ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА СЕРВІСУ ДЛЯ ПРОГНОЗУВАННЯ НА ВЕЛИКИХ НАБОРАХ ДАНИХ МЕТОДАМИ МАШИННОГО НАВЧАННЯ МОВОЮ PYTHON**»

Виконав: студент 4 курсу, групи ПД– 44

спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Поночовний О.В.

(прізвище та ініціали)

Керівник Садовенко В.С.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Нормоконтроль Трінтіна Н.А.

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність - 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного

забезпечення

_____О.В. Негоденко

«____»_____2023 року

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Поночовний Олександр Вікторович

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка сервісу для прогнозування на великих наборах даних методами машинного навчання»
Керівник роботи к.ф.-м.н., доцент Садовенко В.С.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «24» лютого 2023року №26.

2. Строк подання студентом роботи «1» червня 2023 року

3. Вихідні дані до роботи:
3.1.Офіційна документація Python

3.2.Наукова-технічна література по розробці сервісів

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
4.1.Аналіз актуальності та огляд існуючих сервісів

4.2. Анаіз інструментів для реалізації сервіс

4.3.Розробка структури та програмна реалізація.

4.4.Висновки

5. Перелік графічного матеріалу
5.1. Титульний слайд

5.2. Мета, Об’єкт та предмет дослідження

5.3. Завдання дипломної роботи

5.4. Таблиця порівнянь аналогів

5.5. Список головних функціональних вимог

5.6. Програмні засоби реалізації

5.7. Діаграма варіантів використання

5.8. Діаграма класів

5.9. Екранні форми

5.10. Апробація результатів дослідження

5.11. Висновки

---

6. Дата видачі завдання «25» лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Утвердження теми бакалаврської роботи	25.02.2023	Виконано
2	Підбір науково-технічної літератури	26.02.2023 – 15.03.2023	Виконано
3	Аналіз актуальності та дослідження існуючихсервісів	15.03.2023 – 20.03.2023	Виконано
4	Аналіз та вибір інструментів для розробки сервісу	20.03.2023 – 14.04.2023	Виконано
5	Проектування та реалізація	14.04.2023 – 04.05.2023	Виконано
6	Вступ, реферат та висновки	04.05.2023 – 12.05.2023	Виконано
7	Попередній захист	15.05.2023 – 1.06.2023	
8	Захист роботи	15.06.2023	

Студент _____ Поночовний О.В.

(підпис) (прізвище та ініціали)

Керівник роботи_____ Садовенко В.С.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Бакалаврська робота на тему «Розробка сервісу для прогнозування на великих наборах даних методами машинного навчання мовою Python» складається зі вступу, трьох розділів, висновків, переліку джерел посилання, додатків.

Загальний обсяг звіту становить 71 сторінку. Список використаних джерел складається з 33 найменувань.

ВЕЛИКІ ДАНІ, ДЕРЕВА РІШЕНЬ, PYTHON, ШТУЧНИЙ ІНТЕЛЕКТ, РЕГРЕСІЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, МАШИННЕ НАВЧАННЯ.

Об’єкт дослідження – сервіс для прогнозування на великих наборах даних методами машинного навчання.

Предмет дослідження – методи та алгоритми машинного навчання, їх ефективність , точність та застосування для прогнозування значень на великих обсягах даних.

Мета дослідження – прогнозування майбутніх тенденцій, розвитку або подій на основі великих наборів даних.

Методи дослідження – методи системного аналізу, моделювання статистичних методів та алгоритмів машинного навчання для великого набору даних, аналіз наукової літератури, спостереження, абстрагування, узагальнення.

Визначено альтернативний метод використання засобів машинного навчання в процесі розробки сервісу для прогнозування на великих наборах даних.

На основі результатів виконаних досліджень розроблено сервіс для прогнозування на великих наборах даних та модель швидкої адаптації великих даних для машинної обробки та прогнозування.

