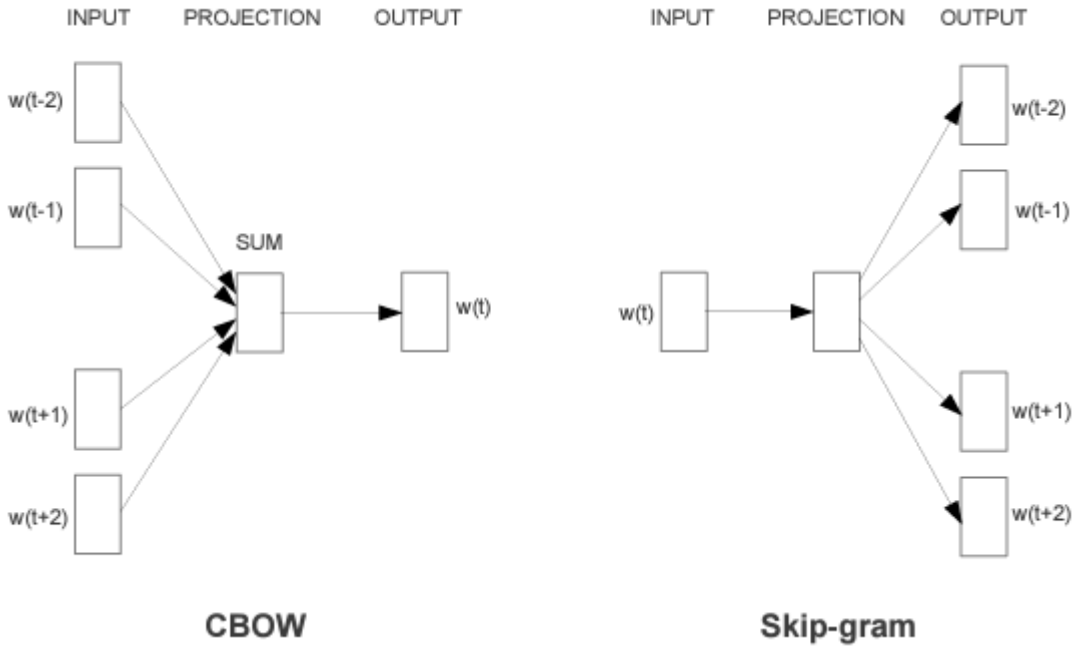


Рис. 7.8 Архітектура моделей методу Word2Vec: CBOW та скіп-грами [44]

Спосіб формування скіп-грам є схожим на спосіб формування ембеддінгів GloVe, але по-перше, в GloVe аналізуються тільки пари слів, а у скіп-грамах – весь сусідніх контекст з багатьох слів, а по-друге, GloVe аналізують весь текст глобально, а скіп-грами – щоразу тільки задане речення чи іншу частину тексту.

Для ознайомлення з процесом навчання моделі Word2Vec (рис. 7.9), у т.ч. скіп-грам та CBOW, рекомендуємо ознайомитись з блогом.

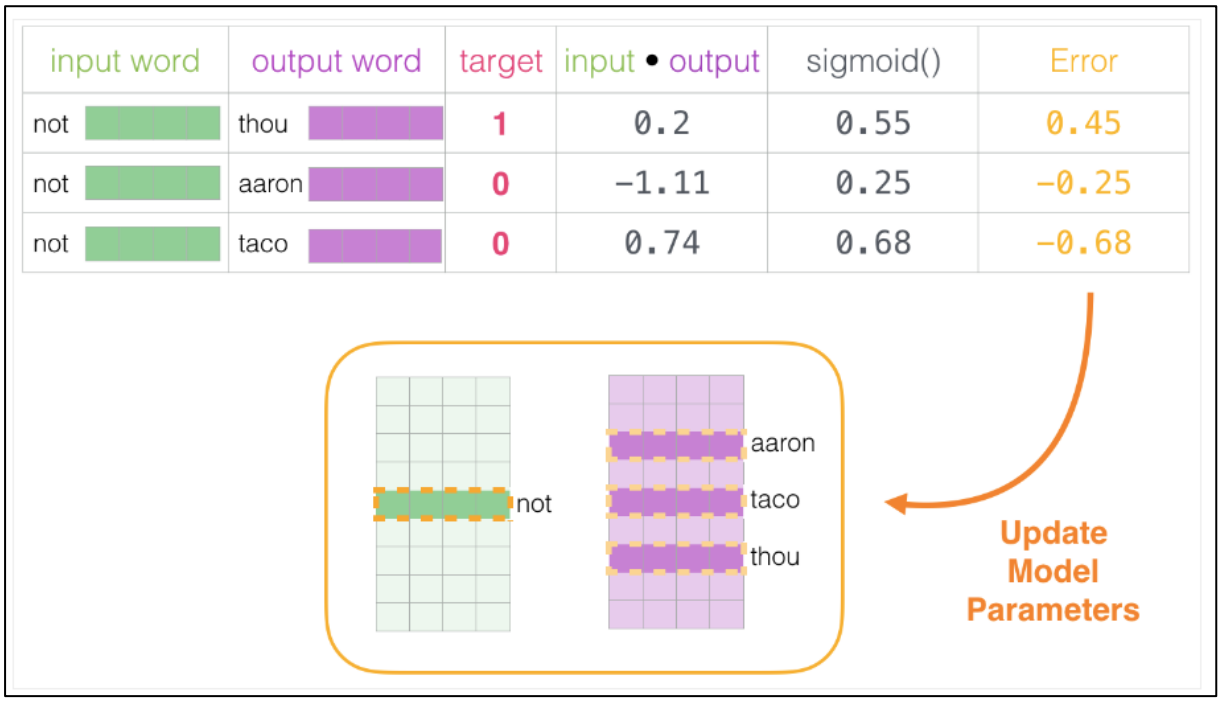


Рис. 7.9 Ілюстрація одного з етапів навчання моделі
за методом Word2Vec у блозі

Також, варто ознайомитись зі статтею [44], де ще у 2013 р. й було запропоновано цей метод (назву методу Word2Vec придумали пізніше). На рис. 7.10 наведено приклад передбачення слів по контексту – слів після двокрапок.

Relationship	Example 1	Example 2	Example 3
Country - Capital	France-Paris	Italy: Rome	Japan: Tokyo
Adjective - Comparative	big-bigger	small: larger	cold: colder
City - State	Miami-Florida	Baltimore: Maryland	Dallas: Texas
Person - Profession	Einstein-scientist	Messi: midfielder	Mozart: violinist
Chemical element - Symbol	copper-Cu	zinc: Zn	gold: Au
Company - Product	Microsoft-Windows	Google: Android	IBM: Linux
Company - CEO	Microsoft-Ballmer	Google: Yahoo	IBM: McNealy
Country - National dish	Japan-sushi	Germany: bratwurst	France: tapas

Рис. 7.10. Приклад зі статті [44] передбачення слова по контексту з використанням Word2Vec, натренованого на корпусі з 783 млн. слів з скіп-грамами з розмірністю 300: наприклад, якщо Relationship – це «Country – Capital», то що писати після «France»?

7.2.5 Моделі-трансформери

Більш сучасним і потужним рішенням є моделі і технологія BERT. BERT (англ. «Bidirectional Encoder Representations from Transformers» – «Двонаправлені кодувальні представлення з трансформерів») використовує одну з найбільш ефективних на даний момент моделей – трансформер (іноді, її називають «трансформатор», але це не стало загально прийнятим).

Двонаправленість означає, що аналізується контекст слова у реченні в обох напрямках: і до, і після нього.

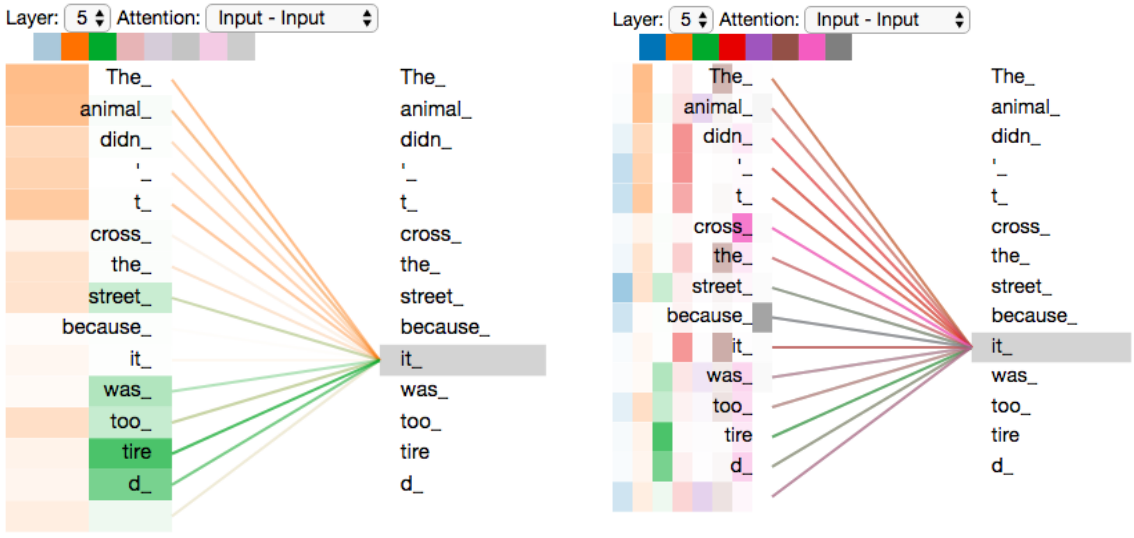
Дуже гарно деталізоване та ілюстроване пояснення механізму роботи трансформерів в NLP наведено у блозі. Можна виділити такі його важливі особливості:

1. Використання архітектури «кодер-декодер» (кодер перетворює (трансформує) текст на векторне представлення, а декодер – навпаки: вектори на текст). Саме такими перетвореннями-трансформаціями пояснюється назва моделі.

2. Окремо обробляються самі слова, окремо – їх позиція у реченні, що дозволяє розпаралелити частину обчислень і суттєво їх прискорити, за рахунок GPU чи TPU.

3. Використання механізму головок уваги (англ. «attention»), щоб враховувати взаємозв'язок між словами в реченні. Для цього використовується спеціальна нейронна мережа, яка оцінює зв'язки між словами та їх важливість для розуміння усього речення та які слова поєднують частини речення. Саме це є ключовою особливістю трансформерів, які спричинили революцію у обробці природномовного тексту, а згодом – і в обробленні зображень.

У блозі наведено гарний приклад для англомовного речення «The animal didn't cross the street because it was too tired». На рис. 7.11 показано діаграму визначення того, з якими словами пов’язано слово «it», яке приєднує останні 4 слова до основного речення, а тому є важливим.



а) б)

Рис. 7.11 Аналіз того, з якими словами найбільше пов’язано слово «it» у реченні «The animal didn’t cross the street because it was too tired» (із блогу): а) з двома головками («attention»); б) з 8 головками

Як видно на рис. 7.11а, доволі очевидно (див. більш темний колір комірок у смугах зліва), що «it» з одного боку пов’язано зі словом «animal», а з іншого – зі словом «tired». Саме в цьому й проявляється двонаправленість

аналізу. Використовуються різні підходи до пошуку цих зв'язків. Однак, цей приклад – доволі вузький. Зазвичай, використовуються 8 різних підходів (головок або механізмів уваги), як показано на рис. 7.11б. Рис. 7.11б можна самостійно дослідити і з іншими словами на інтерактивній діаграмі у ноутбуці у Google Colab (це – ноутбук, який демонструє різні можливості бібліотеки TensorFlow). А потім отримані результати аналізу з 8 головок усереднюється і формується єдиний ембеддінг (рис. 7.12).

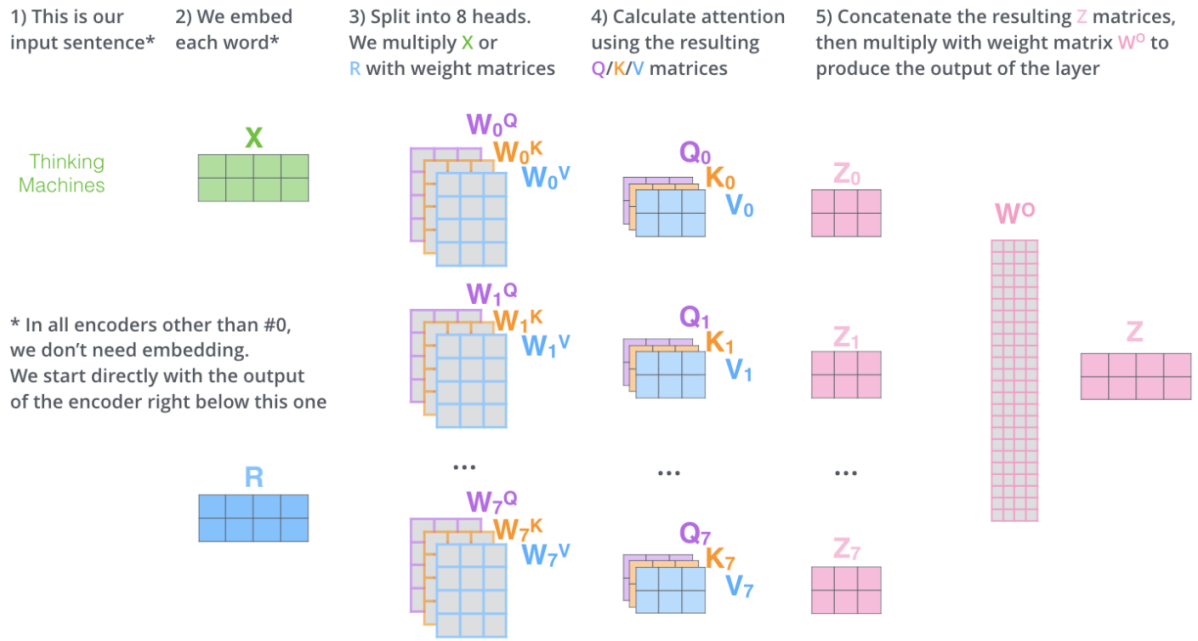


Рис. 7.12 Механізм формування ембеддінгів в моделі трансформера (із блогу)

Трансформер, як модель, вважається однією з найкращих (англ. «State-of-the-art»), як для NLP-задач, так і для задач, пов'язаних із зображеннями (це видно і на рис. 6.14), де вони отримують назву «Vision Transformers» (ViTs). Для цього зображення так само розбиваються на фрагменти (патчі), як текст розбивається на слова, підслова чи склади. А далі застосовується весь потужний апарат трансформерів для пошуку зв'язків і залежностей між цими патчами. І це дозволяє вирішувати різні задачі: сегментація, виявлення і розпізнавання об'єктів, у т.ч. людей, класифікація об'єктів і, навіть, генерування нових зображень.

7.2.6. BERT

На моделі трансформера основана технологія BERT. Щодо неї можна зазначити такі додаткові особливості, які не відзначались вище щодо трансформерів:

1. BERT використовує токенізацію на рівні підслів (англ. «WordPiece Tokenization»), що дозволяє розглядати слова на більш низькому рівні і враховувати морфологічні особливості мови.

2. Використовується своя система токенізації з додатковими позначками, які залежать від задачі, яка розв'язується (рис. 7.13). Також, спеціальним чином маркується початок і кінець кожного речення.

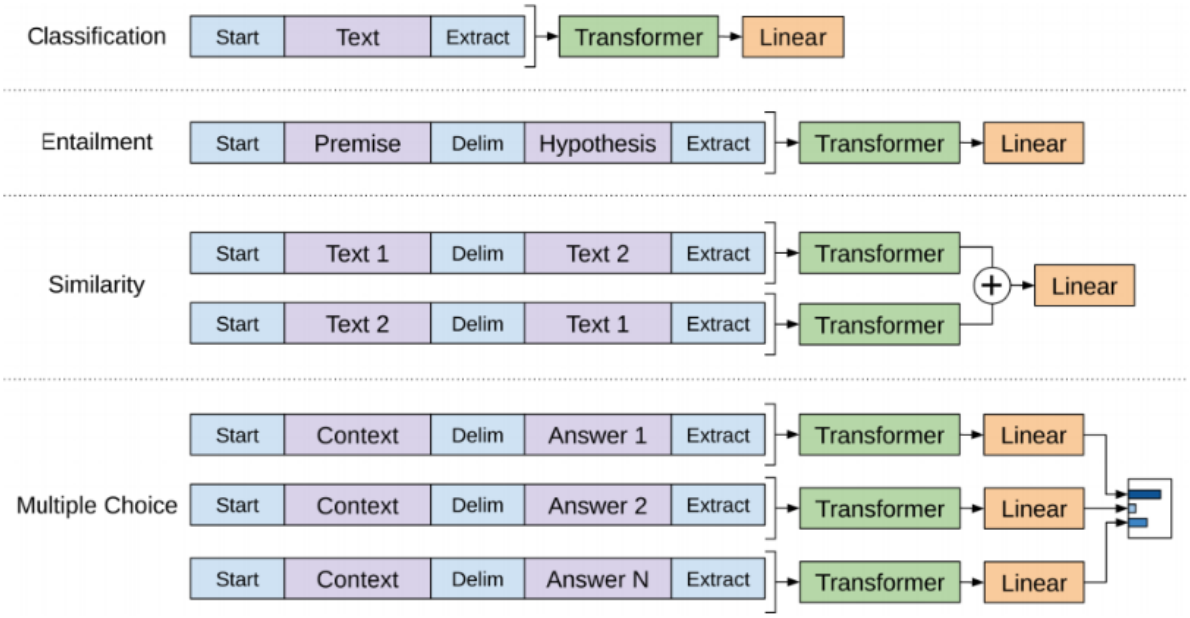


Рис. 7.13 Види задач BERT та особливості формування токенів для них (із блогу)

3. В архітектурі моделі використовується Dropout, що дозволяє краще узагальнювати висновки. Це дозволяє моделі ефективно передбачати пропущені слова у реченні (рис. 7.14).

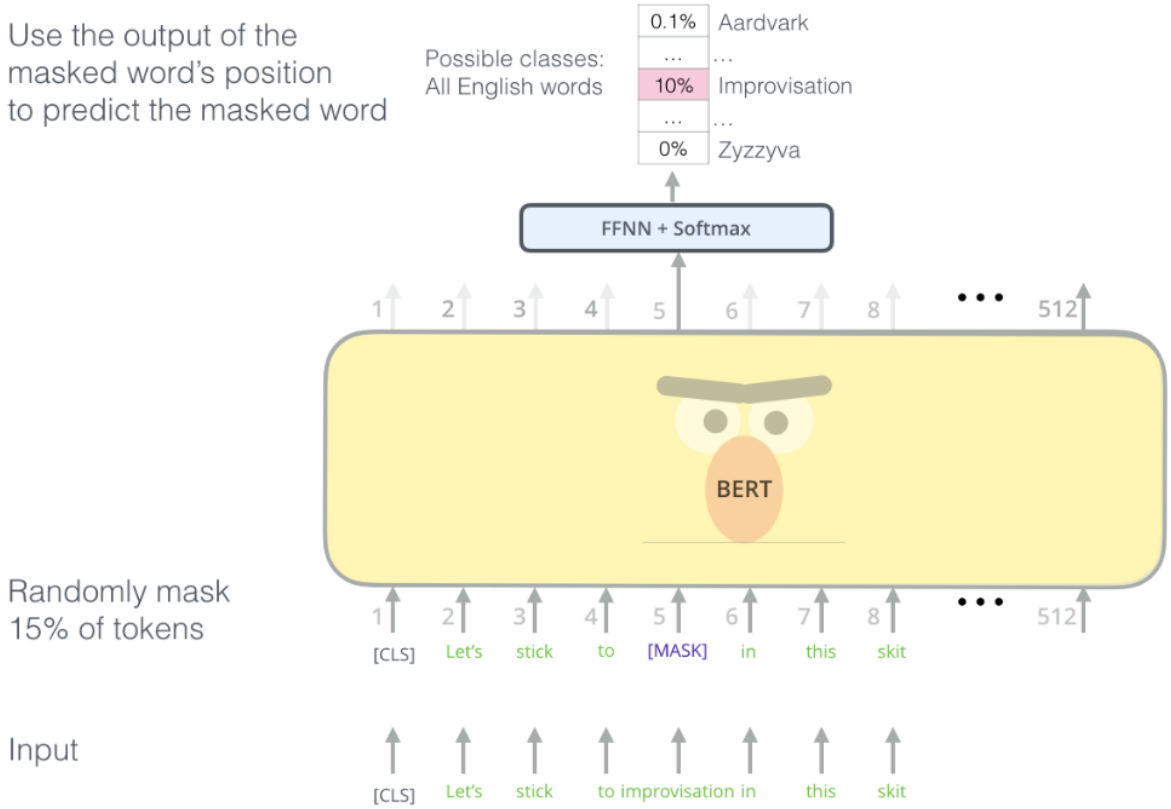


Рис. 7.14 Ілюстрація передбачення пропущеного слова з використанням BERT (із блогу)

Дуже гарно деталізоване та ілюстроване пояснення механізму роботи BERT наведено у блозі.

7.2.7 Hugging Face (HF). Алгоритм класифікації текстів з використанням HF

Бібліотека Hugging Face (HF) містить велику кількість передтренованих мовних моделей BERT, GPT, T5 тощо та адаптованих до них токенізаторів, які одразу можна застосовувати до різних задач щодо тексту різними мовами, з урахуванням різних обмежень на доступні обчислювальні ресурси (рис. 7.15).