Упровадження розробленої методики дозволяє прискорити процес навчання, а також виконання статичних функцій. Після навчання моделей, обиралась найрезультативніша модель за критеріями прогнозної якості.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних;

ЕОМ – електронно-обчислювальна машина;

ІІТ – інтелектуальні інформаційні технології;

ІС – інформаційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

МН – машинне навчання;

ПЗ – програмне забезпечення;

ШНМ – штучна нейронна мережа;

BIG DATA – великі дані;

PYTHON – мова програмування.

ВСТУП

Актуальність дослідження. За останні роки спостерігається значне збільшення обсягу доступних даних у різних галузях. Великі набори даних вимагають використання спеціальних методів та технологій для ефективного аналізу та прогнозування.

Прогнозування є важливим інструментом для прийняття рішень у багатьох сферах, включаючи бізнес, фінанси, медицину, науку та інші. Наявність ефективного сервісу прогнозування на великих наборах даних може значно полегшити процес прийняття рішень та покращити результати.

Варто відмітити, що у наші дні значна частина додатків використовує бази даних. Спектр застосування баз даних досить широкий: від одного користувача додатків, наприклад, «записників», до великих розподілених інформаційних систем, таких як пошукові служби, інтернет-магазини. Існує велика кількість різноманітних СУБД (система управління базами даних), призначених для різних задач, однак зазвичай не так просто зрозуміти, яка СУБД покаже себе краще в тих чи інших умовах. Завдання вибору відповідної бази даних виникає не тільки при створенні нової системи, але в разі виникнення проблем з уже існуючим рішенням.

Використання методів МН. Методи машинного навчання постійно розвиваються та вдосконалюються. Нові алгоритми та підходи дозволяють досягти більш точних та надійних прогнозів на великих наборах даних. Дослідження в цій області дозволить використовувати найновіші досягнення для створення ефективного сервісу прогнозування.

Створення сервісу прогнозування на великих наборах даних може мати практичне значення для багатьох організацій і компаній. Він може допомогти виявити тенденції, зробити прогнози щодо продажів, попиту на товари та послуги, планувати виробництво та поставки, а також покращити стратегію бізнесу.

Отже, враховуючи вищевикладене, актуальність дослідження полягає в необхідності розв'язання складних задач прогнозування на великих наборах даних та використання сучасних методів машинного навчання для досягнення точних та надійних результатів.

Мета дослідження – розробка та реалізація сервісу для прогнозування на великих наборах даних методами машинного навчання з використанням мови програмування Python.

Завдання дослідження:

1. розглянути теоретичні аспекти застосування методів обробки великих даних;
2. охарактеризувати сучасні стандарти та технології Big Data;
3. визначити особливості розробки сервісу для прогнозування на великих наборах даних в умовах невизначеності предметної галузі та характери змінних;
4. проаналізувати підходи до створення ефективної системи навчання для прогнозування неперервних чисельних ознак для великого набору даних.
5. розробити сервіс для прогнозування на великих наборах даних в умовах невизначеності предметної галузі та характери змінних методом Dask.

Об’єкт дослідження – сервіс для прогнозування на великих наборах даних методами машинного навчання.

Предмет дослідження – методи та алгоритми машинного навчання, їх ефективність , точність та застосування для прогнозування значень на великих обсягах даних.

У роботі застосовуються такі загальнонаукові методи дослідження як

- методи системного аналізу,
- статистичний метод,
- аналіз наукової літератури,

- спостереження,
- абстрагування,
- узагальнення,
- валідація та оцінка.

Результати дослідження пропонують альтернативний метод використання засобів системного програмування в процесі розробки сервісу для прогнозування на великих наборах даних методом Dask.

Практичне значення одержаних результатів. В результаті розроблено сервіс для прогнозування на великих наборах даних та модель швидкої адаптації великих даних для машинної обробки та прогнозування.