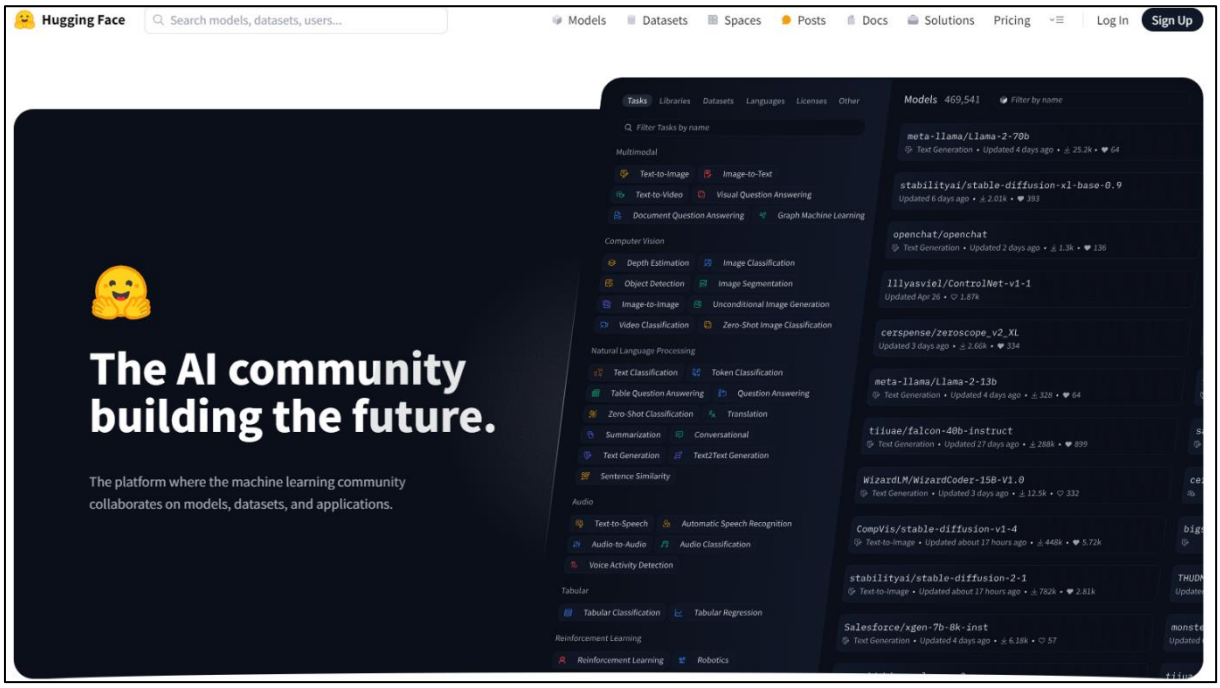


Рис. 7.15 Бібліотека Hugging Face (HF)

Станом на січень 2024 року, HF містить:

- більше 300 тисяч моделей, у т.ч. більше 110 тисяч різних трансформерів;
- більше 100 тисяч прикладних застосунків;
- більше 50 тисяч датасетів;
- біля 10 тисяч швидкісних токенізаторів.

Зазвичай, для класифікації природномовних текстів з використанням HF використовується такий алгоритм:

1. Завантажити дані та відділити з них таргет.
2. Вибрати та інсталювати передтреновану модель з бібліотеки Hugging Face (HF).
3. Здійснити передоброблення даних, нормалізацію та токенізацію речень тексту (часто ці термінів вживаються як синоніми, залежить від контексту: іноді потребується певне передоброблення тексту до токенізації, а іноді токенізацію розглядають як етап передоброблення). Для токенізації використовуються токени теж з бібліотеки Hugging Face (HF).
4. Застосувати вибрану мовну модель та отримати з неї ембеддінги для вхідних даних, які є числовим аналогом вхідного тексту.
5. Сформувати додаткові фічери.
6. Поєднати ембеддінги з п. 4 з іншими фічерами з п. 5 та таргетом із п. 1, щоб отримати датасет для машинного навчання з учителем.
7. Поділити дані на навчальні, валідаційні і тестові.
8. Натренувати одну з моделей із розд. 4 передбачати заданий таргет.
9. Здійснити передбачення тестових даних.
10. Проаналізувати результати. Якщо результат не є задовільним, тоді можна внести зміни в етапи 2, 3, 5, 7, 8 (вибрати іншу мовну модель у HF,

змінити алгоритм передоброблення даних, змінити алгоритм формування додаткових фічерів, змінити поділ на тренувальні та валідаційні дані, удосконалити чи вибрати іншу модель чи ансамбль моделей).

Приклади застосування цього алгоритму наведено у таких ноутбуках, у т.ч. авторських:

1. Ноутбуки конкурсу Kaggle «Natural Language Processing with Disaster Tweets».

2. Ноутбуки датасету «NLP : Reports & News Classification», створеного у Kaggle за участі одного зі співавторів на основі його монографії, виданої українською та англійською, описаної вище. В цьому датасеті наведено розмічений датасет зі 100 речень з тієї монографії, однакові за змістом (окремо українською, окремо англійською мовою), з експертними оцінками щодо того які речення стосуються питань моніторингу, екологічних проблем, очищення вод тощо. Авторам ноутбуків пропонується передбачати ці класи за змістом речень. Для ускладнення речення копіювались студентами з PDF-файлу монографії без редагування (разом з номерами сторінок, колонтитулами тощо):

3. Ноутбуки датасету «NLP with Disaster Tweets - cleaning data» для конкурсу Kaggle «Natural Language Processing with Disaster Tweets».

Існують бібліотеки-«надбудови» до HF, наприклад simpletransformers, які дозволяють в одну команду виконати п. 3 і 4 разом. Якщо знехтувати правилами оформлення програм у зручному для читабельності вигляді, тоді весь алгоритм пп. 1-10 можна розмістити в одній комірці – див. авторський ноутбук «Supershort NLP classification notebook», який досяг доволі високої точності на одному зі згаданих вище навчальних конкурсів Kaggle «Natural Language Processing with Disaster Tweets».

Окремо варто зупинитись на п. 5 алгоритму, який є дещо специфічним для NLP задач.

7.2.8 Інженерія ознак для задач класифікації природномовних текстів

Для ефективного розв’язання NLP задач важливо вміти враховувати додаткову інформацію та, окрім ембеддінгів, які є векторним представленням-аналогом тектів, видобувати й інші ознаки. Для цього можна ефективно використати згадану вище бібліотеку NLTK.

Зазвичай, фічери в NLP задачах формуються одним з таких способів:

4. використання синтаксичних або граматичних фічерів, таких як частини мови (англ. «Part of Speech» – «POS») або граматичні залежності (для цього можна використати бібліотеки NLTK, SpaCy та ін.).

5. статистичні характеристики складів, слів, речень, параграфів, документів у корпусі текстів загалом (для цього можна використати бібліотеки NLTK, re, звичайні бібліотеки статистичного аналізу та ін.) – гарний приклад є у ноутбуці, де автор намагається по фічерах відрізнити тексти

(есе), написані машиною (великими мовними моделями), від текстів, написаних студентами (у призовому конкурсі Kaggle «LLM - Detect AI Generated Text»);

6. показник TF-IDF (див. п. 7.2.2), який відображає частоту унікальних слів, не враховуючи загально вживані слова (див., наприклад, коментар про своє рішення учасника, який посів 19 місце (Top1.5%) у змаганні Kaggle «Google AI4Code – Understand Code in Python Notebooks»);

7. формування N-грам (не плутати зі скіп-грамами!) тощо.

N-грами в NLP – це послідовності з N елементів (слів чи символів), які зустрічаються у реченнях, параграфах чи інших частинах тексту. Зазвичай, задається 2 числа (N_1, N_2), де $N_2 \geq N_1$, що означає пошук послідовностей, які містять від N_1 до N_2 елементів включно.

Наприклад, у конкурсі Kaggle «LLM - Detect AI Generated Text» більшість публічних найкращих рішень використовують (3, 5)- та (3, 6)-грами з використанням команди TfidfVectorizer(ngram_range=(3, 5)) з пакету sklearn.feature_extraction.text.

Ноутбук з видобування ключових слів, створений за участі одного з авторів посібника, використовував (N,N)-грами. Один з результатів наведено на рис. 7.16 [45].

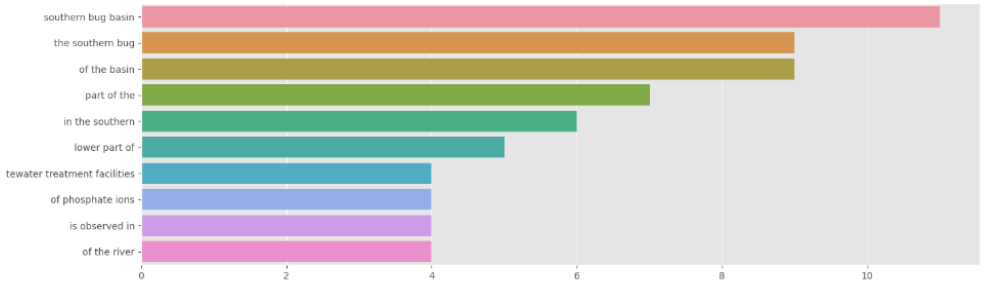


Рис. 7.16. Приклади (3, 3)-грам слів та їх кількість в корпусі слів [45]

На рис. 7.17 наведена інфографіка операцій, згаданих у підрозділах 7.1 та 7.2, у нотації рис. 1.2.

Рисунок 7.17 – Інфографіка інтелектуальних технологій з передоброблення та класифікації природномовного тексту

7.3 Чат-боти та великі мовні моделі (LLM)

Справжню революцію останнім часом спричинила поява ChatGPT (з англ. «Chat Generative Pre-training Transformer» – чат на основі генеративного передтренованого трансформера, враховуючи, що «GPT» перекладається саме як «генеративний передтренований трансформер») версії 3.5, а згодом і 4.0 від компанії «OpenAI». Мільйони людей долучились до його тестування. Мільярди доларів почали виділятись на розвиток аналогічних

сервісів усіма провідними IT-компаніями. Уряди країн та ряд компаній спочатку спробували заборонити цей сервіс, але «скринька Пандори» вже відчинена. Зрозумівши, що прогрес вже не спинити, зосередились над розробленням правил і обмежень. Наприклад, чат-боти частіше стали писати: «Відповісти не можу, бо це порушує… сформулюйте питання по-іншому». «Відповісти не можу, краще зверніться до …» тощо.

Особливо цінними є такі можливості цієї системи:

1) Можливість задати питання рідною (у т.ч. українською) мовою, хоча досвід показує, що найбільш повними є відповіді на англомовні питання;

2) Детальні розгорнуті відповіді з можливістю задавати уточнюючі питання і отримувати продовження пояснень в діалоговому режимі;

3) Можна відредагувати раніше задане питання, а не писати нове, і отримати іншу відповідь;

4) Можна натиснути «Regenerate» (англ. «Перезапуск») і згенерувати інший варіант відповіді, причому система запитає чи нова версія краща, нагадуючи, що усі користувачі є її тестерами і вона постійно на них самонавчається;

5) Генерує програми на Python на рівні доволі професійного програміста, який знає все найцікавіше з програм на Github, Kaggle, StackOverFlow тощо;

6) Дозволяє здійснювати «нативний» («природномовний») переклад з однієї мови на іншу у різних стилях;

7) Дозволяє скорочувати свій же текст до заданої кількості слів чи речень;

8) Є зручні кнопки для копіювання усієї відповіді або тільки програмного коду;

9) Усі попередні ланцюжки питань і відповідей користувача зберігаються в його профілі під назвами, які їм присвоює чат-бот, але користувач може їх перейменовувати;

10) Наявність доступу через API (платного, але недорогого), який дозволяє значно розширити можливості його застосування у своїх програмах;

11) Можливість не слідкувати за грамотністю запиту, оскільки, завдяки вбудованій розвиненій мовній моделі, сервіс, у більшості випадків, правильно «розуміє» та трактує і недолугі чи некоректні формулювання неносієм мови, і орфографічні чи граматичні помилки у запиті, наприклад, порівняйте відповіді чат-бота GhatGPT 3.5 на такі запити (одразу попередимо, що результат буде однаковий і може відрізнятись тільки деякими числовими параметрами, які генеруються там щоразу випадково):

- «Доброго ранку! Якщо тобі не важко, напиши, будь ласка, програмний код на Python з прикладом створення нейронної мережі для оброблення даних»;
- «код для нейромережі»;
- «кд неромрежі»;

12) Можливість задати питання просто навівши вхідні дані, які є, і вихідні, які треба отримати, а він сам здогадається як написати програму, яка здійснить таке перетворення;

13) Може нарисувати примітивний графік для ілюстрування відповіді, але з використанням псевдографіки (із символів з клавіатури), просто, щоб передати ідею, але є й можливості інтегрування з іншими сервісами, які це роблять набагато краще, або – написати код для якісної візуалізації результатів на Python.

Звичайно, є й недоліки:

1) Треба почекати, поки згенерується відповідь (наприклад, викладачі, щоб стимулювати студентів вчити все самостійно, вводять обмеження на час відповіді, який є дещо меншим, ніж чат-бот генерує відповідь на це питання);

2) *Галюцинації* – явище, коли чат-бот не знає відповіді, але, користуючись тим, що він є мовною моделлю, по суті – гарним письменником, а не тільки – програмістом, починає доволі реалістично сочиняти текст у відповідь, наприклад, веб-адреси датасетів чи використаних у відповіді статей, назви книжок та ін., які ніколи не існували, тому усі його відповіді потребують обов'язкової перевірки;

3) Результат виводиться у спеціально відформатованому вигляді, який потребує значного очищення формату, у разі, якщо є бажання вставити текст у Word-файл;

4) Програмний код часто генерується не одним блоком, який було б зручно скопіювати, а – частинами з текстовими коментарями між цими блоками, а тоді доводиться копіювати усю відповідь і потім довго видаляти зайве, або копіювати код з кожного блоку окремо;

5) У відповідях є помилки з різних причин:

- через проблеми у матеріалі, на якому він навчався (ніхто й не казав, що усі програми у Github, Kaggle, StackOverFlow та ін. не мають помилок!);
- через проблеми моделі чат-бота (вона все ще розвивається);
- через неправильне розуміння запиту;
- через грубі помилки користувача під час формулювання запиту (запитав не те, що хотів отримати, або не так);
- через старіння матеріалу, особливо, коли питання стосується бібліотек програм, які динамічно розвиваються;
- через відмінності сучасної версії датасету, на якому навчалась програма чат-боту, та тієї, яка зараз доступна користувачеві (наприклад, один із авторів посібника під час тестування однієї з перших версій чат-боту переконався, що вона неправильно розв'язує відому задачу з передбаченням тих, хто вижив на Титаніку в конкурсі Kaggle, вважаючи, що там дещо інша кількість ознак, ніж було насправді);

6) Намагання спрощувати відповіді, якщо користувач не вимагає додаткових деталей;

7) Обмеження на обсяг запиту та відповіді, особливо у безкоштовній версії;

8) Не існує достатньо надійних інструментів для розпізнавання того, чи текст написано людиною, чи чат-ботом (є спроби на кшталт GPTZero та ін., але вони часто помиляються), однак, це не означає, що їх не буде в подальшому: хтось згенерує текст, який пройде сучасні перевірки, але згодом таки буде доведено, що це – результат роботи штучного інтелекту, однак виправити це вже буде неможливо і будуть репутаційні втрати не тільки в «автора», який це видав за свій оригінальний текст, а й – у тих, хто це опублікував, не маючи змоги вчасно і якісно перевірити правду;

9) Обмежені можливості у роботі з графікою та неможливість готувати слайди презентації у PowerPoint, Google. Slides чи ін.

Саме зазначені недоліки досі залишають актуальним існування професії програмістів, оскільки:

1) Треба вміти правильно задати питання, щоб отримати дійсно ефективну відповіді, а для цього вже треба орієнтуватись в питанні і точно знати що питати, наприклад, порівняйте відповіді на запити:

- «код для нейромережі»;
- «код для нейромережі з 5 прихованими шарами і dropout між ними, проміжні функції активації - relu, остання - sigmoid, зі змінною швидкістю навчання, у разі, якщо точність не змінюється протягом 5 епох, з побудовою кривої навчання»;
- «код для нейромережі з 5 прихованими шарами і dropout між ними, проміжні функції активації - relu, остання - sigmoid, зі змінною швидкістю навчання, у разі, якщо точність не змінюється протягом 5 епох, з побудовою кривої навчання, на pytorch»;

2) Чат-бот не напише всю програму, готову до використання разом з іншими блоками – тільки дасть ключові блоки, які має поєднати досвідчений програміст;

3) Треба вміти розпізнати галюцинації та помилки у відповіді і здогадатись на що їх треба замінити (теоретично, можна йому ж писати які там помилки і він виправляється, але іноді зациклюється – радить попередню першу, теж помилкову, відповідь, потім – другу, потім знову – першу… тобто не може знайти відповідь, але завжди намагається щось пробувати відповісти);

4) Багато компаній категорично забороняють писати у запит приклади даних із цих компаній, щоб він написав код для їх оброблення – програміст сам повинен вигадати аналог і потім перенести його на потрібні дані;

5) Чимало компаній взагалі забороняють використання чат-ботів для написання коду, інакше потім виникають серйозні проблеми з авторським правом на цей код;

6) Комусь же треба писати нові програми для подібних сервісів, створювати нові мовні моделі та датасети, на яких вони будуть навчатись;

7) Часто відповіді потрібні під конкретну дужу вузьку і сучасну сферу з її специфічною термінологією, про яку чат-бот не знає достатньо.

Чат-боти, подібні до ChatGPT 3.5, будуються на використанні великих мовних моделей (англ. «Large Lingual Model» – LLM). Прикладами LLM на даний момент є великі мультимовні моделі GPT3, GPT4, BERT, RoBERTa, XLNet та ін. LLM для ChatGPT 3.5 це – GPT3. До GPT3 були й попередні версії, наприклад GPT2. Дивись, наприклад авторський приклад [46] – ноутбук із застосування GPT2, BERT та XLNet для узагальнення тексту у 2022 р. у Kaggle, розроблений ще до появи ChatGPT 3.5.

LLM мають значно більш складну архітектуру, значно більше параметрів (мільярди і сотні мільярдів) та навчаються на більш складних і різноманітних багатомовних текстах.

ChatGPT має параметр «temperature» (з англ.: «температура»), який регулює ступінь різноманітності відповідей (ймовірно, через аналогію з тим, що з підвищенням температури об’єкта зростає хаотичність його

Виникли і бурхливо розвиваються 2 відносно нових напрями у цій сфері:

- *Інженерія запитів* (англ. «Prompt Engineering») – набір знань і навичок для генерування запитів до чат-бота, які нададуть дійсно релевантну і максимально корисну відповідь, іноді, для цього використовуються сторонні інтерфейси, які спеціалізуються на певній предметній галузі;
- *LLM Fine Tuning* – технології «довчання» або продовження навчання великої мовної моделі на заданій предметній галузі, на спеціальному корпусі слів (при цьому певні базові шари і параметри можуть «заморажуватись», щоб не зіпсувати базові знання та «розмовні» навички).

У наш час щодня з’являються все новіші можливості для створення власних чат-ботів на основі готового коду та LLM, оптимізованих під відносно обмежені обчислювальні можливості користувачів. Популярним є «LLM Fine Tuning».

Хоча раніше, користувачі створювали свої чат-боти, переважно, у сервісі Amazon Lex чи йому подібних, які містять вбудовані засоби для аналізу голосових чи текстових інтентів (шаблонів запитів). Тобто користувач-розробник має створити систему таких шаблонів. Сервіс орієнтується який саме шаблон використав користувач-тестер та з якими параметрами (назви, кількість, час тощо) і за шаблоном визначає яку саме програму слід запустити та передає їй ті параметри, щоб отримати відповідь. Відповідь теж виводить за певними шаблонами. Сучасні чат-боти, за рахунок LLM дозволяють набагато більше і ця IT-індустрія тільки-но почала свій бурхливий розвиток.

Є LLM з відкритим вихідним кодом (англ. «open-source» наприклад: LLAMA2 (https://ai.meta.com/llama/) від Meta (версії 7B, 13B і 70B), Falcon

(https://falconllm.tii.ae/), Mistral (https://docs.mistral.ai/), MPT-7B (https://www.mosaicml.com/blog/mpt-7b) та багато інших [58].

7.4 Інтелектуальний аналіз природномовного тексту і мовлення

Технології NLP дозволяють вирішувати багато прикладних задач.

Наведемо короткий огляд конкурсів Kaggle з цієї тематики з реалістичними постановками задач:

Крім того, є чимало інших прикладів розв’язання реальних задач з використанням інтелектуального аналізу природномовного тексту:

1. Видобування ключових слів.
2. Створення WISEST.
3. Створення рекомендаційних систем.
4. Пошук знань та побудова онтологічних мереж.
5. Аналіз якості есе за стилем їх написання.
6. Синтез або верифікація тестових завдань.

Створення рекомендаційних систем.

У конкурсі, Linking Writing Processes to Writing Quality, учасники використовують дані про клавішні натискання, щоб передбачити загальну якість письма. Їх робота досліджує зв'язки між поведінковими особливостями під час письма та результативністю. Основна мета полягає в ідентифікації взаємозв'язків між способами письма та його якістю, що може сприяти самостійності, метапізнанню та саморегуляції учнів у письмі. Додатково, це дослідження може збагатити розуміння процесу письма та його впливу на кінцевий результат.

Суть рішення, що зайняло перше місце, полягає у використанні методів очищення даних та відтворення речень з даних про натиснені клавішні, для покращення якості письма. Після цього використовуються різноманітні методи інженерії ознак та моделювання, такі як статистика інтервалів між натисканнями, використання зовнішніх даних про оцінку есе та використання різних алгоритмів класифікації та регресії для прогнозування оцінок тексту. Використано також ансамбль моделей для підвищення точності прогнозів, а післяпроцесування включало обмеження значень передбачень для покращення результатів (рис. 7.17).

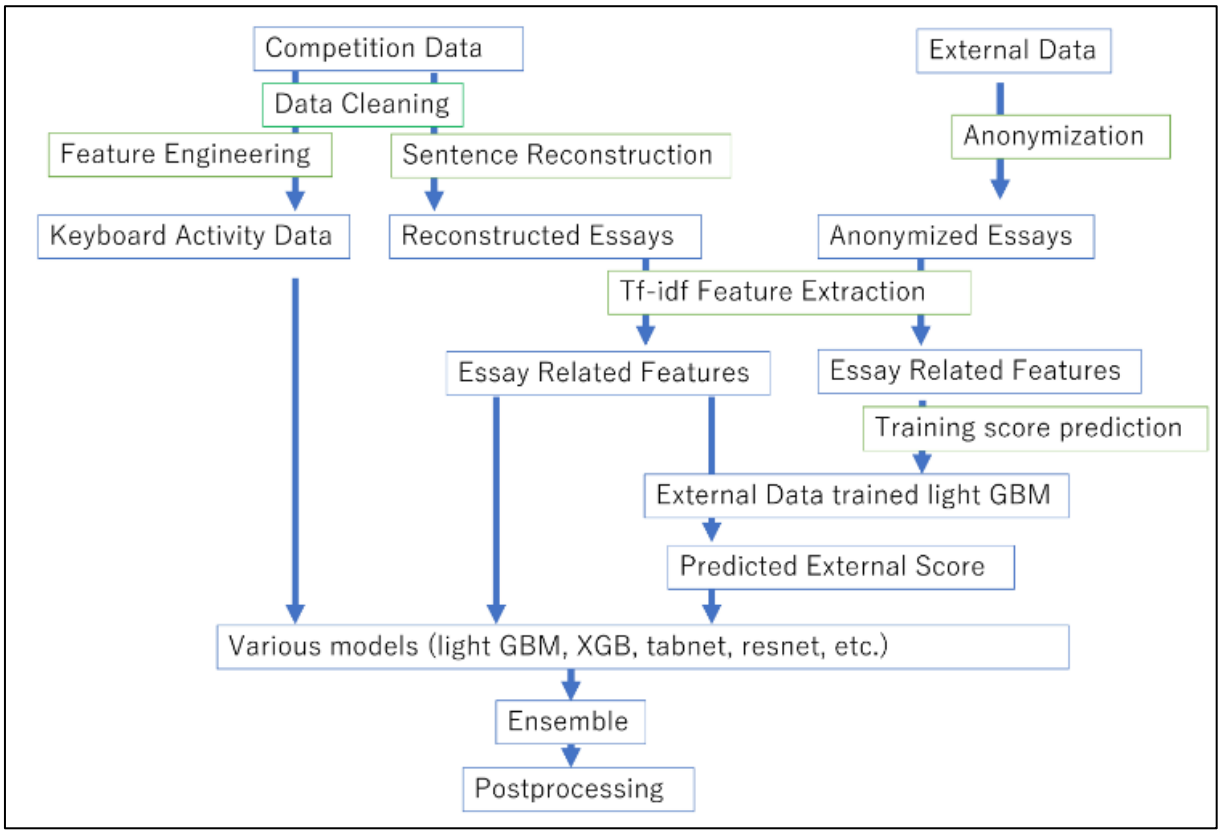


Рисунок 7.17 – Загальний алгоритм переможного рішення

У конкурсі CommonLit - Evaluate Student Summaries Учасники стикаються з складним набором поведінкових дій та когнітивних процесів у процесі письма, включаючи планування, редагування, та стратегічне розподілення часу. Ці малі дії можуть впливати на якість письма, проте більшість оцінок зосереджуються лише на кінцевому продукті. Застосування даних у науці може розкрити ключові аспекти письмового процесу через аналіз даних про натискання клавіш для передбачення загальної якості письма та виявлення взаємозв'язків між поведінкою письменників та їх продуктивністю. Це може сприяти увазі учнів на їх процес вироблення тексту та підвищити їхню автономію, метакогнітивну усвідомленість та саморегулювання при письмі.

Рішення, яке посіло перше місце використали модель microsoft/deberta v3 large з 4-випробуваною складкою для свого остаточного внеску (рис. 7.18). Вони зосередилися на покращенні різноманітності тем у навчальних даних, використовуючи псевдо-мітки, підказки для генерації тематичного контенту та двоетапне навчання моделі. Зміни у підготовці даних та упорядкування тексту для зменшення часу інференції також були ключовими елементами підходу. Вони використовували відкритий вихідний код і не вносили жодних змін у модель. Крім того, вони надали велике значення досвіду та внутрішнім інсайтам інших учасників, сподіваючись, що їхні дії приносять користь спільноті.

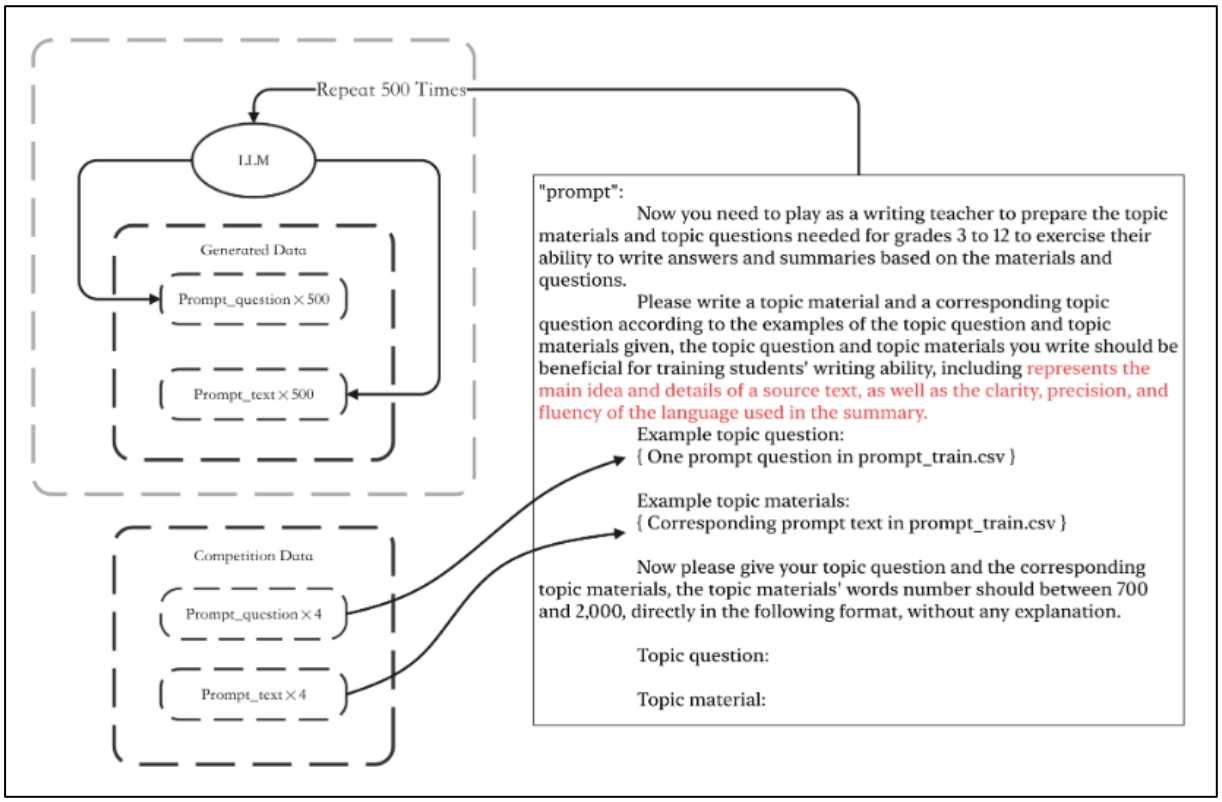


Рисунок 7.18 – Підказка, яка використовується для створення підказок LLM в рішенні

Завдання конкурсу, Kaggle - LLM Science Exam, полягає у відповіді на складні наукові питання, які були створені великомасштабною мовною моделлю (LLM). Учасники мають вирішити це завдання, щоб допомогти дослідникам краще зрозуміти здатність LLM тестувати себе та потенціал LLM, які можуть працювати в умовах обмежених ресурсів.

Основна мета полягає в тому, щоб визначити, наскільки ефективно моделі відповідають на питання, створені більш великою моделлю, і вивчити можливості використання LLM для тестування самих себе.

Учасники, що зайняли перше місце вирішували завдання конкурсу з використанням широкого спектру контекстно-орієнтованих моделей мовного оформлення. Їх підхід базувався на пошуку контексту з статей Вікіпедії та застосуванні великої кількості моделей, включаючи LLM. Вони експериментували з різними підходами до кодування тексту Вікіпедії та питань, а також використовували різні варіації довжини контексту під час навчання та інференсу. Збалансований підхід до навчання, включаючи використання штучно створених даних та оптимізовану архітектуру моделі, спричинив успішний результат у вирішенні завдання (рис 7.19).

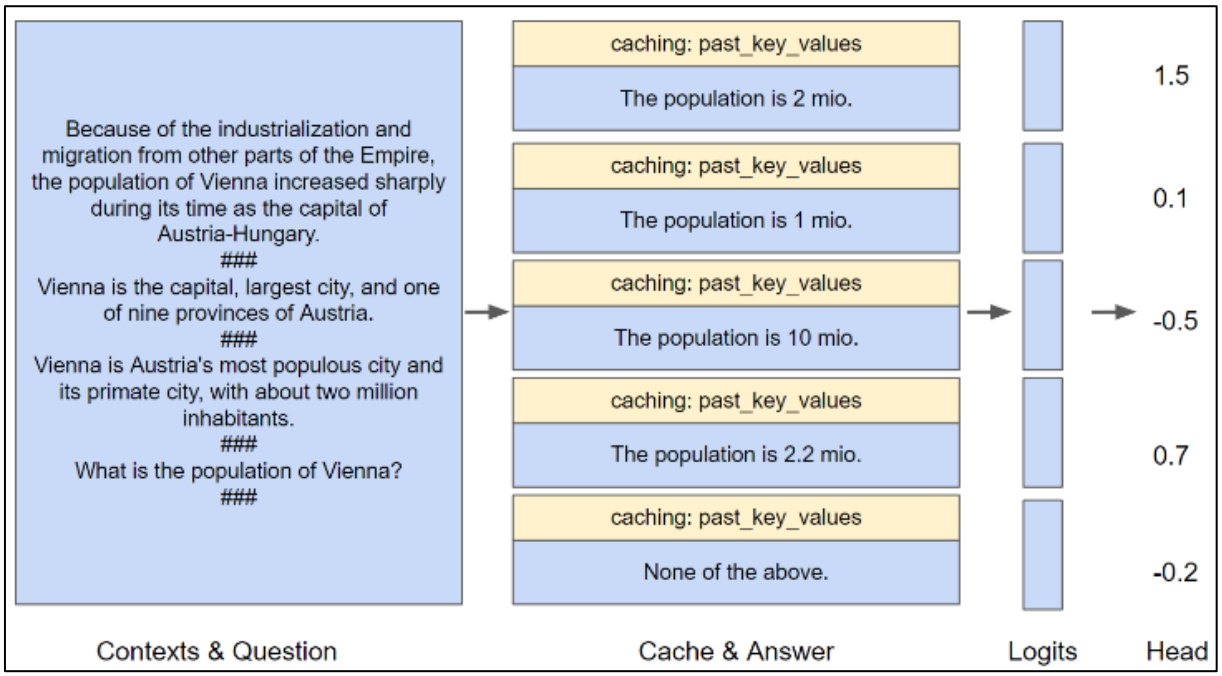


Рисунок 7.19 – Діаграма пояснює архітектуру та налаштування висновків

Про WISEST та задачі з Каггла.
Про SpaCy, NER та ГІС.
Класифікація тексту на прикладі нашої розробки по Півд. Бугу.
Узагальнення даних (саммаризація)
Пошук знань та побудова онтологічних мереж.
Створення рекомендаційних систем.

На рис. 7.18 наведена інфографіка інструментарію, згаданого у розділі 7 в цілому, у нотації рис. 1.2.

Рисунок 7.18 – Інфографіка інтелектуальних технологій з аналізу та генерування природномовного тексту (NLP)

Можливі теми практичних і лабораторних завдань

Тема № 1. Аналіз та класифікація тексту природною мовою з використанням технологій глибокого навчання та NLP (або **«Інтелектуальний аналіз текстових даних про стан складної системи з використанням технологій штучного інтелекту на Python»**).

Метою роботи є вивчення інформаційних технологій і Python-бібліотек для аналізу та класифікації тексту природною мовою (NLP) з використанням технологій глибинного навчання й опанування практичних навичок застосування деяких із них на прикладі одного із датасетів Kaggle чи за даними, завантаженими через API.

План заняття:

1. Вибрати датасет (наприклад датасет «NLP : Reports & News Classification»):

2. Завантажити дані (якщо вони – кирилицею, тоді – правильно налаштувати кодування). Відібрати дані для заданої у варіанті бригади цільової ознаки («target»).

3. Підключити модель одну з передтренованих BERT-моделей з бібліотеки «Hugging Face», наприклад 'distilbert-base-multilingual-cased' («multilingual» - працює саме з кирилицею, як і з іншими 104 мовами світу) та правильно адаптувати до неї дані (токени, маски тощо).

4. Натренувати модель для післяоброблення даних – для класифікації за навчальними даними виходу BERT-моделі, тобто даних ембедингів, на які вхідний текст перетворює модель з п.3. Такою моделлю може бути й дуже складна модель – логістична регресії, дерево рішень чи нейронна мережа з 1 чи 2 шарами. Налаштувати цю модель.

5. Зробити передбачення даних на тестовому датасеті та порівняти із заданими. Оцінити точність за accuracy_score для випадку, коли у тестовому датасеті опиняється 40% від вхідних даних. Спробувати оцінити як змінюється точність за розміру тестових даних у 10%, 20%, 30% від загальної кількості (решта – навчальні дані).

6. Здійснити візуалізацію отриманих результатів.

7. Навести посилання на створений ноутбук, який містить усі результати роботи (п. 2-6 та коментарі англійською мовою щодо того, яка модель - оптимальна).

Приклади ноутбуків із розв’язання задач класифікації текстових даних з використанням NLP-технологій:

- Data Science with DL & NLP: Advanced Techniques
- NLP - EDA, Bag of Words, TF IDF, GloVe, BERT
- NLP with Disaster Tweets - EDA and Cleaning data
- NLP for UA : BERT Classification for Water Report
- NLP for EN : BERT Classification for Water Report

Контрольні питання

1) Що включає в себе передоброблення даних у природномовному тексті?

2) Які основні поняття використовуються в мовних моделях для розв'язання NLP-задач?

3) Що таке "мішок слів" (Bag of Words) і як він використовується в аналізі тексту?

4) Як працює TF-IDF і в чому його переваги у порівнянні з "мішком слів"?

5) Що таке GloVe та в чому полягає поняття ембеддінгів у контексті мовних моделей?

6) Як працює Word2Vec і які можливості воно відкриває для аналізу тексту?

7) Що таке моделі-трансформери, і як вони використовуються в розв'язанні завдань NLP?

8) Як працює BERT і як він відрізняється від інших моделей для аналізу тексту?

9) Які підходи використовуються для інженерії ознак у задачах класифікації природномовних текстів?

10) Що таке чат-боти і як вони використовуються разом з великими мовними моделями для покращення комунікації з користувачами?

8 ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ І ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

8.1 Основні поняття та види задач

Задачі аналізу та прогнозування часових рядів є одними з найпоширеніших, які потребують розв'язання. Часто важливо не просто побудувати модель, яка зможе точно і надійно прогнозувати ряди даних – цінність може мати сама модель. Ідентифікований період і частота сезонності, тип найкращої моделі може багато сказати про сам ряд. Іноді важливо припасувати модель не до останніх значень ряду, а – до усіх значень за весь період спостережень, щоб краще зрозуміти відхилення та зміну його поведінки у різний час. Всі ці задачі вирішуються з використанням технологій аналізу і прогнозування, основані на специфічних моделях часових рядів, які враховують час, коли було зафіксоване кожне значення.

Отже, якщо у задачах передбачення важливо видаляти самі дати значень після видобування з них певних фічерів, то в задачах з використанням часових рядів, навпаки, час видаляти не можна. Більше того, для розв'язання задачі часто достатньо одного фічера та часу. Але більшість моделей є багатофакторними, тобто дозволяють враховувати й інші фічери, які можуть впливати на основний фічер. З поняттям основного фічера пов'язана класифікація цієї задачі на задачу машинного навчання з учителем і без вчителя:

1. Задача прогнозування часового ряду методами машинного навчання є *задачею із учителем*, якщо визначена основна ознака, яку слід прогнозувати і тоді її відомі значення можна розглядати як таргет, навіть, якщо інші фактори не аналізуються. Наприклад, задача прогнозування щоденної кількості хворих на коронавірус у країні є задачею із учителем, якщо є ряд таких значень і треба знайти наступні. Можна враховувати ще й ряд щоденної кількості вакцинованих, протестованих, Google-тренди щодо мобільності людей, погодні фактори тощо. Але ця задача буде задачею з учителем і без врахування цих факторів, оскільки таргет є чітко визначеним.

2. Задача аналізу часового ряду методами машинного навчання є задачею *без вчителя*, якщо задача поставлена таким чином, що немає єдиної основної ознаки, яку слід аналізувати, а треба просто знайти певні закономірності з використанням методів кластеризації, виявлення аномалій, статистичного аналізу тощо. Наприклад, проаналізувати міжрегіональний вплив кількості хворих на коронавірус в часі з метою виявлення осередків та напрямів поширення хвиль, що дозволить прийняти рішення щодо локалізації проблеми чи мінімізації її масштабів.

Приклади на коронавірусній хворобі є доволі зручними, оскільки, по-перше, усі читачі, напевне, знайомі з цим явищем, по-друге, захворювання носило сезонний багатохвильовий характер, по-третє, мав (і, на жаль, досі має) місце вплив великої кількості факторів, які впливають на поширення

кількості хворих, вчетверте, роботи авторів посібника довели ефективність саме часових рядів для аналізу цих даних, вп’яте, масштаби зусилля вчених багатьох країн дозволили накопичити потужні датасети, доступні через API, на яких тепер можна відпрацьовувати різні моделі, щоб підготуватись до нових подібних хвороб та викликів щодо застосування свого вміння прогнозувати поширення хвороб та здійснювати підтримку прийняття урядових рішень щодо мінімізації їх негативного впливу.

Важливими поняттями є крок ряду і горизонт прогнозування.

Крок ряду – це різниця між сусідніми значеннями. Ряд моделей, наприклад, ARIMA вимагає, щоб крок був однаковим і не було пропущених значень. Якщо вони є, то спочатку їх слід чимсь заповнити, а вже потім будувати модель. Натомість, більш сучасні моделі, на кшталт Facebook Prophet чи нейромережевих рекурентних моделей, не мають такого обмеження.

Горизонт прогнозування – це кількість кроків, на які робиться прогноз. Важливо, що це можуть бути і не тільки майбутні значення. На цьому основано поняття *кросвалідаційного діагностування моделі* ряду., яке використовує, наприклад, бібліотека Facebook Prophet. Задача розглядається як задача з учителем. Горизонтом стають значення всередині наявного інтервалу даних – вибірково беруться різні частини даних і використовуються як валідаційні, а тоді аналізується як гарно модель їх прогнозує, використовуючи для тренування усі інші дані (рис. 8.1).

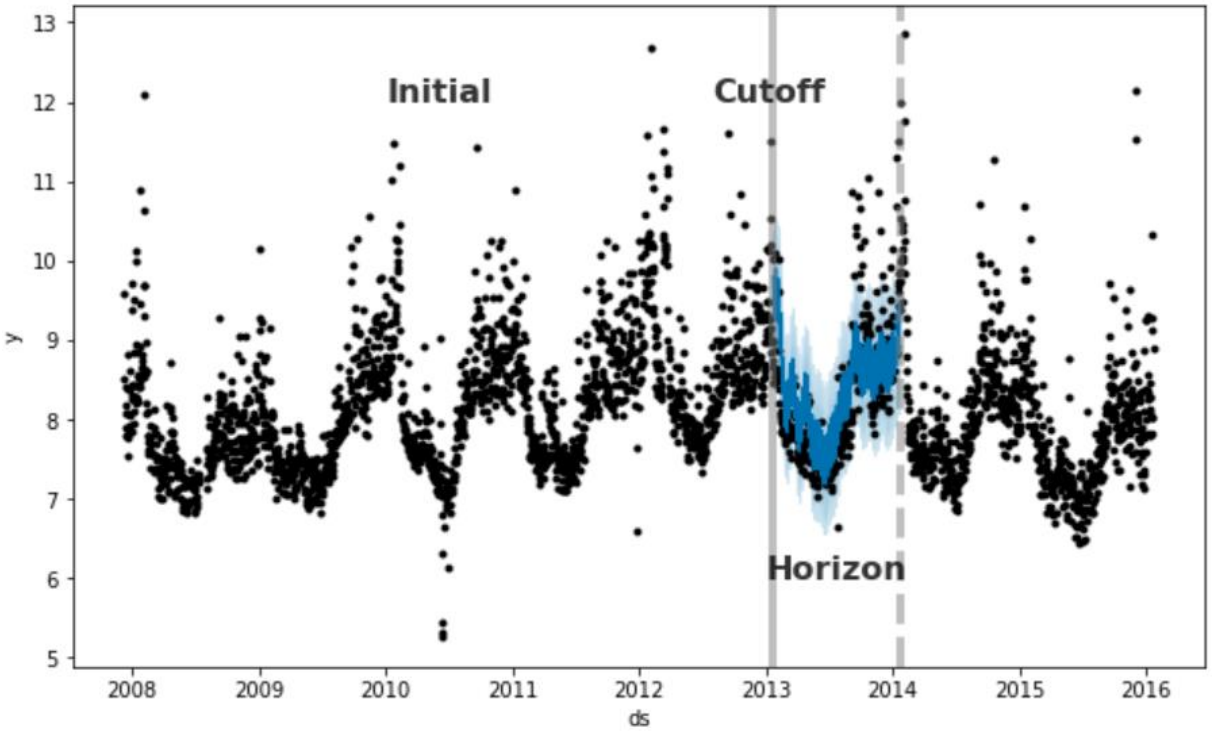


Рисунок 8.1 Кросвалідація під час діагностування часового ряду з горизонтом усередині інтервалу відомих даних спостережень[47]

Розрізняється коротко- і довгострокове прогнозування. Як правило, це пов'язано не з тривалістю в секундах, годинах чи добах, а – в кількості кроків та закономірностях самого ряду, передусім з сезонністю. Якщо прогнозування здійснюється на порівняно невелику кількість кроків або у межах однієї чверті періоду сезонності (тільки наростання чи тільки зменшення значень даних), тоді це – *короткострокове прогнозування*, інакше воно є *довгостроковим*. Наприклад, якщо слід спрогнозувати щодобове зростання кількості хворих на коронавірус з горизонтом в 1-2 тижні, тоді це – короткострокове прогнозування, а якщо прогнозувати коли завершиться поточне наростання чи збільшення кількості хворих або коли почнеться нова хвиля через 1-2 місяці, тоді це буде вже довгострокове прогнозування, оскільки слід враховувати дуже багато факторів та великі ряди цієї та аналогічних хвороб. А для короткострокового прогнозування можна використовувати й короткі ряди.