Упровадження розробленої методики дозволяє прискорити процес навчання, а також виконання статичних функцій. Після навчання моделей, обиралась найрезультативніша модель за критеріями прогнозної якості.

1 АНАЛІТИЧНИЙ ОГЛЯД ІНФОРМАЦІЙНИХ ДЖЕРЕЛ З ТЕХНОЛОГІЙ ОБРОБКИ ВЕЛИКИХ ДАНИХ

1.1. Загальна характеристика технологій обробки великих даних

Основна ідея сучасних інформаційних технологій базується на концепції баз даних. Згідно з цією концепцією, основою інформаційних технологій є дані, які повинні бути організовані в бази даних, щоб адекватно відображати мінливий реальний світ.

Визначення терміну база даних різняться, але завжди передбачається, що вони відповідають таким характеристикам:

- база даних містить деякий безліч даних необхідних (і, бажано, достатніх) для вирішення конкретних завдань багатьох користувачів (в тому числі як реальних, так і потенційних) або, що в принципі майже одне і те ж, - задоволення відповідних інформаційних потреб;
- дані або інформаційні елементи в базі даних певним чином структуровані і пов’язані між собою (т. е. організовані), при цьому структура, склад даних і їх зміст в базі даних не залежать від особливостей прикладних програм, що використовуються для управління базою даних;
- дані (інформаційні елементи) представлені на машинозчитуваних носіях у формі придатній для оперативного використання їх із застосуванням засобів обчислювальної техніки, включаючи і систем управління базами даних (СУБД).

Основні завдання, що виникають при роботі з базою даних:

- зберігання таблиць і записів;
- управління записами і таблицями.

Отже, базу даних (або систему баз даних) можна уявити як набір програмно-апаратних засобів, що вирішують завдання централізованого зберігання і обробки даних користувача.

Інформаційна система (ІС) - це система, що включає в себе інформаційні, програмні, технічні, мовні, організаційні та функції, призначені для збору, накопичення, передачі, пошуку, обробки даних.

За значущістю функцій інформаційні системи класифікуються на інформаційно пошукові системи і системи обробки даних.

Основні функції інформаційно пошукових систем - це пошук даних і висновок необхідних даних.

Основні функції систем обробки даних - це оновлення даних і висновок необхідних даних.

Між власне фізичної базою даних і користувачами розташовується рівень програмного забезпечення, тобто система керування базою даних (СУБД).

Функції СУБД:

- безпосереднє управління даними у зовнішній пам’яті;
- управління розділами оперативної пам’яті з метою збільшення швидкості роботи бази даних;
- управління транзакціями. Транзакція - це послідовність операцій над базою даних, що розглядаються СУБД як єдине ціле. Поняття транзакції необхідне для підтримки логічної цілісності бази даних, механізм транзакцій забезпечує захист бази даних від апаратних збоїв, можливість багатокористувацького доступу до даних у віддалених базах;
- журналізація - ведення журналу змін інформації в цілях підтримки надійності зберігання даних, а також підтримка мов бази даних, мовні засоби сучасних СУБД.

На рисунку 1.1 показані три види архітектури з’єднання клієнта з даними. Перший вид являє однозвенную архітектуру, яка крім даних містить єдину ланку – клієнт.

Дволанкова архітектура містить ще й сервер бази даних, що забезпечує значну частину логіки управління даними, в той час як клієнт в основному зайнятий відображенням даних в зручному для користувача вигляді.