Один з авторів посібника Мокін В.Б. протягом 2020-24.02.2022 рр. був членом Робочої групи з математичного моделювання проблем, пов'язаних з епідемією коронавірусу SARS-CoV-2 в Україні, яка регулярно готувала аналітику для РНБО України та інших установ та інституцій. Усі більше 60 звітів доступні на сайті Президії Національної академії наук України (http://www.nas.gov.ua/UA/Activity/covid/Pages/wg.aspx). Мокін В.Б. у звітах цієї Робочої групи, спільно зі своїм аспірантом Лосенком А.В., вів розділ «Прогноз розвитку епідемії в Україні з використанням статистичної моделі часових рядів Facebook Prophet». Ця модель часто показувала кращі результати, ніж модель на основі системи диференціальних рівнянь типу SEIR, яку використовував колектив науковців НАН України (рис. 8.2).

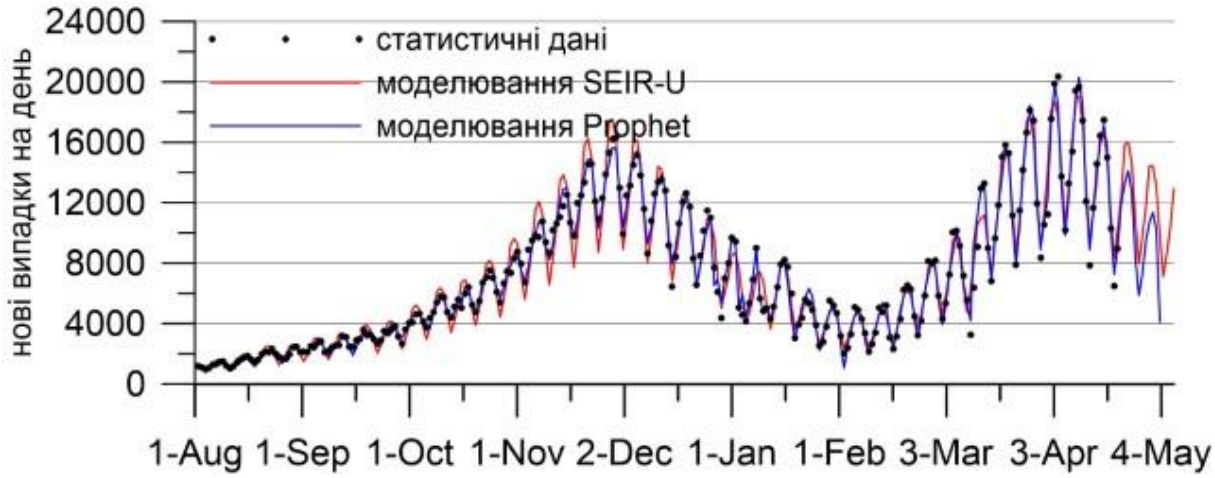


Рисунок 8.2 – Порівняльний графік моделі SEIR-U та моделі на основі Facebook Prophet з горизонтом у 2 тижні у звіті «Прогноз РГ-42», де точність статистичної моделі в 1,35 рази виявилась вищою, ніж точність моделі SEIR-U

На жаль, аналіз показав, що процес поширення коронавірусу не має стабільної сезонності. За відносно короткими рядами спостережень у 2020

році не вдалось побудувати модель для дійсно довгострокового прогнозування. А тому модель та розроблені для неї методи машинного навчання (методи ідентифікації структури і параметрів) добре працювали тільки на коротку перспективу і тільки під час активного зростання чи зменшення кількості нових хворих. А отже, цей розділ є тільки у 25 звітах, які робились під час проходження так званих хвиль, коли різко зростала кількість хворих і потім – зменшувалась до відносно невисокої. У міжхвильовий період модель була неадекватною і ці результати у звіти не включались. Але, з іншого боку, саме під час хвиль такі результати і були найбільш цікавими для усіх.

Однак, важливо чітко розуміти, що кожен математичний апарат, кожна модель машинного навчання має свої обмеження, свою область застосування, де її слід застосовувати і де це буде дійсно ефективно та корисно. Тобто щоразу слід не тільки знайти оптимальну модель, а й – оптимальні умови, коли вона буде найбільш ефективною. А в інших умовах використовувати інші моделі.

Іншими популярними прикладами задач прогнозування часових рядів є прогнозування:

- даних моніторингу параметрів навколишнього середовища: метеоданих, стану вод, стану атмосферного повітря, сонячної активності тощо;
- курсу криптовалют чи інших активів на фінансових ринках;
- значень показників приладів у різних технічних системах;
- результатів аналізів медичних приладів для прогнозування стану пацієнтів;
- кількості хворих під час пандемій чи епідемій у різних регіонах для прогнозу масштабів хвороби та аналізу можливих напрямків її поширення;
- зміни радіосигналів, у т.ч. космічного походження, тощо.

8.2 Розвідувальний аналіз та інженерія ознак даних часових рядів

Розвідувальний аналіз часових рядів іноді ще називають *діагностуванням часового ряду*. Цей процес містить усі ті ж етапи, які і класичні задачі машинного навчання і пов’язаний з пошуком та усуненням пропущених даних, виявленням аномалій, пошук чи синтез нових фічерів, аналіз їх важливості, аналіз закономірностей самого ряду. Однак, в силу, залежності в часі та статистичної природи самих моделей часових рядів, ці етапи мають чимало відмінностей.

Як правило, аналіз часового ряду проводиться за такими етапами (їх послідовність не є догмою – кожен дослідник може застосовувати свою):

1. Аналіз пропущених значень і регулярності кроку. За можливості, варто усунути пропущені дані та зробити їх регулярними, особливо, якщо планується використання ARIMA-моделей.

2. Аналіз аномалій ряду. Бажано виписати моменти, коли мали місце такі аномалії (для моделі FB Prophet формується спеціальний датафрейм з такими датами та «вікном» їх впливу на сусідні значення). Більш детально про це буде нижче.

3. Аналіз закону розподілу ряду. В першу чергу, перевіряється чи є він розподіленим за нормальним законом, оскільки більшість методів аналізу побудовані, виходячи з гіпотези про нормальність цього закону. Якщо закон розподілу не є нормальним, то є можливість його нормалізувати.

4. Аналіз стаціонарності ряду. За можливості, його варто стаціонаризувати, наприклад взяти першу різницю ряду (наприклад, за формулою $\Delta x_i = x_{i+1} - x_i$) – це особливо цінно, якщо планується використовувати нейромережеві моделі, а, наприклад, для моделі ARIMA цей етап не є необхідним – вона автоматично ідентифікує порядок різниці (складова «I» моделі), яка стаціонаризує ряд.

5. Аналіз сезонності ряду. Бажано виявити не тільки факт наявності сезонності, а й скільки і з яким періодом є варіантів цієї сезонності, наприклад, лабораторії аналізу тестів на коронавірус працювали з чітким тижневим циклом, а ще були хвилі кількості хворих, обумовлені іншими факторами, а ще були й інші закономірності у межах кожного тижня, тобто аналізу показав, що має місце не менше 3-х видів різної сезонності [48]. Крім того, на цьому етапі може бути прийнято рішення згладити ряд, щоб усунути вплив одного з видів сезонності і, у такий спосіб, знизити зашумленість даних, принаймні візуально.

6. У разі, якщо на етапах 3-6 здійснювалась трансформація ряду, варто заново порахувати аномалії з п.2, або додати нові аномалії до тих (наприклад, у разі переходу до першої різниці для стаціонаризації ряду, до аномальних значень курсу біткоїна варто додати аномально великі стрибки цього курсу вгору чи вниз).

7. У залежності від результатів етапів 1-8, варто прийняти рішення які далі варто будувати моделі. Цьому присвячено підрозд. 8.3.

8. Інженерія ознак (FE). У разі, якщо далі планується пробувати використовувати класичні багатофакторні моделі машинного навчання (регресійні, дерева рішень, звичайні нейромережі тощо), тоді варто спробувати знайти або синтезувати побільше ознак, навіть, якщо задано тільки один фічер.

9. Визначити горизонт(и) прогнозу та сформувати тренувальний і валідаційні датасети.

Можуть бути й інші етапи, наприклад аналіз ряду на *гетероскедастичність* (коли статистичні характеристики змінюються в часі за певним законом, який можна окремо ідентифікувати) [49], але, в загальному випадку, цей етап пропускають.

Розглянемо ці етапи більш детально.

8.2.1 Аналіз та усунення пропущених у часових рядах даних

Аналіз та усунення пропущених даних робиться у такий спосіб:

1. Визначаються перший і останній моменти даних спостережень та їх крок.

2. Генерується послідовність моментів часу між цими моментами з визначеним кроком.

3. Здійснюється злиття датафрейму з моментами часу з п.2 і даними спостережень (pd.merge() чи аналогічна команда).

4. Аналізуються чи є пропущені дані у новому датафреймі з використанням типових команд.

5. Якщо планується використовувати моделі, які вимагають регулярний крок, наприклад ARIMA, тоді слід здійснити імпутинг даних (див. розд. 2). Якщо ж плануються менш вимогливі моделі на кшталт FB Prophet, нейромережевих LSTM чи ін., тоді імпутинг не є обов'язковим, оскільки будь-який імпутинг – це псування оригінальних даних, оскільки алгоритм імпутингу може давати значення, які суттєво відрізняються від справжніх.

8.2.2 Аналіз аномалій часових рядів

На етапі EDA важливо виявити основні аномалії ряду, оскільки, якщо їх не врахувати і намагатись будувати єдину модель і для них, і для інших значень, то вона наврядчи буде адекватною. Найпростішим способом є згадане у розд. 3 фільтрування по квантилях, тобто на основі порівняння екстремальних значень (максимум/мінімум) та граничних квантилів P10, P05, P90, P95 тощо. Але є й спеціальні бібліотеки [50]:

- SESD;
- Isolation Forest (scikit-learn);
- Statsmodels.

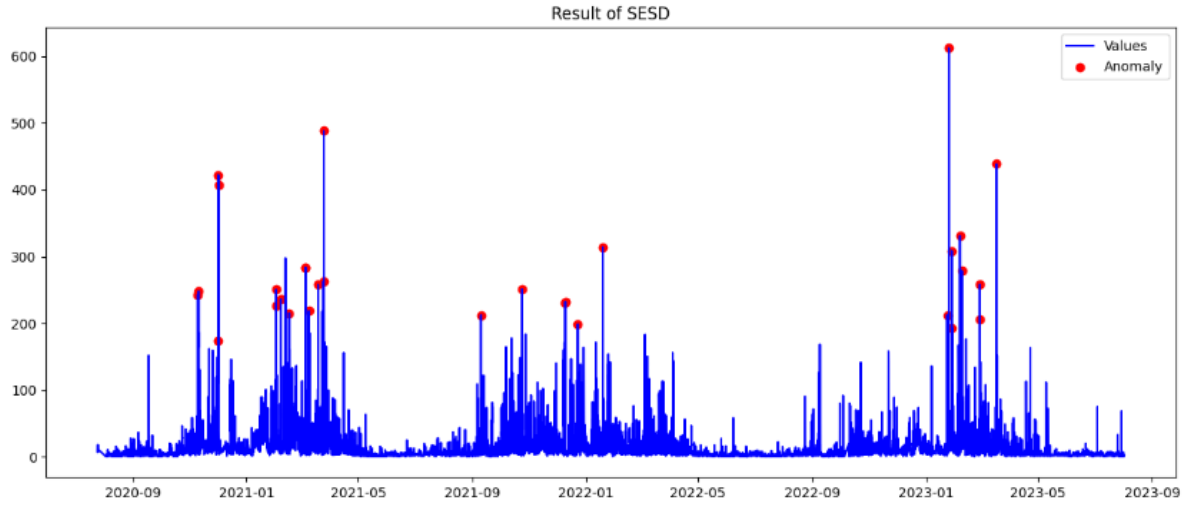
Python-бібліотека SESD автоматизує метод Seasonal Hybrid ESD з комбінуванням сезонної адаптації з експоненційно зваженим середнім (Exponential Smoothing) і використовує Extreme Studentized Deviate (ESD) для виявлення відхилень від очікуваного розподілу даних. Перевагами методу є такі: 1) врахування сезонності; 2) гнучкість налаштовування (період сезонності і порогові значення); 3) підтримка широкого спектра даних (числові часові ряди, а також категоріальні та багатовимірні дані). Недоліки: 1) обмежена робота з незвичайними аномаліями: метод краще працює з аномаліями, які досить схожі на звичайні сезонні зміни; 2) доволі повільний, у порівнянні з іншими.

Модуль Isolation Forest бібліотеки scikit-learn автоматизує метод виявлення аномалій шляхом ізоляції аномалій у даних. За даними будується класичне дерево рішень (див. 4.1.11), але по усьому простору ознак і без жодної регуляризації. Значення anomaly_score для алгоритму є глибиною листка в побудованому дереві. Тобто аномальними вважаються «листя» (кластери даних), найбільш віддалені від «кореня». Переваги моделі: простота використання; швидкість обчислень, навіть за значних обсягів даних; стійкість

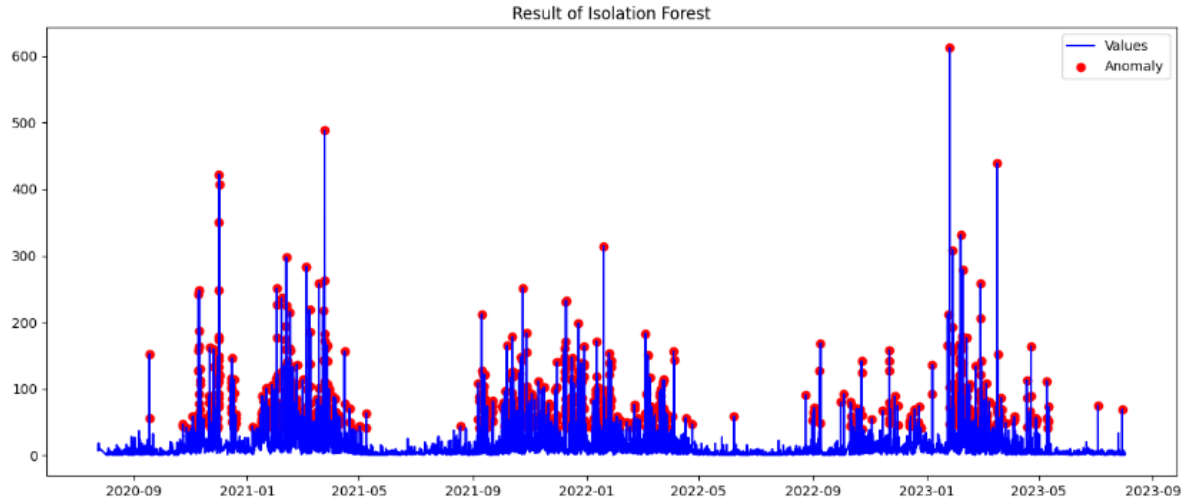
(має гарну стійкість до викидів та шуму в даних). Недоліки: може давати хибні результати, якщо буде забагато даних, які будуть віднесені до аномальних (дуже глибокі дерева); низька ефективність для великої кількості ознак (великорозмірні дані), оскільки усі дані обробляються одразу разом, а не – партіями; обмеженість щодо параметрів для налаштування, у порівнянні з іншими бібліотеками.

Python-бібліотека statsmodels немає якоїсь однієї функції для виявлення та фільтрації аномалій у часових рядах. Однак, можна використовувати інші функції та методи з бібліотеки для реалізації такого аналізу. Один із підходів – це метод аналізу залишків (seasonal_decompose). Можна використовувати статистичні моделі, такі як ARIMA або ETS для прогнозування часового ряду. Порівнюючи фактичні значення з прогнозованими, можна виявити аномалії, коли значення значно відрізняються від прогнозу – це доволі поширений підхід з використанням й інших методів і моделей із розділів посібника 4 і 5. Інший підхід – це використання статистичних методів. Це включає встановлення порогових значень, використання стандартного відхилення, Z-перетворення та інших методів для виявлення значень, які перевищують певний поріг [50]. Перевагами є гнучкість у виборі параметрів налаштувань, а недоліками є складність у виборі моделі чи методу.

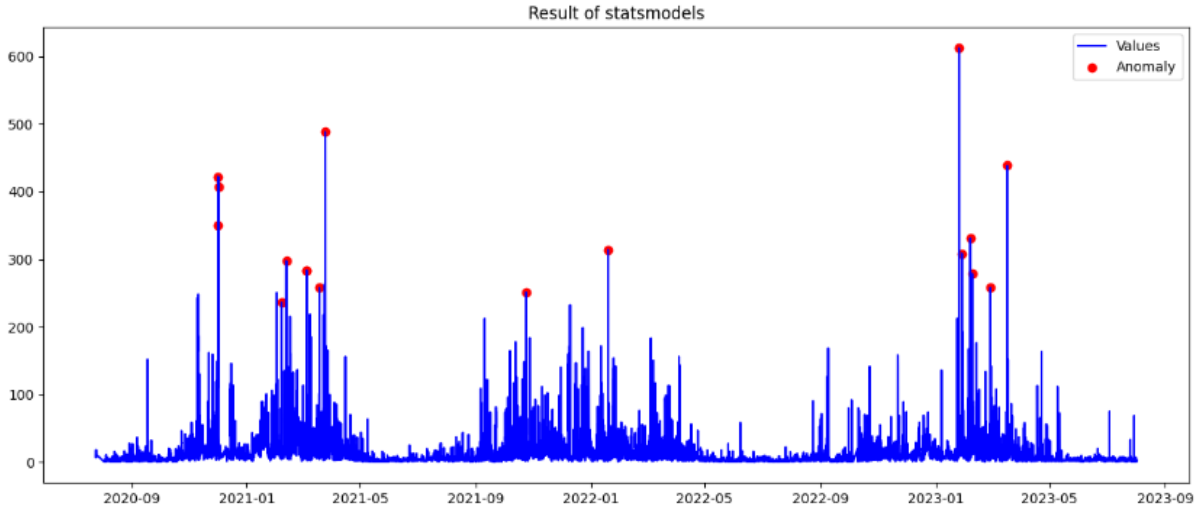
На рис. 8.3 відображені різні варіанти аномалій, ідентифіковані у різний спосіб, за даними мережі громадського моніторингу стану атмосферного повітря EcoCity із ноутбука «Anomaly detection for air pollution».



а)



б)



в)

Рисунок 8.3 – Автоматична ідентифікація аномалій значення концентрації пилу в атмосферному повітрі з використанням різних бібліотек [50]: а) SESD; б) Isolation Forest; в) seasonal_decompose із Statsmodels

Є й більш простий спосіб визначення аномалій – експертним шляхом:

1. Побудувати графік значень. Краще використовувати інтерактивні графіки бібліотеки Plotly (рис. 8.4). А тоді візуально вибрати аномалії (найбільші і найменші значення, які сильно відрізняються) та визначити відповідні їм моменти часу. Те саме можна повторити з першою і другою різницями ряду (найбільші прирости і прирости приростів даних).

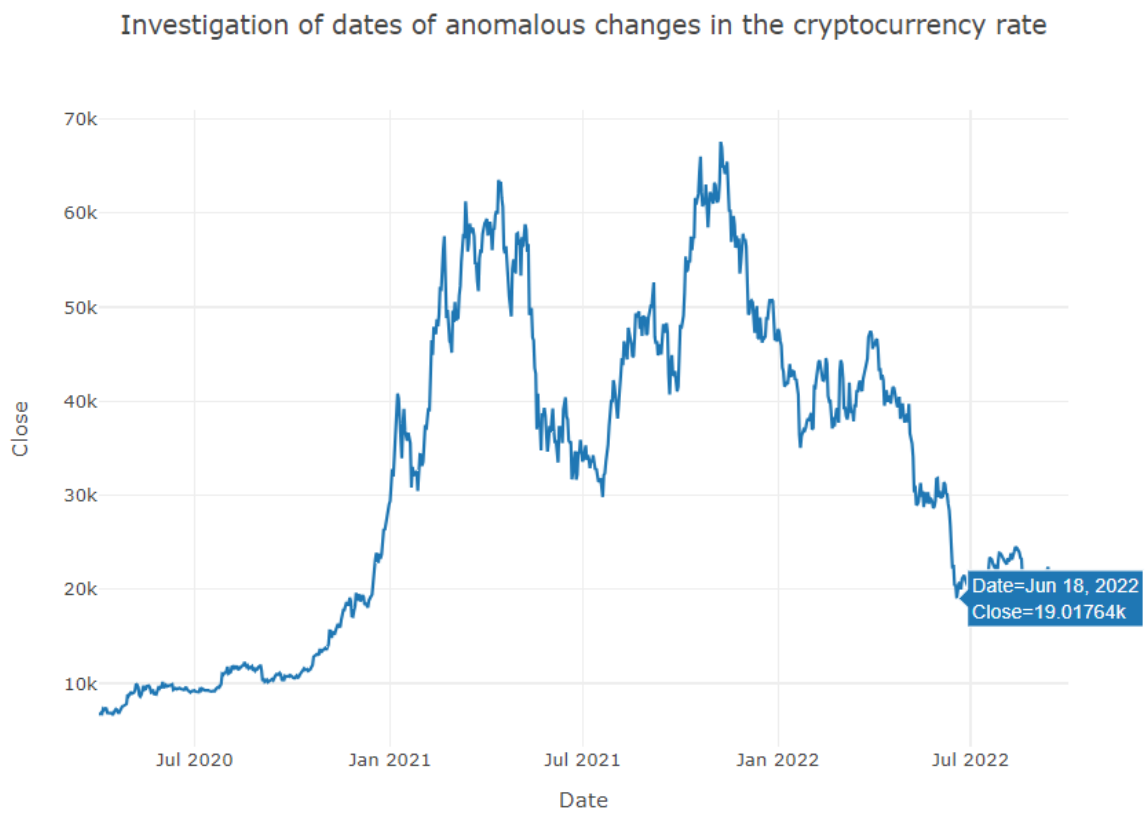


Рисунок 8.4 – Різке падіння курсу біткоїна у червні 2022 р. [18] через крах криптобіржи Celsius Network, стейблкоїна Tron (USDD) та ін. [51]

2. Проаналізувати предметну область – чи дійсно є підстави вважати, що з цим моментом може бути пов'язана певна аномалія (наприклад, в задачі прогнозування курсу криптовалюти, це може бути злам системи, рішення уряду однієї з провідних країн щодо неї чи придбання/продаж активу однією з провідних компаній світу тощо) чи це може бути звичайне значення ряду (наприклад, теплі дні влітку, якщо їх порівнювати з холодними днями взимку). Дуже гарний аналіз спекулятивних криптовалютних бульбашок є у Вікіпедії – яскравий приклад аналіз дат аномалій. Загалом, спекулятивні дії на фінансових ринках є доволі звичним явищем, але статистичним моделям їх важко прогнозувати, тому для підвищення точності моделей їх треба маркувати як аномалії.

3. Якщо є підстави вважати це аномалією, тоді оцінюється «вікно» впливу, оскільки вже за декілька моментів часу чи днів міг бути певний анонс події, крім того, ця подія може мати й певну післядію (перед- і післяноворічний розпродаж товарів має вплив на продажі не тільки 31 грудня).

4. Чи впливає аномалія миттєво чи є певна затримка? Наприклад, щодо коронавірусу, святкові дні в Україні явно мали вплив на кількість хворих, але із запізненням на 4-7 днів, тобто слід було робити зсув дат і «вікна» навколо них [52].

8.2.3 Аналіз закону розподілу даних

Як було зазначено вище, більшість методів аналізу даних побудовані, виходячи з гіпотези про нормальність закону розподілу цих даних. Тому важливо, щоразу, перевіряти цю гіпотезу. Якщо вона не виконується, тоді варто спробувати нормалізувати дані.

Одна з найбільш поширених проблем із даними, це – коли більш частими є менші значення. Це характерно, наприклад для даних спостережень за станом довкілля, за відсутності аномальних антропогенних чи природних впливів. Тоді можна спробувати застосувати математичну трансформацію і посунути пік гістограми вправо, щоб він став більш схожим на нормальний закон розподілу. Для цього застосовується метод Бокса-Кокса, який підбирає параметр λ для забезпечення кращої відповідності нормальному закону. Його легко автоматизувати з використанням функції boxcox(x) бібліотеки stats (рис. 8.5). На рис. 8.5 (з документації бібліотеки stats) видно, що зручним способом аналізу нормальності закону розподілу є функція stats.probplot(x, dist=stats.norm). Вона обчислює квантилі. Якщо закон розподілу – нормальний, тоді ці квантилі шикуються у лінію. Трансформація за методом Бокса-Кокса саме забезпечує вирівнювання цих квантилів.

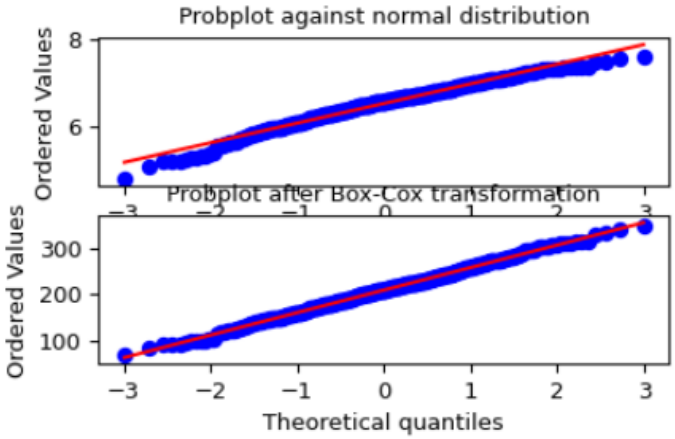


Рисунок 8.5 – Результат застосування методу Бокса-Кокса до нормалізації часового ряду

Важливо пам’ятати про необхідність зворотності цієї операції. Тобто алгоритм має бути таким:

1. EDA: аналіз закону розподілу: **stats.probplot(x, dist=stats.norm)**
2. Трансформація даних за методом Бокса-Кокса:
   x_new, λ = stats.boxcox(x)
3. Виконання необхідних обчислень з даними: навчання моделей, здійсненням за ними прогнозування даних.
4. Зворотна трансформація за методом Бокса-Кокса:
   stats.boxcox_inv(x_new, λ)
5. Візуалізація і трактування результатів.

Ще одним важливим аспектом цього етап є порівняння законів розподілу навчальних і валідаційних даних – якщо вони надто відрізняються, тоді модель не буде ефективною.

Часто цей етап опускають і сподіваються, що той факт, що закон розподілу фічера чи фічерів не є нормальним, не суттєво вплине на результат прогнозування. Але все ж таки, рекомендується його не пропускати.

8.2.4 Аналіз стаціонарності

Стаціонарним є ряд, в якому статистичні показники – незмінні в часі. А отже, статистичні показники різних вибірок ряду будуть однаковими чи співставними у межах заданої похибки.

Класичним прикладом стаціонарного ряду є напруга в розетці протягом року, за відсутності збоїв. Як правило, вона зазнає змін тільки у межах певного розкиду, інакше вийде з ладу все електрообладнання. Класичним прикладом нестаціонарного ряду є температура повітря в Україні протягом року. Якщо взяти вибірку взимку чи влітку, звичайно, середнє буде суттєво відрізнятись. Для нестаціонарних рядів є складні моделі і складні підходи. Часто, висувають гіпотезу про стаціонарність ряду і, навіть, це не перевіряють, але рекомендується це робити.

Для цього використовується тест Діка-Фуллера, який виконує функція statsmodels.tsa.stattools.**adfuller**. На рис. 8.6 наведено приклад коду, який виконує тест і виводить результат у зручному вигляді, та результат його застосування до курсу біткоїна (фічер «Close») за 2020-2022 рр.

```
result = adfuller(series.values)

print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
    print('\t%s: %.3f' % (key, value))

if (result[1] <= 0.05) & (result[4]['5%'] > result[0]):
    print("\u001b[32mStationary\u001b[0m")
else:
    print("\x1b[31mNon-stationary\x1b[0m")
```

```
ADF Statistic: -1.571470
p-value: 0.497907
Critical Values:
        1%: -3.438
        5%: -2.865
        10%: -2.568
Non-stationary
```

```
ADF Statistic: -31.128922
p-value: 0.000000
Critical Values:
        1%: -3.438
        5%: -2.865
        10%: -2.568
Stationary
```

а) б) в)

Рисунок 8.6 – Тест Дика-Фулера часового ряду курсу біткоїна за 2020-2022 рр. на стаціонарність з авторського ноутбука: а) Python-код, б) результат перевірки ряду; в) результат перевірки першої різниці ряду

На рис. 8.6 наведено типовий випадок, коли сам ряд не є стаціонарним, а його перша різниця є стаціонарною. У разі використання моделей машинного навчання чи простих моделей часових рядів, які непридатні для роботи з нестаціонарними часовими рядами, здійснюють перехід до різниць таргета, навчають моделі, прогнозують і потім виконують зворотній перехід.

8.2.5 Аналіз сезонності

Головною відмінністю моделей часових рядів від інших моделей машинного навчання є те, що вони спеціалізуються на врахуванні сезонності рядів. Саме на періодичних рядах моделі часових рядів демонструють вищу точність, аніж універсальні моделі для роботи з даними, описані у розділах 4 і 5.

Є типовий тест чи має ряд *x* хоча б одну сезонну складову. Для цього є функція statsmodels.tsa.seasonal.**seasonal_decompose**(*x, T*), де *T* – період ряду в його кроках (година, доба,…).

Як відомо,

Сезонність, як правило, описується рядами Фур'є:

Як відомо, ряд Фур'є має вигляд [1-4]:

$$y(t) = \frac{a_0}{2} + \sum_{i=1}^{n} a_i \cos\left(\frac{2\pi}{T} it\right) + \sum_{i=1}^{n} b_i \sin\left(\frac{2\pi}{T} it\right), \quad (8.1)$$

де коефіцієнти Фур'є знаходяться за допомогою виразів:

$$a_0 = \frac{1}{T}\int_0^T y(t)dt, \quad a_i = \frac{2}{T}\int_0^T y(t)\cos\left(\frac{2\pi}{T} it\right)dt,$$

$$b_i = \frac{2}{T}\int_0^T y(t)\sin\left(\frac{2\pi}{T} it\right)dt, \quad i = \overline{1,n}, \quad (8.2)$$

де, у разі, якщо ряд починається не з нуля, а з деякого часу t_0, тоді час t замінюється на $(t - t_0)$.

Як видно з виразів (8.1), (8.2) перші гармоніки мають найбільший період коливання *T* (при *i* = 1), а усі інші – менший у 2, 3, … *n* разів.

[48,53, 54,55,56]:

Сучасні функції для навчання моделей часових рядів, як правило, самі ідентифікують параметри Фур'є. Для цього їм потрібно лише 2 гіперпараметри у позначеннях формули (8.1): період «period» *T* і кількість гармонік ряду Фур'є «fourier_order» *n* (насправді, є ще деякі, але про них – пізніше).

8.2.6 Інженерія ознак часових рядів

Популярним є формування нових ознак із заданого із різним «вікном», тобто коли визначається певне усереднене чи максимальне за 3, 7, 10, 14 тощо кроків значення фічера. Для цього використовується команда rolling бібліотеки pandas. Наприклад «df['x'].rolling(window=7).mean()» обчислить середнє за тиждень значення фічера 'x' по його щоденних значеннях у датафреймі df. Його ще називають «*ковзним вікном*».

Іноді, ковзне вікно використовують не для синтезу нових ознак, а для *згладжування* таргета, тобто основного ряду, який прогнозується. Наприклад, наявність чіткого тижневого циклу роботи лабораторій для тестування на коронавірус привів до того, що багато дослідників одразу застосовували до нього «ковзне вікно» і моделювали згладжений у такий спосіб ряд, що суттєво знижувало зашумленість даних і дозволяло краще виявити основні

закономірності. Однак, слід розуміти, що така трансформація не дає можливості співставляти дані з реальними даними на кожному кроці всередині такого «вікна». На прикладі з коронавірусом, по таких прогнозах важко прогнозувати дані на завтра – вони більше ефективні в кінці кожного 7-денного циклу або – тільки для аналізу закономірностей ряду.

Існує Python-бібліотека TSFresh, яка дозволяє *синтезувати* до 1200 фічерів для заданого часового ряду. Обчислюється велика кількість різних статистичних характеристик заданого часового ряду (не тільки дисперсія чи середньоквадратичне відхилення, а й енергія та ін.) із різним «вікном». Якщо задати фільтрацію, то бібліотека дозволить одразу й відфільтровувати маловажливі варіанти. Досвід показує, що після цього, все одно, можуть бути не дуже гарні варіанти, наприклад, коли є забагато пропущених значень чи усі значення фічера є однаковими. Після вбудованого фільтрування і після власного, зазвичай, залишається 50-100 фічерів, доволі цікавих та ефективних для прогнозування. Наприклад, див. авторські приклади прогнозування середнього за добу курсу біткоїна за даними декількох років років [15, 57] у 2020-2022 рр., які дозволили спрогнозувати щодобовий курс на 10 днів вперед з точністю 3,6%, використавши 50 ознак, отриманих за допомогою бібліотеки Tsfresh із ознаки «Close».

8.2.7 Формування тренувального і валідаційного датасетів в задачах прогнозування часових рядів

Вище було описано як може вибиратись горизонт прогнозу – це можуть бути і реально майбутні значення, і значення в минулому – для перевірки точності моделі. Відповідно, слід правильно формувати й тренувальний та валідаційний датасети:

1) Якщо стоїть задача прогнозування майбутніх даних, тоді валідаційними є останні дані ряду, а тренувальними – значення до них;

2) Якщо стоїть задача аналізу даних і наближення моделі до усіх значень ряду одночасно, тоді валідаційні значення вибираються у класичний спосіб: випадково або з використанням крос-валідації з усіх наявних даних.

Однак, є важлива особливість багатофакторних моделей. Вона полягає в тому, що для прогнозування наступних N значень заданого фічера будуть вимагатись наступні N значень інших фічерів, які теж, як правило, невідомі. А тому, практикують штучний зсув даних на N значень назад. Тобто будують модель, яка прогнозує заданий фічер в i-й момент по інших фічерах у моменти (i-N). А тоді для прогнозування значень заданого фічера у моменти i+1, i+2…, N це дійсно можна буде зробити за значеннями інших фічерів у моменти часу i+1-N, i+2-N …, i. Приклад дивись у ноутбуках для прогнозування курсу біткоїна [15,57].

На рис. 8.5 наведена інфографіка операцій та моделей, згаданих у підрозділах 8.1 та 8.2, у нотації рис. 1.2.

Рисунок 8.5 – Інфографіка етапів EDA та FE для часових рядів

8.3 Побудова моделей часових рядів: ARIMA, Facebook Prophet та ін.

Для прогнозування часових рядів можна використовувати усі моделі розділів 4 і 5, якщо дещо змінити постановку задачі. Також, можна використовувати нейромережеві рекурентні моделі, зокрема GRU та LSTM, описані у підрозд. 5.3. Але, частіше, використовуються специфічні для часових рядів моделі ARIMA та Facebook Prophet.

Математичний апарат ARIMA (англ. «Autoregressive Integrated Moving Average» – модель авторегресії та проінтегрованого ковзного середнього (АРПКС)) детально описаний у підручнику одного зі співавторів [21]. Для її автоматизації на Python використовується один із двох варіантів:

1) метод ARIMA пакету arima_model бібліотеки «Time Series Analysis», який дозволяє ідентифікувати модель для заданого ряду із заданим порядком ARIMA(p,d,q), де p – порядок авторегресії, d – порядок різниці, q – порядок ковзного середнього;

2) метод auto_arima бібліотеки pmdarima, який сам підбирає параметри моделі SARIMAX, де на додаток до p,d,q, ще є параметри сезонних складових (рис. 8.7).

```
Performing stepwise search to minimize aic
 ARIMA(4,1,4)(0,0,0)[0] intercept   : AIC=6196.011, Time=2.35 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=6185.373, Time=0.03 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=6187.379, Time=0.08 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=6187.375, Time=0.04 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=6183.686, Time=0.01 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=6189.382, Time=0.04 sec

Best model:  ARIMA(0,1,0)(0,0,0)[0]
Total fit time: 2.564 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  345
Model:               SARIMAX(0, 1, 0)   Log Likelihood               -3090.843
Date:                Fri, 07 Apr 2023   AIC                           6183.686
Time:                        12:05:59   BIC                           6187.527
Sample:                             0   HQIC                          6185.216
                                  - 345
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2      3.72e+06   2.24e+05     16.620      0.000    3.28e+06    4.16e+06
===================================================================================
Ljung-Box (L1) (Q):                   0.76   Jarque-Bera (JB):                22.63
Prob(Q):                              0.38   Prob(JB):                         0.00
Heteroskedasticity (H):               0.82   Skew:                            -0.16
Prob(H) (two-sided):                  0.28   Kurtosis:                         4.21
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
CPU times: user 5.52 s, sys: 4.47 s, total: 9.98 s
Wall time: 2.57 s
```

Рис. 8.7 Приклад результату роботи auto_arima з ідентифікації моделі SARIMAX для прогнозування курсу біткоїна (з ноутбука)

Основні аспекти моделі Facebook Prophet (далі просто – Prophet) описані у статті [59] та у документації. Математично модель Prophet для моделювання та прогнозування значень ряду $y(t)$, в залежності від часу t, записується таким чином [59, 48]:

- для адитивного випадку:

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t, \tag{8.1}$$

- для мультиплікативного випадку:

$$y(t) = g(t)s(t)h(t)\epsilon_t, \tag{8.2}$$

де $g(t)$ – тренд ряду (логістична або шматково-лінійна апроксимація даних), $s(t)$ – сезонна складова, апроксимована рядом Фур'є, $h(t)$ – складова, яка враховує вплив свят чи інших аномалій, які відбуваються нерегулярно

протягом одного або кількох днів і діють з певним «вікном», тобто – в діапазоні певних дат, ϵ_t – похибка («шум») з нульовим середнім, розподілена за нормальним законом.

Більш докладно приклад ефективного налаштування параметрів моделі FB Prophet на прикладі прогнозування кількості щоденних нових хворих на коронавірус в Україні у 2020 р. (рис. 8.8) див. у статтях [52,48].

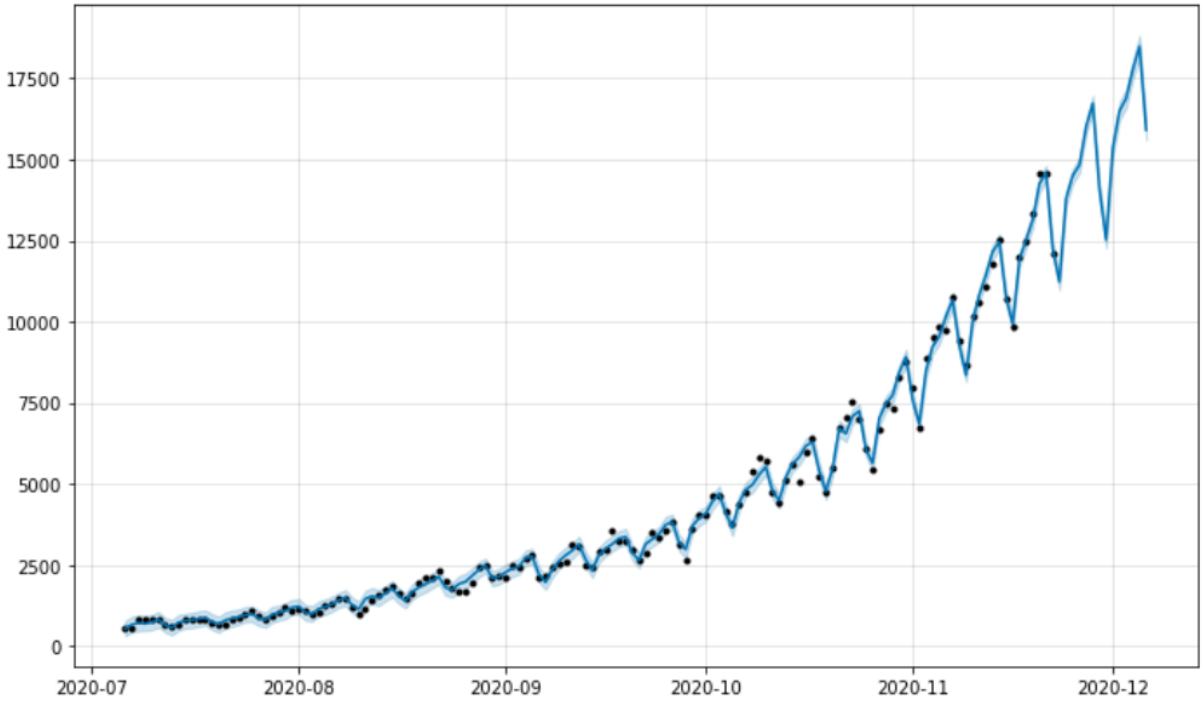


Рис. 8.8. Приклад прогнозування кількості щоденних нових хворих на коронавірус в Україні у 2020 р. з використанням моделі Facebook Prophet (з ноутбука)

На рис. 8.9 наведено приклад моделювання курсу біткоїна різними моделями без ретельного тюнінгу, який показує, що модель FB Prophet демонструє вищу точність, але і її треба налаштовувати більш ретельно, про що можна почитати у статтях [52,48]

name_model	type_data	r2_score	rmse	mape
Prophet_7_days_3_order	valid	0.948961	377.283507	2.724119
Prophet_7_days_12_order	valid	0.932915	432.541405	3.095432
Linear Regression	valid	0.546218	1124.963248	7.598622
MLP Regressor	valid	0.591935	1066.791733	7.632433
Prophet_4_days_3_order	valid	0.56331	1103.574012	7.692877
Prophet_4_days_12_order	valid	0.494738	1187.060858	8.28186
KNeighbors Regressor	valid	0.37277	1322.599121	8.970068
Random Forest Regressor	valid	0.321687	1375.402431	9.175315
Prophet_365_days_3_order	valid	0.043321	1633.419954	11.046927
Prophet_30_days_3_order	valid	-0.034667	1698.693218	11.329504
Prophet_30_days_12_order	valid	-0.508503	2051.104402	14.480213
ARIMA_auto	valid	-1.043162	2387.075372	15.823972
Support Vector Machines	valid	-1.222799	2489.801995	16.240231
Linear SVR	valid	-1.434648	2605.75055	17.124106
Prophet_365_days_12_order	valid	-2.839393	3272.246318	18.751056
XGB Regressor	valid	-9.462468	5401.716501	37.091622
Bagging Regressor	valid	-74.651107	14525.208186	101.13883

Рис. 8.9. Прогнозування курсу біткоїну різними моделями (з ноутбука)

8.4 Інтелектуальний аналіз та прогнозування часових рядів

Технології інтелектуального аналізу та прогнозування часових рядів дозволяють вирішувати багато прикладних задач.