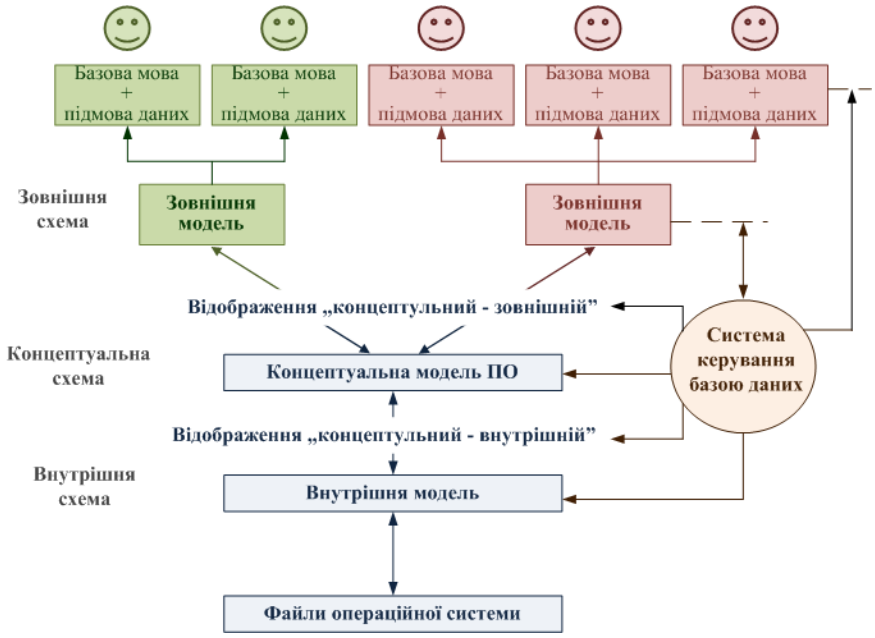


Рисунок 1.1 - Архітектура баз даних

У триланкових СУБД додано ще одну проміжну ланку - сервер додатків. Сервер додатківпризначений для забезпечення зв’язку клієнта з сервером бази даних і повністю позбавляє клієнта від будь-яких проблем щодо управління даними.

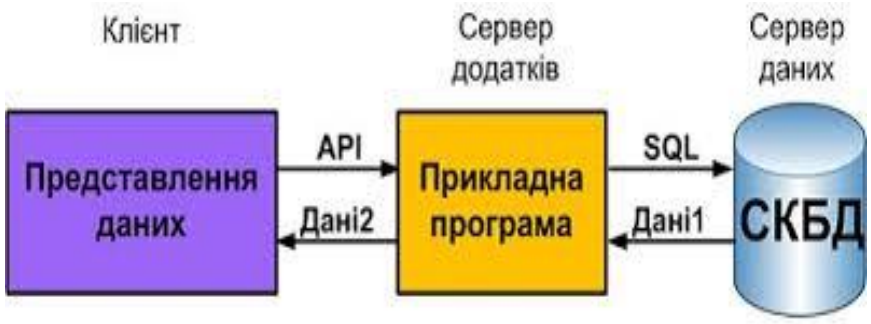


Рисунок 1.2 - Архітектури з’єднання клієнта з даними - одноланкова, двуланкова

Великі дані в першу чергу відносяться до наборів даних, які надто великі або складні для обробки за допомогою традиційного програмного забезпечення для обробки даних. Дані з великою кількістю записів (рядків) мають більшу статистичну силу, а дані з більш високою складністю (більше атрибутів або стовпців) можуть призвести до більш високого рівня помилкових виявлень.

Хоча інтерпретація, яка іноді використовується вільно, частково через відсутність формального визначення, здається, що найкраще описує великі дані, яка пов’язана з великим обсягом інформації, яку ми не можемо зрозуміти, якщо використовувати тільки в менших кількостях.

Проблеми аналізу великих даних включають збір даних, зберігання даних, аналіз даних, пошук, спільне використання, передачу, візуалізацію, запити, оновлення, конфіденційність інформації та джерело даних. Спочатку великі дані асоціювалися з трьома ключовими поняттями: обсяг, різноманітність та швидкість.
При аналізі великих даних виникають проблеми з вибіркою, і таким чином раніше допускалися лише спостереження та вибірка. Таким чином, четверте поняття, істинність відноситься до якості або проникливості даних. Без достатніх інвестицій у досвід для забезпечення достовірності великих даних обсяг та різноманітність даних можуть призвести до витрат та ризиків, які перевищують можливості організації зі створення та вилучення цінності з великих даних.