Наведемо короткий огляд конкурсів Kaggle з цієї тематики з реалістичними постановками задач:

- Прогнозування взаємодій між молекулами;
- Прогнозування настання етапу сну та пробудження людини;
- Прогнозування курсу валют, у т.ч. криптовалют, або акцій;
- Прогнозування кількості хворих чи померлих на коронавірус у заданому регіоні;
- Прогнозування ціни на вживане авто;
- Прогнозування якості річкової води;
- Прогнозування метеорологічних явищ.

У конкурсі, Open Problems – Single-Cell Perturbations, учасники мають передбачити, які зміни в генному вираженні викликають невеликі молекули у різних типах клітин. Їх робота сприятиме розробці методів передбачення реакції клітин на дії лікарських засобів, що може мати важливі застосування у відкритті ліків та основній біології. Методи передбачення дії лікарських засобів вже розвиваються, проте бракує адекватних наборів даних для визначення їх загальної придатності. Конкурс спрямований на заповнення цієї прогалини і на пошук ефективних методів передбачення реакції клітин на хімічні збурення

Перше місце посіло рішення, яке використовувало методи обробки природної мови та біологічних вбудовувань для збагачення вхідного простору ознак. Починаючи з використання вбудовувань біологічних термінів,

автор виявив, що використання вбудовувань ChemBERTa для SMILES кодування структур хімічних сполук принесло значне покращення результатів. Також автор розробив техніки додавання даних, включаючи статистику диференційної експресії за клітинним типом та назвою хімічних сполук. Моделі використовували LSTM, GRU та 1-d CNN архітектури, оптимізатор Adam та функції втрат MSE, MAE, LogCosh та BCE для збалансованості навчання. Учасник досліджував також стійкість моделі, включаючи варіанти обробки даних з різними об'ємами тренувальної вибірки.

У конкурсі, Child Mind Institute - Detect Sleep States, учасники розробляють модель для визначення періодів сну та пробудження на основі даних з акселерометра, що носиться на зап'ясті. Це дозволить дослідникам проводити більш надійні та масштабні дослідження сну, що має важливе значення для здоров'я та розвитку. Результати конкурсу можуть покращити розуміння зв'язку між сном та настроєм у дітей, а також сприяти розробці персоналізованих стратегій втручання для дітей з проблемами у цій сфері.

Рішення, яке принесло перше місце в конкурсі, використовує модель, що базується на структурі з використанням зваженого глибокого навчання, такого як CNN і GRU. Додатково, воно використовує фільтр для періодичних сигналів та пост-обробку результатів для оптимізації метрик. Розв'язок також включає удосконалення в підготовці даних, такі як розподіл даних на щоденні частини з врахуванням часових зсувів (рис. 8.10).

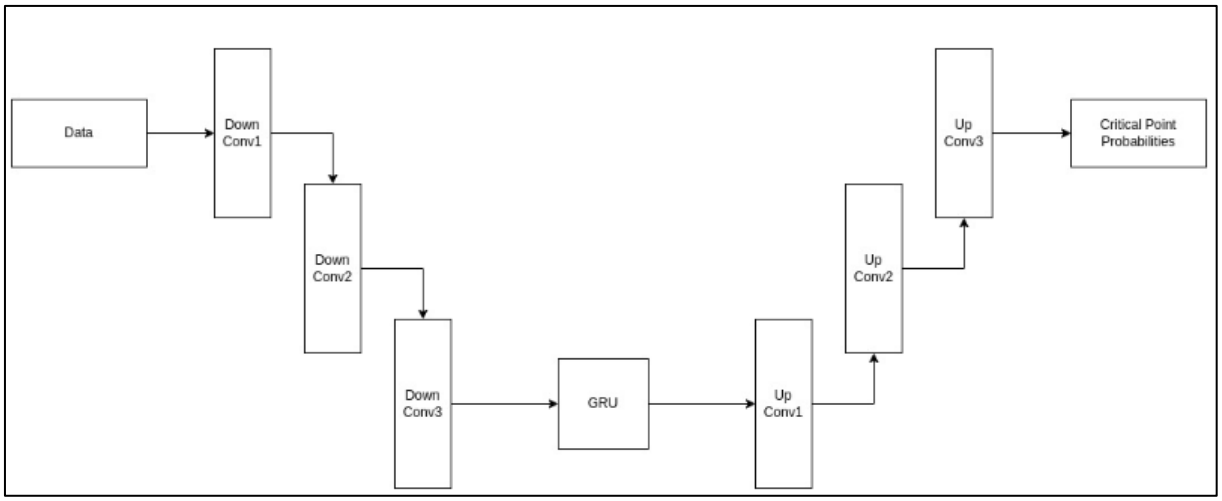


Рисунок 8.10 – Структура моделі рішення

GoDaddy - Microbusiness Density Forecasting, у цьому конкурсі метою є прогнозування щомісячної щільності мікропідприємств у певній області США. Учасники розроблятимуть точну модель на основі даних на рівні округів США. Ця робота допоможе політикам отримати уявлення про мікропідприємства, невеликі сутності, і виробляти нові політики та програми для покращення їхнього успіху та впливу.

Перше місце в конкурсі посіло рішення де використали просту лінійну регресію, а друге місце використало модель XGBoost. На відміну від них

учасник, що зайняв 3-тє місце у конкурсі використовував модель прогнозування мультиплікаторів з використанням TensorFlow GRU для передбачення щільності мікропідприємств у США на майбутні місяці. Він скористався досвідом з попередніх конкурсів та використав методику передбачення мультиплікаторів, що дозволило значно покращити результати. Учасник провів аналіз та обробку даних, а також оптимізував модель GRU з використанням GroupKFold для кращої узгодженості результатів крос-валідації. Після прогнозування він вдосконалив результати за допомогою постпроцесингу (рис. 8.11).

Рисунок 8.11– Створення 18 часових рядів для кожного округу. Використано 13 місяців для навчання GRU та передбачення наступних 5 місяців. Після цього доступно 56,000 часових рядів для тренування моделі. Рішення 3rd Place - Predict Multipliers with GRU

Крім того, є чимало інших прикладів розв'язання реальних задач з використанням інтелектуального аналізу та прогнозування часових рядів:

На рис. 8.8 наведена інфографіка інструментарію, згаданого у розділі 8 в цілому, у нотації рис. 1.2.

Рисунок 8.8 – Інфографіка аналізу та прогнозування часових рядів

Можливі теми практичних і лабораторних завдань

Тема № 1. Ідентифікація моделі часових рядів та розв'язання задачі прогнозування.

Метою роботи є вивчення інформаційних технологій і Python-бібліотек ідентифікації моделі часових рядів та розв'язання задачі прогнозування й опанування практичних навичок застосування деяких із них на прикладі одного із датасетів Kaggle чи за даними, завантаженими через API.

План заняття:

1. Вибрати датасет (див. розд. 1 або приклади нижче).
2. Завантажити дані.
3. Або сформувати дві різні вибірки (наприклад, усі дані разом та якась їх найбільш цікава частина) і промоделювати моделлю одного типу, наприклад FB Prophet, або взяти один ряд і промоделювати двома моделями, наприклад ARIMA і FB Prophet. Спробувати удосконалити налаштування моделей, щоб підвищити точність. Зробити висновки яка модель точніша чи для якої частини датасету вона є точнішою.
4. Зробити прогноз даних.
5. Побудувати графіки і візуально оцінити якість прогнозу та адекватність моделі.
6. Навести посилання на створений ноутбук, який містить усі результати роботи, та зробити висновки про те яка модель – оптимальна і яка в неї похибка чи точність.

Рекомендується використати заготовку ноутбука: COVID-19 UA: one region forecasting.

Можна послухати відео з коментарями:

- Аналіз та прогнозування часових рядів на прикладі курсу біткоїна;
- Розвідувальний аналіз даних (EDA) на прикладі курсу біткоїна;
- Прогнозування часових рядів на прикладі моделювання коронавірусу в Україні - Курс AI-ML-DS Training;
- Прогнозування часових рядів. Ч II. На прикладі моделювання коронавірусу в одній області України;
- Приклад розв'язання реальної задачі – прогнозування кількості нових хворих на коронавірус в Україні;
- Приклад розв'язання реальної задачі – прогнозування кількості COVID-госпіталізацій в Україні;
- Моделі-регресори табличних даних на прикладі моделювання якості води - Курс AI-ML-DS Training;
- Дерева рішень-регресори на прикладі моделювання якості води - Курс AI-ML-DS Training;

Приклади ноутбуків:

- COVID in UA: Prophet with 4, Nd seasonality
- COVID-19 UA: one region forecasting
- AI-ML-DS Training. L1T : COVID in UA - Prophet
- COVID-19 : Hospitalizations in Ukraine
- COVID-19 in Ukraine: EDA & Forecasting

Контрольні питання

1) Що включає в себе розвідувальний аналіз даних часових рядів?
2) Які аспекти аналізу аномалій в часових рядах важливі для виявлення непередбачуваних відхилень?
3) Які методи використовуються для аналізу закону розподілу даних в часових рядах?
4) Що таке стаціонарність у контексті аналізу часових рядів і чому вона важлива?
5) Як визначається сезонність в часових рядах і як вона впливає на моделювання?
6) Які методи використовуються для інженерії ознак часових рядів?
7) Що включає в себе формування тренувального і валідаційного датасетів в задачах прогнозування часових рядів?
8) Які основні характеристики моделей ARIMA і як вони використовуються для прогнозування часових рядів?
9) Які особливості має модель Facebook Prophet і в чому її переваги у порівнянні з іншими методами?
10) Які моделі нейромереж використовуються для прогнозування часових рядів і як вони працюють?

ЛІТЕРАТУРА

1. Yuriev, S., Rodinkova, V., Mokin, V., Varchuk, I., Sharikadze, O., Marushko, Y., ... & Kurchenko, A. (2023). Molecular sensitization pattern to house dust mites is formed from the first years of life and includes group 1, 2, Der p 23, Der p 5, Der p 7 and Der p 21 allergens. *Clinical and Molecular Allergy*, *21*(1), 1-11.
2. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с. – Режим доступу: https://moodle.znu.edu.ua/pluginfile.php/593075/mod_folder/intro/Базовий%20підручник_2%20%28Кононова%20К.%20Ю.%20Машинне%20навчання%20-%20методи%20та%20моделі%29.pdf – мова R
3. Machine learning: стартовий курс : електронний навчальний посібник / Штовба С.Д., Козачко О.М. – Вінниця : ВНТУ, 2020. – 81 с.
4. Гороховатський В.О., Творошенко І.С. Методи інтелектуального аналізу та оброблення даних: навч. посібник. – Харків: ХНУРЕ, 2021. – 92 с. - Режим доступу: https://openarchive.nure.ua/server/api/core/bitstreams/2e55d639-52fd-48d9-b7b7-14989f49f291/content - Prolog
5. Методи та системи штучного інтелекту: навч. посібник / А.С. Савченко, О.О. Синельников. – К.: НАУ, 2017. – 176 с. – Режим доступу: https://pdf.lib.vntu.edu.ua/books/2020/Savchenko_2017_176.pdf - Prolog
6. Машинне навчання: комп'ютерний практикум з дисципліни «Машинне навчання» [Електронний ресурс]: навч. посіб. для студ. спеціальності 121 «Інженерія програмного забезпечення» (освітня програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»)/ Л.М. Олещенко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані. – Київ: КПІ ім. Ігоря Сікорського, 2022. – 92 с.
7. Басюк Т.М., Литвин В.В., Захарія Л.М., Кунанець Н.Е. Машинне навчання: Навчальний посібник призначений для студентів, що навчаються за першим (бакалаврським) рівнем вищої освіти за спеціальностями галузі знань 12 „Інформаційні технології". – Видавництво «Новий Світ - 2000», 2019. – 335 с.
8. Інтелектуальний аналіз даних: Комп'ютерний практикум [Електронний ресурс] : навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки та інформаційні технології», спеціалізацій «Інформаційні системи та технології проектування», «Системне проектування сервісів» / О. О. Сергеєв-Горчинський, Г. В. Іщенко ; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 1,72 Мбайт). –Київ : КПІ ім. Ігоря Сікорського, 2018. –

73 с. – Режим доступу: https://ela.kpi.ua/bitstream/123456789/24971/1/Komp_prakt.pdf - про WEKA

9. Інтелектуальний аналіз даних та машинне навчання. Частина 1. Базові методи та засоби аналізу даних / Я. В. Іванчук, В. І. Месюра, А. А. Яровий, О. Д. Манжілевський – Вінниця : ВНТУ, 2021. – 69 с.

10. Kaggle Notebook «50 Tips: Data Science (tabular data) for beginner» : веб-сайт. URL: https://www.kaggle.com/code/vbmokin/50-tips-data-science-tabular-data-for-beginner/notebook

11. Kaggle Notebook «50 Advanced Tips: Data Science for tabular data» : веб-сайт. URL: https://www.kaggle.com/code/vbmokin/50-advanced-tips-data-science-for-tabular-data/notebook

12. Kaggle Notebook «santander-eda-to-model-comparison» : веб-сайт. URL: https://www.kaggle.com/davidmarkalbert/santander-eda-to-model-comparison

13. Kaggle Notebook «Santander EDA and Prediction» : веб-сайт. URL: https://www.kaggle.com/gpreda/santander-eda-and-prediction

14. Kaggle Notebook «santander-eda-to-model-comparison» : веб-сайт. URL: https://www.kaggle.com/code/davidmarkalbert/santander-eda-to-model-comparison/notebook

15. Kaggle Notebook «Crypto - BTC : Advanced Analysis & Forecasting» : веб-сайт. URL: https://www.kaggle.com/code/vbmokin/crypto-btc-advanced-analysis-forecasting

16. Мокін Б.І. Методологія та організація наукових досліджень : електронний навчальний посібник / Б. І. Мокін, О. Б. Мокін. – 2-е вид., змін. та доп. – Вінниця : ВНТУ, 2015. – 317 с

17. В. Б. Мокін, А. В. Лосенко, і М. В. Дратований, «ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЦІН НА ВЖИВАНІ АВТОМОБІЛІ», *Вісник ВПІ*, вип. 6, с. 62–72, Груд. 2019.

18. Kaggle Notebook «Crypto - BTC : Advanced EDA» : веб-сайт. URL: https://www.kaggle.com/code/vbmokin/crypto-btc-advanced-eda/notebook

19. Giatsidis, C., Malliaros, F. D., Tziortziotis, N., Dhanjal, C., Kiagias, E., Thilikos, D. M., & Vazirgiannis, M. (2016). A k-core decomposition framework for graph clustering. *arXiv preprint arXiv:1607.02096*.

20. Vepa, A., Saleem, A., Rakhshan, K., Daneshkhah, A., Sedighi, T., Shohaimi, S., ... & Chakrabarti, P. (2021). Using machine learning algorithms to develop a clinical decision-making tool for COVID-19 inpatients. *International journal of environmental research and public health*, *18*(12), 6228.

21. Методологія та організація наукових досліджень : підручник – вид. 3-е, змін. та доп. [Електронний ресурс] / Б.І.Мокін, О.Б.Мокін, В.Б. Мокін. – Вінниця: ВНТУ, 2023. – (PDF, 230 с.)

22. Kaggle Notebook «Heart Disease - Automatic AdvEDA & FE & 20 models» : веб-сайт. URL: https://www.kaggle.com/code/vbmokin/heart-disease-automatic-adveda-fe-20-models/notebook

23. How to Interpret Black Box Models using LIME (Local Interpretable Model-Agnostic Explanations) : веб-сайт. URL: https://www.freecodecamp.org/news/interpret-black-box-model-using-lime/

24. Kaggle Notebook «Tutorial : Classification models» : веб-сайт. URL: https://www.kaggle.com/code/vbmokin/tutorial-classification-models

25. Kaggle Notebook «Tutorial : Ensembles of classification models»: веб-сайт. URL: https://www.kaggle.com/code/vbmokin/tutorial-ensembles-of-classification-models

26. МАШИННЕ НАВЧАННЯ ПРОСТИМИ СЛОВАМИ. ЧАСТИНА 1 : веб-сайт. URL: http://www.mmf.lnu.edu.ua/ar/1739

27. В. Б. Мокін і М. В. Дратований, «ІНТЕЛЕКТУАЛЬНИЙ МЕТОД З ПІДКРІПЛЕННЯМ СИНТЕЗУ ОПТИМАЛЬНОГО КОНВЕЄРУ ОПЕРАЦІЙ ПОПЕРЕДНЬОГО ОБРОБЛЕННЯ ДАНИХ У ЗАДАЧАХ МАШИННОГО НАВЧАННЯ», *НаукПраці ВНТУ*, вип. 4, Груд 2022.

28. Лабораторний практикум з дисципліни «Комп'ютерна лінгвістика» для студентів спеціальності 126 – «Інформаційні системи та технології» : електронний лабораторний практикум комбінованого (локального та мережного) використання [Електронний ресурс] / Бісікало О. В. , Севастьянов В. М., Богач І. В. – Вінниця : ВНТУ, 2022. – 102 с.

29. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

30. Keras 3 API documentation : веб-сайт. URL: https://keras.io/activations/

31. Wikipedia «Activation function»: веб-сайт. URL: https://en.wikipedia.org/wiki/Activation_function

32. Mokin V. B. Development of intelligent technologies for energy-saving optimization of grain elevator operation using neural network models and reinforcement learning methods [Text] / V. B. Mokin, M. V. Dratovany, A. Lukhverchyk // The 5th International scientific and practical conference «Scientific progress: innovations, achievements and prospects», Munich, Germany, February 6-8, 2023. – Munich, Germany : MDPC Publishing, 2023. – Pp. 138-144.

33. Дратований М. В. Інтелектуальний метод з підкріпленням синтезу оптимального конвеєру операцій попереднього оброблення даних у задачах машинного навчання [Електронний ресурс] / М. В. Дратований, В. Б. Мокін // Наукові праці ВНТУ. – 2022. – № 4. – Режим доступу: https://praci.vntu.edu.ua/index.php/praci/article/view/670.

34. Мокін В. Б. Системний аналіз розмірів фрагмента зображень аерофотозйомки сільськогосподарських угідь для пошуку аномалій у них методами машинного навчання / В. Б. Мокін, Д. М. Грузман, С. О. Довгополюк, А. О. Лотоцький // Вісник Вінницького політехнічного інституту. – Вінниця: ВНТУ, 2019.– No3. – С. 75-85

35. Kaggle Notebook «Convolutional Neural Network (CNN) Tutorial»: веб-сайт. URL: https://www.kaggle.com/code/kanncaa1/convolutional-neural-network-cnn-tutorial/notebook

36. Kaggle Notebook «Convolutional Neural Network (CNN) Tutorial»: веб-сайт. URL: https://www.kaggle.com/code/rafetcan/convolutional-neural-network-cnn-tutorial/notebook

37. Я. О. Ісаєнков і О. Б. Мокін, «АНАЛІЗ ГЕНЕРАТИВНИХ МОДЕЛЕЙ ГЛИБОКОГО НАВЧАННЯ ТА ОСОБЛИВОСТЕЙ ЇХ РЕАЛІЗАЦІЇ НА ПРИКЛАДІ WGAN», *Вісник ВПІ*, вип. 1, с. 82–94, Берез. 2022.

38. YOLO: Algorithm for Object Detection Explained : веб-сайт. URL: https://www.linkedin.com/pulse/yolo-algorithm-object-detection-explained-arnab-mukherjee/

39. A Guide to YOLOv8 in 2024Read more at: веб-сайт. URL: https://viso.ai/deep-learning/yolov8-guide/

40. YOLOv8 : Comprehensive Guide to State Of The Art Object Detection: веб-сайт. URL: https://learnopencv.com/ultralytics-yolov8/

41. Інтелектуальний аналіз даних. Частина 1 / М.В. Талах, В.В. Дворжак – Чернівці: Технодрук, 2022. – 367 с.

42. How to process textual data using TF-IDF in Python: веб-сайт. URL: https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/

43. Kaggle Notebook «NLP - EDA, Bag of Words, TF IDF, GloVe, BERT»: веб-сайт. URL: https://www.kaggle.com/code/vbmokin/nlp-eda-bag-of-words-tf-idf-glove-bert

44. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

45. Bondalietov, K., Mokin, V. (2023). Notation System for Comparing and Synthesis of Intelligent Key Phrase Extraction Methods for Ontological Models in Information Systems. In: Dovgyi, S., Trofymchuk, O., Ustimenko, V., Globa, L. (eds) Information and Communication Technologies and Sustainable Development. ICT&SD 2022. Lecture Notes in Networks and Systems, vol 809. Springer, Cham. https://doi.org/10.1007/978-3-031-46880-3_11

46. Kaggle Notebook «NLP for WR: Summarizing using BERT, GPT2, XLNET»: веб-сайт. URL: https://www.kaggle.com/code/vbmokin/nlp-for-wr-summarizing-using-bert-gpt2-xlnet/notebook

47. Документація Facebook Prophet, розділ «Diagnostics»: веб-сайт. URL: https://facebook.github.io/prophet/docs/diagnostics.html

48. В. Б. Мокін, А. В. Лосенко, і А. Р. . Ящолт, «ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ БАГАТОХВИЛЬОВОЇ КІЛЬКОСТІ НОВИХ ВИПАДКІВ ЗАХВОРЮВАНЬ НА КОРОНАВІРУС COVID-19 НА ОСНОВІ МОДЕЛІ PROPHET», *Вісник ВПІ*, вип. 6, с. 65–75, Груд. 2020.

49. Копняк В. Є., Мокін В. Б. Дослідження проблем із гетероскедастичністю даних моніторингу якості атмосферного повітря // Матеріали LII Науково-технічної конференції факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, Вінниця, 21 – 23 червня 2023 р. – Електрон. текст. дані. – 2023. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023/paper/view/18962/15728

50. Шмундяк Д.О., Н.С. Іжаковська, Д.О. Литвиненко, А.О. Судець. Аналіз можливостей Python-бібліотек щодо виявлення аномальних даних у задачі прогнозування стану атмосферного повітря // Матеріали LII Науково-технічної конференції факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, Вінниця, 21 – 23 червня 2023 р. – Електрон. текст. дані. – 2023. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2023/paper/view/18957/15722

51. Wikipedia «Cryptocurrency bubble»: веб-сайт. URL: https://en.wikipedia.org/wiki/Cryptocurrency_bubble

52. Мокін В. Б. Інформаційна технологія аналізу та прогнозування кількості нових випадків на коронавірус SARS-CoV-2 в Україні на основі моделі Prophet / В. Б. Мокін, А. В. Лосенко, А. Р. Ящолт // Вісник Вінницького політехнічного інституту. – 2020. – № 5. – С. 71–83

53. Мокін Б. І. Практикум для самостійної роботи студентів з навчальної дисципліни «Методологія та організація наукових досліджень». Частина 1: від постановки задачі до синтезу та ідентифікації математичної моделі / Б. І. Мокін, В. Б. Мокін, О. Б. Мокін. – Вінниця : ВНТУ, 2018. – 179 с

54. Мокін Б. І. Математичні методи ідентифікації динамічних систем : навчальний посібник / Б. І. Мокін, В. Б. Мокін, О. Б. Мокін. – Вінниця : ВНТУ, 2010. – 260 с

55. Моделювання та оптимізація систем: підручник / [Дубовой В. М., Квєтний Р. Н., Михальов О. І., Усов А.В.] – Вінниця : ПП «ТД «Едельвейс», 2017. – 804 с

56. Легеза В.П. Математичний аналіз. Підручник у 4-х томах. Т.1. – К.: Політехніка, 2019. – 336 с.

57. Kaggle Notebook «Crypto - BTC : Analysis & Forecasting »: веб-сайт. URL: https://www.kaggle.com/code/vbmokin/crypto-btc-analysis-forecasting

58. Glossary of LLM Terms: веб-сайт. URL: https://vectara.com/glossary-of-llm-terms/

59. Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, *72*(1), 37-45.

ДОДАТОК А
ОСНОВИ PYTHON: СИНТАКСИС, ТИПИ ДАНИХ, ОСНОВНІ КОМАНДИ ТА БАЗОВІ БІБЛІОТЕКИ

Рекомендуємо ознайомитись з такою інфографікою з Python:

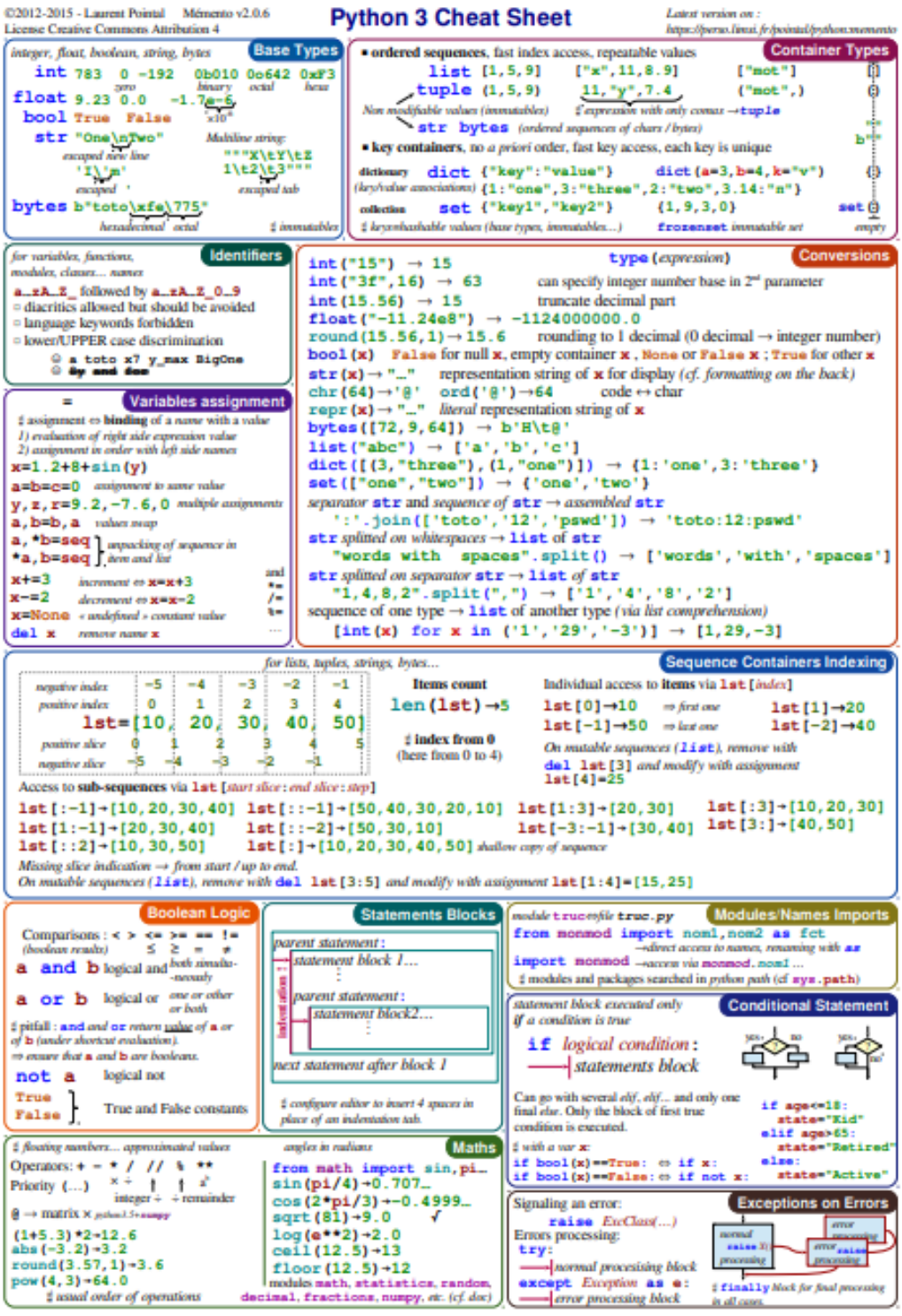


Рис. А.1. Синтаксис Python. Частина I

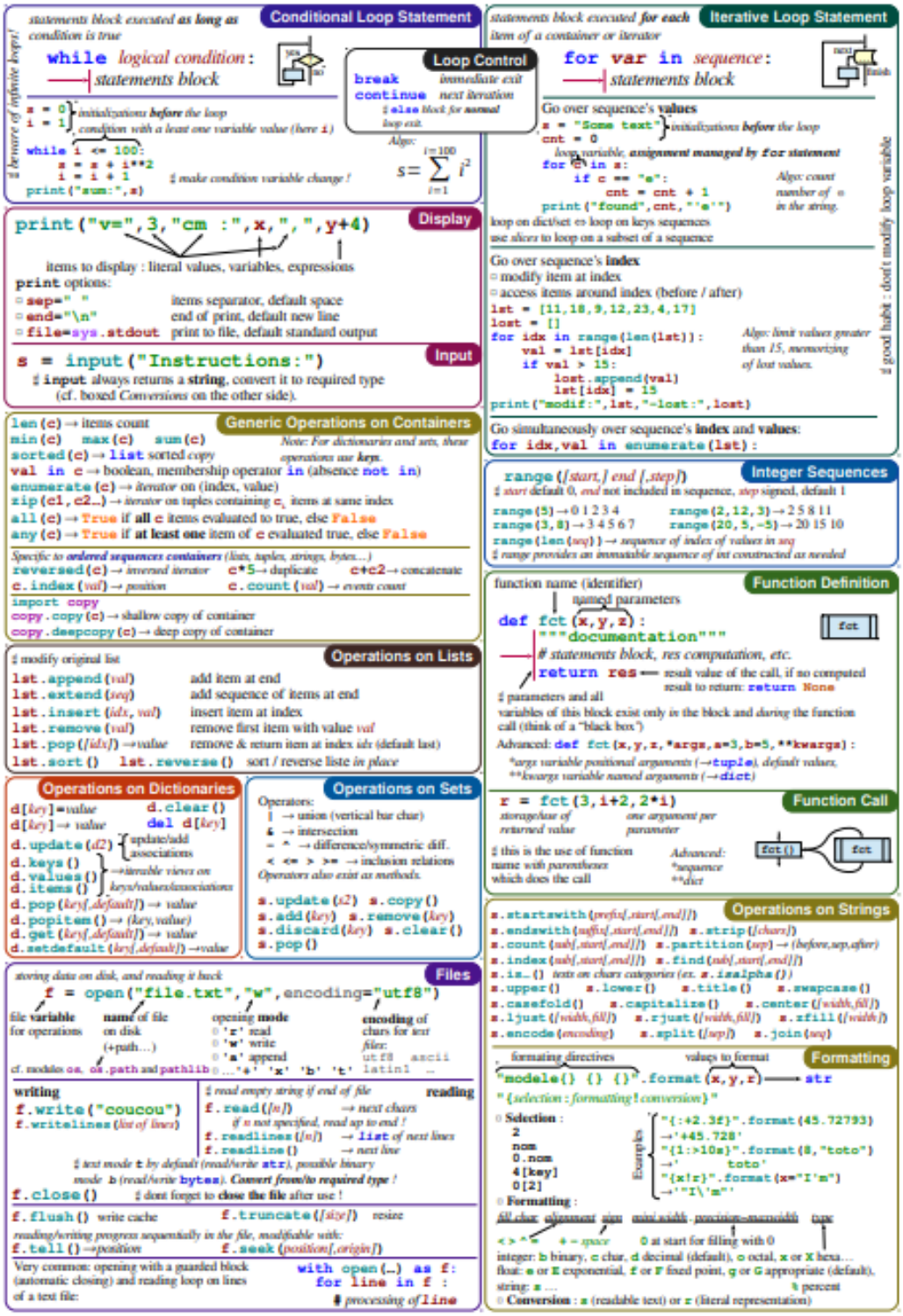


Рис. А.2. Синтаксис Python. Частина II

Також, варто ознайомитись з відкритими матеріалами щодо Python у документації.

Для написання якісних і читабельних програм рекомендуємо ознайомитись із «PEP 8» – це документ, який визначає стандарти оформлення коду для мови програмування Python (з англ. "PEP" – "Python Enhancement Proposal"): рекомендації і правила щодо форматування коду, імен змінних, розташування дужок, відступів тощо.

Корисною є інфографіка про операції з датою і часом.

NumPy – це фундаментальний пакет для наукових обчислень на Python. Підтримуються операції з масивами і матрицями, у т.ч. операції лінійної алгебри, різні математичні та логічні функції, сортування, вибір, введення/виведення, основні статистичні операції тощо.

Корисно вивчити наступну інфографіку основних операцій з даними в бібліотеках NumPy, Pandas та ін.:

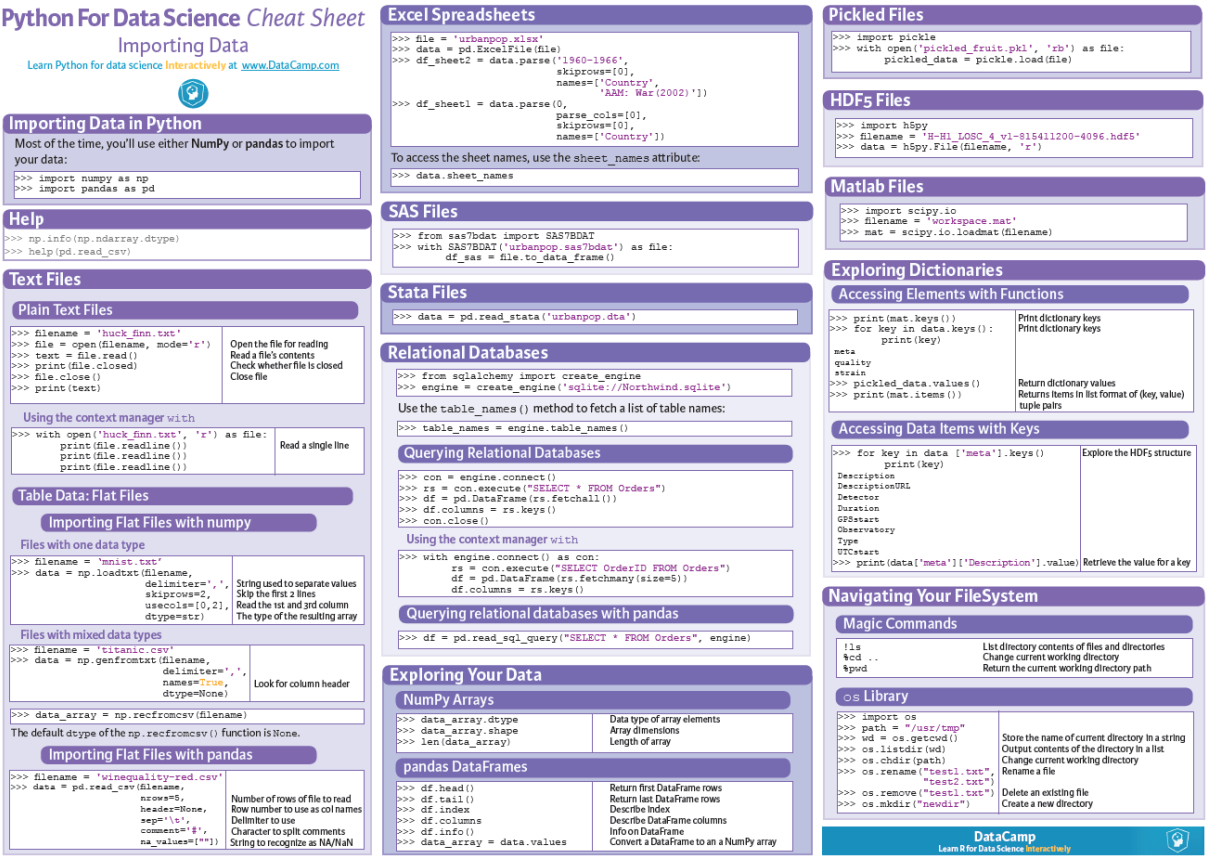


Рис. А.3. Основні команди Python щодо імпорту даних

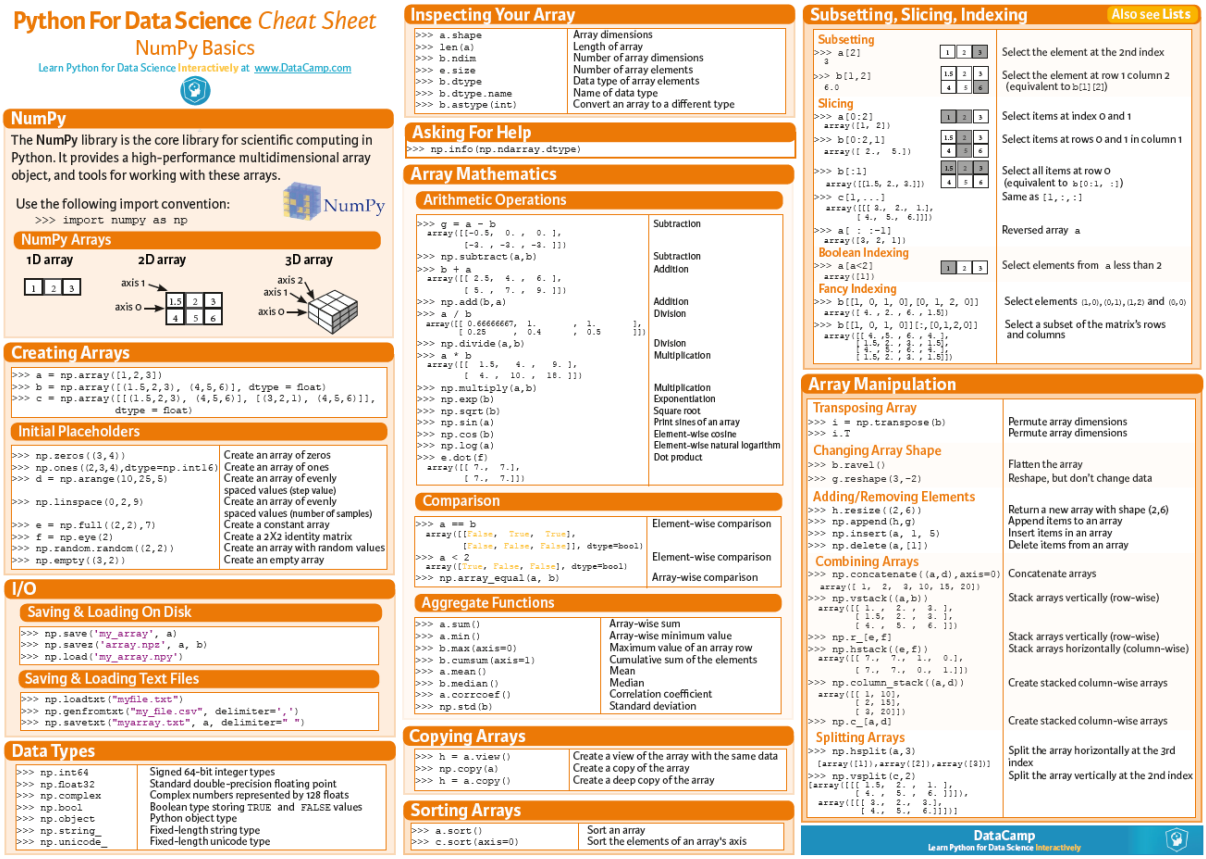


Рис. А.4. Бібліотека NumPy

Більше детально про NumPy див. у посібниках

- Tutorial
- QuickStart
- 100 задач на NumPy

Pandas – це головна базова високошвидкісна бібліотека Python для роботи з табличними даними у вигляді датафреймів.

Більшість бібліотек Python з аналізу даних працюють з інформацією у форматі даних або NumPy, або Pandas. Більш детально про операції в Pandas див. в інфографіці:

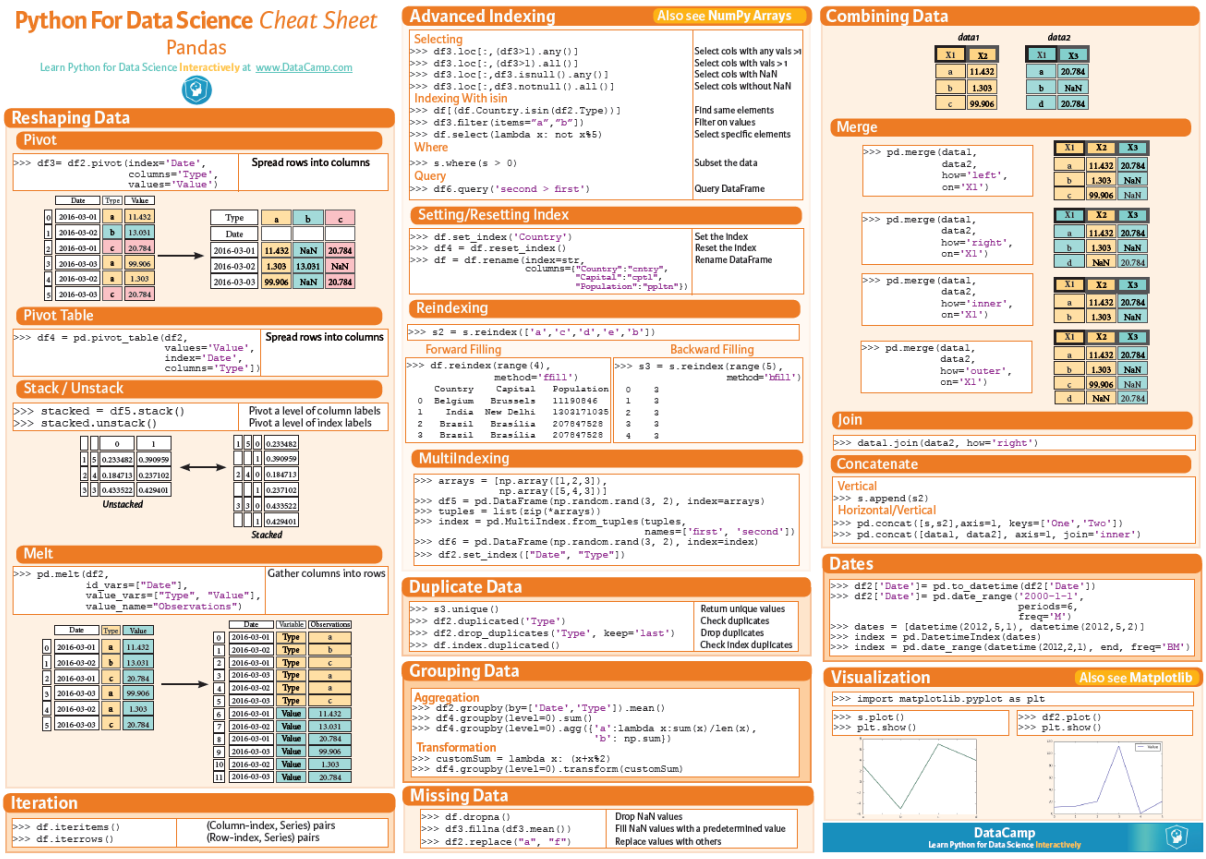


Рис. А.5. Бібліотека Pandas

Важливими операціями Pandas з датафреймами є наступні 3:

- **concat** – приєднання уздовж однієї з осей (стовпець чи рядок, тобто приєднання по вертикалі чи горизонталі)
- **join** – об'єднання зліва, справа чи ін.;
- **merge** – злиття за найрізноманітнішими варіантами.