Великі дані - це різноманітні дані, що надходять із високою швидкістю, обсяг яких постійно зростає. Таким чином, три основні властивості великих даних - це різноманітність, висока швидкість надходження та великий обсяг.

Основні властивості великих даних:

- Обсяг.

Кількість даних – важливий фактор. Маючи в своєму розпорядженні велику кількість, Вам потрібно буде обробляти великі обсяги неструктурованих даних низької щільності. Цінність таких даних не завжди відома. Це можуть бути дані каналів Twitter, дані відвідуваності веб-сторінок, дані мобільних додатків, мережевий трафік, дані датчиків. Деякі організації можуть надходити десятки терабайт даних, до інших - сотні петабайт.

- Швидкість.

Швидкість у цьому контексті - це швидкість прийому даних і, можливо, дій з їхньої основі. Зазвичай високошвидкісні потоки даних надходять у оперативну пам’ять, а чи не записуються на диск. Деякі "розумні" продукти, що функціонують на основі Інтернету, працюють у режимі реального чи практично реального часу. Відповідно, такі дані вимагають оцінки та дій у реальному часі.

- Різноманітність.

Різноманітність означає, що доступні дані належать до різних типів. Традиційні типи даних структуровані і можуть бути одразу збережені в реляційній базі даних. З появою Big Data дані почали надходити у неструктурованому вигляді. Такі неструктуровані та напівструктуровані типи даних як текст, аудіо та відео вимагають додаткової обробки для визначення їх значення та підтримки метаданих.

Ще дві якості сформувалися останні кілька років: цінність і достовірність. Дані мають внутрішньо властиву їм цінність. Однак, щоб вони приносили користь, цю цінність необхідно розкрити.

Найновіші досягнення у сфері технологій дозволили значно знизити вартість сховищ та обчислень, що дає можливість зберігати та обробляти обсяги даних, що постійно зростають. Сучасні технології дозволяють зберігати та обробляти більше даних за меншу вартість, що дозволяє Вам приймати більш точні та виважені бізнес-рішення.

Вилучення цінності з великих даних не зводиться тільки до їх аналізу (це їхня окрема перевага). Йдеться про комплексний дослідний процес за участю фахівців з глибокого аналізу, корпоративних користувачів та керівників, які будуть ставити правильні питання, виявляти шаблони, робити обґрунтовані припущення та передбачати поведінку.

1.2. Сучасні стандарти та технології Big Data

Поява платформ на основі відкритого коду, таких як Hadoop та пізніше Spark, відіграла значну роль у поширенні великих даних, оскільки ці інструменти спрощують обробку великих даних та знижують вартість зберігання. За минулі роки обсяги високих даних зросли на порядки. Величезні обсяги даних з’являються в результаті діяльності користувачів, але тепер не лише їх.

З появою Інтернету речей (IoT) все більше пристроїв отримує підключення до Інтернету, що дозволяє збирати дані про моделі дій користувачів і роботу продуктів. А коли з’явилися технології машинного навчання, обсяг даних зріс ще більше.

Великі дані мають довгу історію розвитку, проте їхній потенціал ще далеко не розкритий. Хмарні обчислення розсунули межі застосування великих даних набагато ширше. Хмарні технології забезпечують по-справжньому гнучкі можливості масштабування, що дозволяє розробникам розгортати кластери для тестування вибіркових даних на вимогу. Крім того, також все більш значущими стають графові

бази даних, що дозволяють відображати величезні обсяги даних так, щоб їх аналізувати можна було швидко і всеосяжно (рис. 1.3).

Перевагами великих даних є такі:

1. Великі дані дають можливість отримувати повніші відповіді, тому вони надають більше інформації. Більш докладні відповіді означають, що можна бути більш впевненими у достовірності даних, що забезпечує абсолютно новий підхід до вирішення завдань.

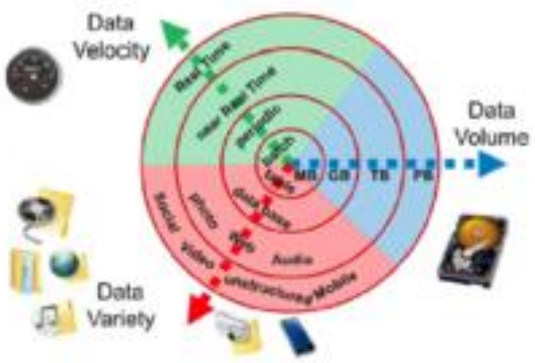


Рисунок 1.3 - Популярність основних характеристик великих даних за обсягом, швидкістю та різноманітністю

2. Готове рішення вже інтегрує всі ці інструменти один з одним і постачається з повним пакетом програмної документації, що полегшує процес керування та підтримки вашої інфраструктури великих даних. Однак слід підкреслити, що ці практичні рамки вимагають відповідних зарплат. Дані надходять у всіх видах форматів, від структурованих записів, числових даних у традиційних базах даних до неструктурованих текстових документів, електронних листів, відео, аудіо, даних фондового ринку та фінансових операцій. До цього часу електронні таблиці та бази даних були єдиними джерелами даних для більшості програм. Програми для аналізу тепер також розглядають дані у

формі електронних листів, фотографій, відео, пристроїв спостереження, файлів PDF та аудіо. Ця різноманітність неструктурованих даних створює певні проблеми при зберіганні, вилученні та аналізі даних. Різноманітність даних зображено на рис.1.4.

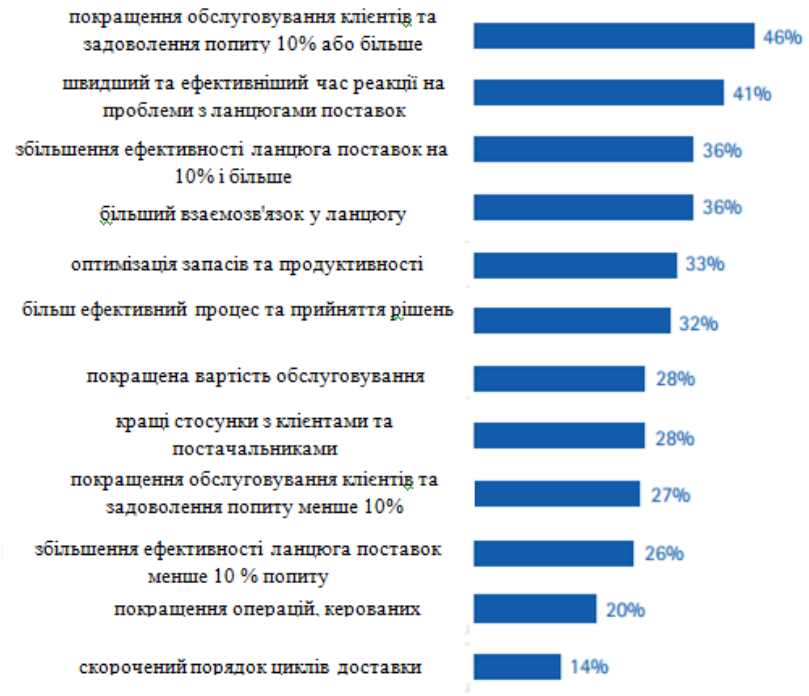


Рисунок 1.4 – Існуючі технології даних в області Big Data

Разом з тим, існуть складнощі при використанні великих даних, а саме:

1. Насамперед великі дані передбачувано займають багато місця. Хоча нові технології зберігання постійно розвиваються, обсяг даних зростає вдвічі майже кожні два роки. Організації досі стикаються з проблемами зростання обсягів даних та їх ефективного зберігання. Але недостатньо просто знайти велике сховище. Дані необхідно використовувати, щоб вони приносили зиск, і розмір цієї вигоди залежить від обробки даних.