Більш детально про ці операції див. у документації Pandas (корисно хоча б 1 раз проглянути весь цей файл документації).

ДОДАТОК Б
ПОБУДОВА ВЛАСНОГО ДАТАСЕТУ У СЕРЕДОВИЩІ KAGGLE

Для *створення власних датасетів* у середовищі Kaggle рекомендується застосовувати такий алгоритм:

1. Створити таблицю даних у редакторі, який може зберігати файли у форматі csv (наприклад, MS Excel, Google.Table, Calc Apache OpenOffice або LibreOffice, Spreadsheets WPS Office тощо) і ввести в неї усі необхідні дані та здійснити очищення форматування та рефакторинг даних, щоб підготувати до зручного вигляду для автоматичного оброблення:
   - кожна таблиця – на окремому листі (англ. «sheet»);
   - комірки не об'єднувати,
   - усі стовпці назвати словами з маленькими англійськими літерами чи цифрами, намагатись уникати спецсимволів;
   - якщо у назві більше одного слова, тоді їх поєднувати символом «_», а не – пробілами): «body_temperature»;
   - дати оптимально подавати у форматі «Короткий формат дати» (наприклад: 04.01.24), уникати варіанту дати, де пропущені нулі в потрібних розрядах «4.1.24», оскільки для них на Python немає готових функцій і треба розробляти нову спеціальну функцію, натомість у редакторах, на кшталт MS Excel, є зручні функції, які дозволяють легко виконати перетворення 4.1.24 на 04.01.24;
2. Зберегти кожну таблицю в окремий csv-файл (пам'ятайте, що кожен лист таблиці зберігається в окремий csv-файл);
3. Визначити кодування символів чи змінити його на точно відоме. Відкрити створений csv-файл у редакторі «Sublime Text» чи аналогічному. Пересвідчитись, що усі символи нормально прочитались, або перекодувати їх, натиснувши в меню «File/Reopen with Encoding» і вибрати необхідне кодування зі списку, а потім – зберегти файл. Зазвичай, вибирають UTF-8 або інші. Тоді, у разі зчитування даних у Kaggle точно буде відомо яке кодування вказувати (див. приклад у пораді «Tip 2.3» в [10]). Ще існують сервіси, які дозволяють ідентифікувати кодування тексту, наприклад, Python-бібліотека chardet. У разі роботи з датасетами важливо точно знати кодування, щоб гарантувати його коректне зчитування, інакше можна втратити інформацію або він взагалі не прочитається.
4. Створити у Kaggle Dataset новий датасет: натиснути на кнопку «New Dataset» та перетягнути у нове вікно створений у п.2 файл.
5. Присвоїти датасету основну назву. Саме вона буде фігурувати далі у веб-посиланні в кінці адреси (англійські літери через дефіс) і потім змінити її не можна!), додаткову уточнюючу назву та описати датасет, виконуючи подальші інструкції та шаблони (див. приклади https://www.kaggle.com/vbmokin/datasets).

Щодо Kaggle важливо знати, що там використовується вбудована система перевірки на плагіат і він не дає зберегти датасет, який вже є в системі! Тому, якщо маєте намір створити власний датасет, то він має бути дійсно

оригінальним чи хоча б містити якесь оброблення наявного, щоб не було збігу в даних.

Систематизуємо операції з побудови датасету у Kaggle у вигляді інфографіки у системі координат *S(I)* (рис. Б.1).

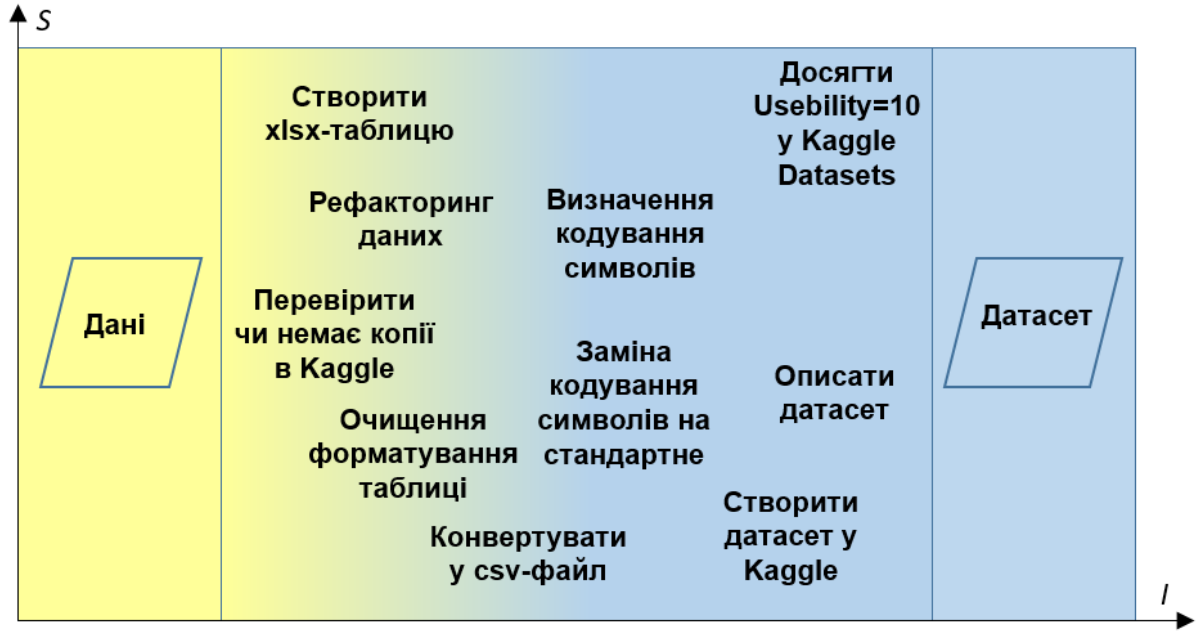


Рисунок Б.1 – Інфографіка операцій з побудови датасету у Kaggle

Важливо максимально використовувати відкриті дані. Це корисно ще й тим, що тоді легше публікувати результати (з посиланням на першоджерела інформації).

Сервіси Amazon для інженерії даних. SageMaker
Стисло за даними жпт та їх документацією.

Сервіси Google Cloud для інженерії даних
Стисло за даними жпт та їх документацією.

Сервіси Kaggle для інженерії даних: побудова та використання датасетів і ноутбуків
Теорія і практика.
Мій опис цього є отут:

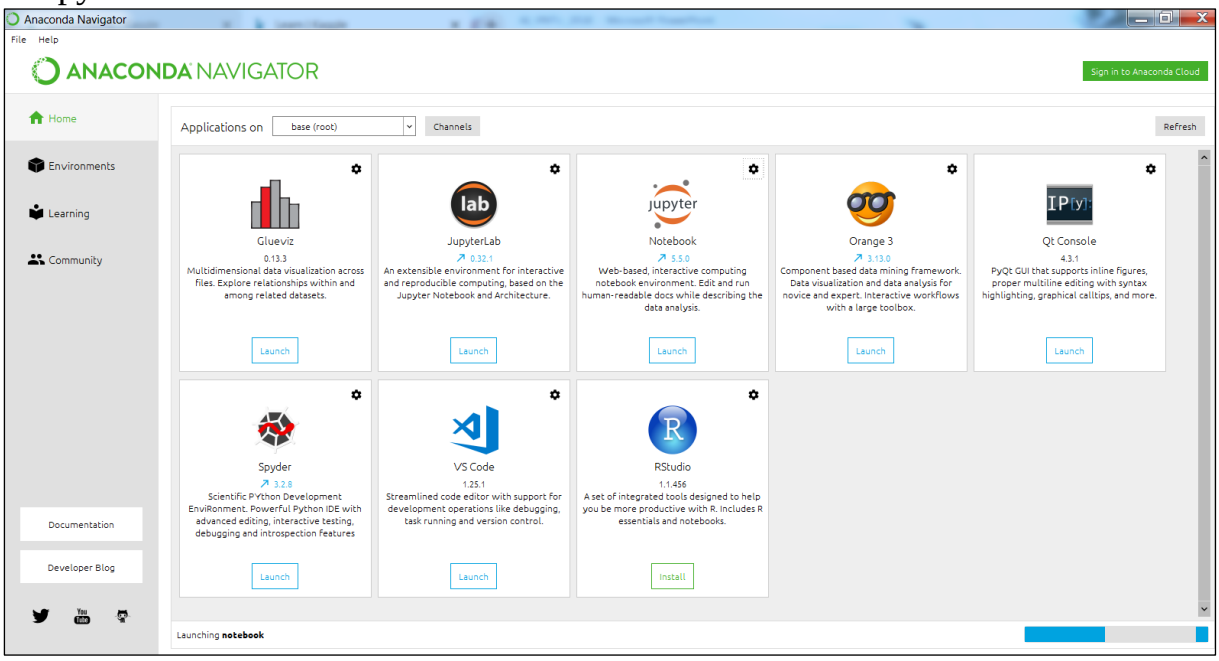
1. https://www.youtube.com/watch?v=3W2ekdE2eqM&list=PL4DHq-xU-ebUiB6T6vjd0SoDha4GOm8zV&index=5
2. https://www.youtube.com/watch?v=KSEs5w-u9sQ&list=PL4DHq-xU-ebUiB6T6vjd0SoDha4GOm8zV&index=6

3. https://www.youtube.com/watch?v=6mmBpXFz-kw&list=PL4DHq-xU-ebUiB6T6vjd0SoDha4GOm8zV&index=7

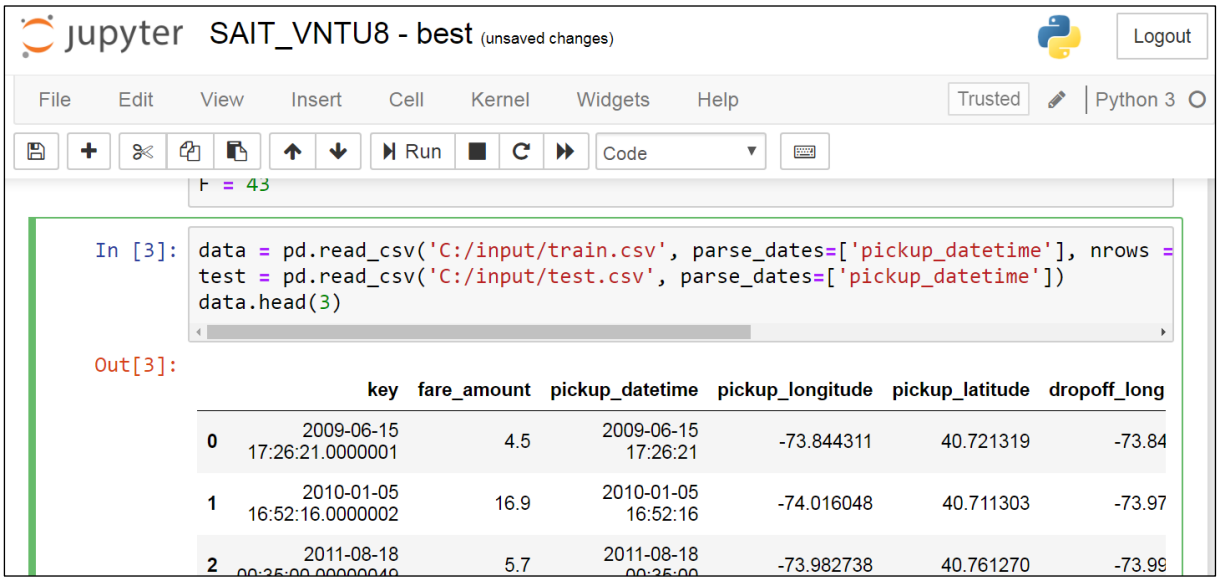
PyCharm – можливості та обмеження. Склад пакету

Visual Studio – можливості та обмеження. Склад пакету

Anaconda – можливості та обмеження. Склад пакету. Spider, Jupyter Lab, Jupyter Notebook



Про Jupyter Notebook більш детально і про те, що це – оптимальний варіант, бо його редактор є в усіх вище названих середовищах або схожий на них

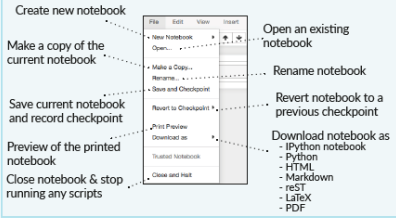


Python For Data Science *Cheat Sheet*

Jupyter Notebook

Learn More Python for Data Science Interactively at www.DataCamp.com

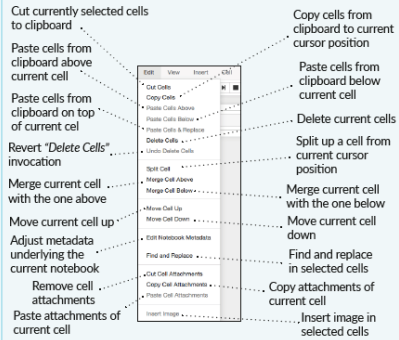
Saving/Loading Notebooks



Writing Code And Text

Code and text are encapsulated by 3 basic cell types: markdown cells, code cells, and raw NBConvert cells.

Edit Cells



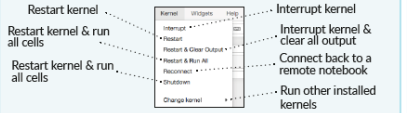
Insert Cells

Working with Different Programming Languages

Kernels provide computation and communication with front-end interfaces like the notebooks. There are three main kernels:

IPython | IRkernel | Uulia

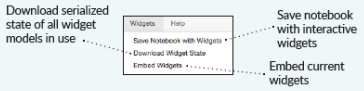
Installing Jupyter Notebook will automatically install the IPython kernel.



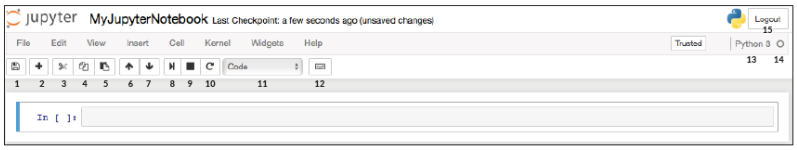
Widgets

Notebook widgets provide the ability to visualize and control changes in your data, often as a control like a slider, textbox, etc.

You can use them to build interactive GUIs for your notebooks or to synchronize stateful and stateless information between Python and JavaScript.



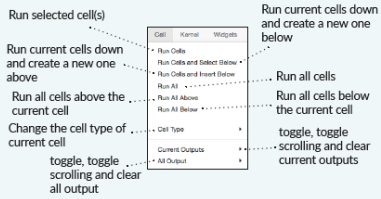
Command Mode:



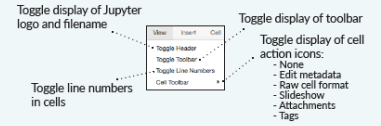
Edit Mode:

1. Save and checkpoint
2. Insert cell below
3. Cut cell
4. Copy cell(s)
5. Paste cell(s) below
6. Move cell up
7. Move cell down
8. Run current cell
9. Interrupt kernel
10. Restart kernel
11. Display characteristics
12. Open command palette
13. Current kernel
14. Kernel status
15. Log out from notebook server

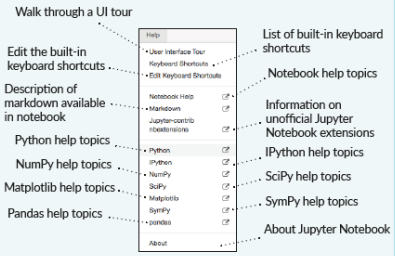
Executing Cells



View Cells



Asking For Help



DataCamp
Learn Python for Data Science Interactively

ДОДАТОК В
ПРИКЛАДИ ПОСТАНОВКИ ЗАДАЧ З МАШИННОГО НАВЧАННЯ ТА ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

Наведемо приклади постановки задач з машинного навчання та інтелектуального аналізу даних на прикладі реальних завдань, а також на прикладі Kaggle датасетів і Kaggle змагань.

1. Прогнозування продажів у роздрібній торгівлі: На основі історичних даних про продажі, погоду, свята та інші фактори створюється модель для прогнозування майбутніх продажів у магазинах. Результати такої моделі можуть допомогти управлінцям приймати рішення щодо запасів, маркетингових кампаній тощо.
2. Класифікація тексту: Задача полягає в розпізнаванні категорії тексту. Наприклад, класифікація електронних листів на "спам" та "не спам", або аналіз настроїв у коментарях на соціальних мережах.
3. Виявлення аномалій у медичних даних: За допомогою алгоритмів машинного навчання аналізуються медичні дані, щоб виявити аномалії або потенційні хвороби на ранній стадії. Це може бути корисно для діагностики різних захворювань, наприклад, раку чи серцевих захворювань.
4. Рекомендаційні системи: Завдання полягає у розробці систем, які рекомендують користувачам товари, послуги або контент на основі їхніх попередніх взаємодій. Наприклад, рекомендації фільмів на основі історії перегляду користувача.
5. Розпізнавання образів: Метою є розпізнавання об'єктів або патернів на зображеннях. Це може бути використано для автоматичного розпізнавання номерних знаків на фотографіях або для аналізу медичних зображень для виявлення патологій.

Крім того, платформа Kaggle надає доступ до різноманітних датасетів та організує змагання, які можуть слугувати як приклади реальних завдань з машинного навчання:

1) Завдання Titanic: Machine Learning from Disaster полягає в розробці моделі машинного навчання для передбачення того, виживе пасажир або ні, на основі інформації про пасажирів, таку як вік, стать, клас квитка тощо. Головна мета - покращити точність передбачень та розробити ефективні моделі, що допоможуть визначити ключові фактори, що впливають на виживання пасажирів у катастрофі. Оцінка роботи полягає в порівнянні передбачених результатів з фактичною виживаністю пасажирів та визначенні ефективних методів машинного навчання для цієї задачі.
2) У завданні House Prices: Advanced Regression Techniques потрібно розробити модель для прогнозування цін на житло на основі різноманітних фізичних та географічних характеристик нерухомості. Метою є створення ефективної моделі, яка може точно передбачати вартість житла з урахуванням різноманітних факторів, таких як площа, кількість кімнат, розташу-

вання тощо. Оцінка роботи полягає у порівнянні прогнозованих цін з фактичними ринковими цінами на нерухомість та розробці оптимальних методів регресійного аналізу для цієї задачі.

3) У завданні Sentiment Analysis on Movie Reviewsучасники мають розпізнати настрій текстових відгуків про фільми та класифікувати їх як позитивні, негативні або нейтральні. Метою є розробка ефективної моделі для аналізу настрою текстів, яка зможе автоматично визначати емоційне відношення до фільмів на основі текстових відгуків. Оцінка роботи полягає у порівнянні передбачених настроїв з фактичними емоційними відгуками та розробці ефективних методів аналізу тексту для класифікації настроїв у відгуках про фільми.

4) У завданні COVID-19 Open Research Dataset Challenge (CORD-19) необхідно розробити інструменти для системного аналізу наукових даних, що стосуються COVID-19, з метою допомоги у боротьбі з пандемією. Задача полягає у створенні ефективних алгоритмів та моделей, які зможуть автоматично обробляти великі обсяги наукових даних, включаючи статті, дослідження, клінічні дані тощо, для виявлення корисної інформації щодо властивостей вірусу, методів лікування, профілактики та поширення захворювання. Оцінка роботи полягає у розробці інноваційних технік аналізу даних, які дозволять вченим та медичним фахівцям швидше зрозуміти та відреагувати на виклики, пов'язані з пандемією COVID-19.

5) У GoDaddy Data Challenge учасники мають прогнозувати місячну активність мікропідприємств в певній області на основі даних на рівні округів в США. Задача полягає у розробці точної моделі, навченої на внутрішніх та переписних даних, з використанням нових підходів машинного навчання для поліпшення прогнозів та надання додаткової інформації для прийняття рішень. Ці прогнози можуть бути використані для створення політики та програм для поліпшення успішності та впливу найменших бізнесів, що відповідає стратегічним цілям уряду та бізнесу. Оцінка роботи здійснюється на основі SMAPE між прогнозованими та фактичними значеннями, а також на основі якості поданої моделі та її впливу на прийняття рішень політиками та бізнес-аналітиками.

6) Конкурс Data Science for Good: City of Los Angeles. Завдання полягає у покращенні якості вакансійних оголошень Міста Лос-Анджелеса через аналіз їх змісту та формату. Це необхідно зробити для залучення різноманітного та якісного пулу кандидатів, оскільки 1/3 працівників міста можуть вийти на пенсію до липня 2020 року. Головна мета - перетворити тексти вакансій у структурований CSV-файл, виявити мовні та стилістичні особливості, що можуть створювати упередження, та розробити рекомендації для поліпшення якості та різноманітності кандидатів. Оцінка роботи включатиме аналіз якості створеного CSV-файлу, документації та рекомендацій для міської адміністрації Лос-Анджелеса.

Навчальне електронне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Віталій Борисович Мокін
Михайло Володимирович Дратований

НАУКА ПРО ДАНІ: МАШИННЕ НАВЧАННЯ ТА ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

Навчальний посібник

Рукопис оформив М. Дратований

Видається в авторській редакції

Оригінал-макет виготовила О. Кушнір

Підписано до видання
Гарнітура Times New Roman.
Зам. No Р2022-071
Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
Email: irvc.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК No 3516 від 01.07.2009 р.