2. Чисті дані, тобто дані, актуальні для клієнта та організовані для ефективного аналізу, потребують ретельної обробки. Фахівці з вивчення даних витрачають від 50 до 80% робочого дня на обробку та підготовку даних для використання.

3. І, нарешті, технології великих даних розвиваються семимильними кроками. Декілька років тому Apache Hadoop була найпопулярнішою технологією для роботи з великими даними. Платформа Apache Spark з'явилася у 2014 році. Сьогодні оптимальним підходом є спільне використання цих двох платформ. Щоб встигати за розвитком великих даних, потрібно докладати великих зусиль.

Основні компоненти великих даних описуються наступним чином:

- Методи аналізу даних, такі як A/B-тестування, машинне навчання та обробка природної мови.
- Технології великих даних, такі як бізнес-аналітика, хмарні обчислення та бази даних.
- Візуалізація, така як діаграми, графіки та інші види відображення даних.

Практики процесів аналізу великих даних, як правило, вороже ставляться до повільніших загальних сховищ, віддаючи перевагу сховищу з прямим підключенням (DAS) в його різних формах від твердотільного накопичувача (SSD) до диска SATA великої ємності, захованого всередині вузлів паралельної обробки. Сприйняття загальних архітектур зберігання – мережі зберігання даних (SAN) та мережевого сховища (NAS) – таке, що вони відносно повільні, складні та дорогі. Ці якості несумісні з системами аналітики великих даних, які процвітають за рахунок продуктивності системи, стандартної інфраструктури та низької вартості.

- Доставка інформації в реальному або близькому до реального часу є однією з визначальних характеристик великих аналітики даних. Таким чином, затримки уникають завжди і скрізь, де це можливо. Дані в пам'яті або на диску з прямим підключенням в порядку, а дані в пам'яті або на диску на іншому кінці з'єднання FC SAN немає. Вартість SAN у масштабі, необхідному для аналітичних додатків

набагато вище, ніж в інших методів зберігання. Великі дані дозволяють отримувати нові цінні відомості, які відкривають нові можливості та бізнес-моделі.

Щоб розпочати роботу з великими даними, необхідно виконати три дії, а саме:

1. Інтеграції.

Технологія великих даних дозволяє об’єднувати дані із розрізнених джерел та додатків. Традиційні механізми інтеграції, такі як засоби для отримання, перетворення та завантаження даних (ETL), не справляються з подібними завданнями. Для аналізу наборів даних розміром в терабайт, а то й петабайт, потрібні нові стратегії та технології. Під час етапу інтеграції відбувається додавання, обробка та форматування даних, щоб корпоративним аналітикам було зручно з ними працювати.

2. Управління.

Великим даним потрібне об’ємне сховище. Рішення для зберігання може бути розміщене в локальному або хмарному середовищі або там і там. Можна зберігати дані у бажаному форматі та застосовувати бажані вимоги до обробки (і необхідні механізми обробки) до наборів даних у міру необхідності. Більшість організацій обирають рішення для зберігання даних, залежно від того, де вони зберігаються в даний час. Хмарні сховища користуються зростаючою популярністю, оскільки підтримують актуальні вимоги до обчислень і дозволяють використовувати ресурси в міру потреби.

3. Аналіз.

Вкладення у великі дані окупляться сповна, коли Ви приступите до аналізу даних і почнете робити дії, виходячи з отриманих відомостей. Забезпечте новий рівень прозорості завдяки візуальному аналізу різноманітних наборів даних. Використовуйте глибокий аналіз даних, щоб робити нові відкриття. Ділиться своїми відкриттями з іншими. Створюйте моделі даних за допомогою машинного навчання та штучного інтелекту. Застосуйте свої дані насправді.

Дані самі по собі не мають цінності, доки вони не перетворені на корисну інформацію та знання, які допомагають керівництву приймати рішення. Для цієї мети на ринку є різноманітне програмне забезпечення для великих даних. Це програмне забезпечення допомагає зберігати, аналізувати, звітувати тощо.

1.3. Постановка задачі розробки моделі прогнозування Big Data за обмежених технічних ресурсів

Метою є розробка та практична реалізація алгоритму машинного навчання для прогнозування неперервних чисельних ознак для великого набору даних.

Ставимо перед собою такі завдання:

1. розглянути теоретичні аспекти застосування методів обробки великих даних;

2. охарактеризувати сучасні стандарти та технології Big Data;

4. визначити особливості розробки сервісу для прогнозування на великих наборах даних в умовах невизначеності предметної галузі та характери змінних;

5. проаналізувати підходи до створення ефективної системи навчання для прогнозування неперервних чисельних ознак для великого набору даних.

6. розробити сервіс для прогнозування на великих наборах даних в умовах невизначеності предметної галузі та характери змінних методом Dask.

1.4 Висновки за розділом

Підсумовуючи перший розділ, можемо зробити такі висновки:

1. Визначено, що великі дані (Big Data) - це різноманітні великі дані, які можуть генеруватися, збиратися та поступати в реальному часі або з високою

швидкістю. Таким чином, три основні властивості великих даних - це різноманітність, висока швидкість надходження та великий обсяг.

2. Охарактеризовано стандарти та технології Big Data, їх компоненти, переваги та недоліки.

3. Сформовано задачі розробки моделі прогнозування Big Data за обмежених технічних ресурсів.

2 МЕТОДИ ТА ЗАСОБИ РОЗРОБКИ СЕРВІСУ ДЛЯ ПРОГНОЗУВАННЯ НА ВЕЛИКИХ НАБОРАХ ДАНИХ В УМОВАХ НЕВИЗНАЧЕНОСТІ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ХАРАКТЕРУ ЗМІННИХ

2.1. Огляд сучасних моделей машинного навчання, побудованих на Big Data

Оскільки даний дипломний проект буде розроблений мовою програмування Python, то наведені нижче приклади можуть бути реалізовані цією мовою. Існуючі інструменти для роботи з великими даними :

- Apache Spark є відкритою системою обробки та аналізу великих обсягів даних.

Він надає широкі можливості для розподіленого обчислення, обробки стрімів даних, машинного навчання та аналізу даних в реальному часі. Apache Spark розроблений для високошвидкісної обробки даних. Він використовує оптимізований двигун в пам'яті, що дозволяє виконувати операції на даних в оперативній пам'яті, що значно прискорює обробку.Spark дозволяє розподіляти обчислення на кластері з великою кількістю вузлів. Він автоматично керує розподіленими задачами та даними, забезпечуючи високу масштабованість та надійність.

Apache Spark має багато модулів та бібліотек, що дозволяють виконувати різноманітні завдання. Він підтримує мови програмування, такі як Scala, Java, Python та R, що дозволяє розробникам використовувати їх у власному стеку технологій.

Spark Streaming дозволяє обробляти та аналізувати стріми даних в реальному часі. Це дозволяє виконувати аналітику та обробку в реальному часі на великих обсягах даних.

Машинне навчання та глибинне навчання: Spark має багато бібліотек для машинного навчання та глибинного навчання, такі як MLlib і TensorFlow на Spark. Це дозволяє використовувати Spark для побудови та тренування моделей машинного навчання на великих обсягах даних.Може працювати як серед кластера Apache Hadoop під управлінням YARN, так і без компонентів ядра Apache Hadoop, підтримує кілька розподілених систем зберігання - HDFS, OpenStack Swift, NoSQL-СУБД Apache Cassandra, Amazon S3 (рис. 2.2).

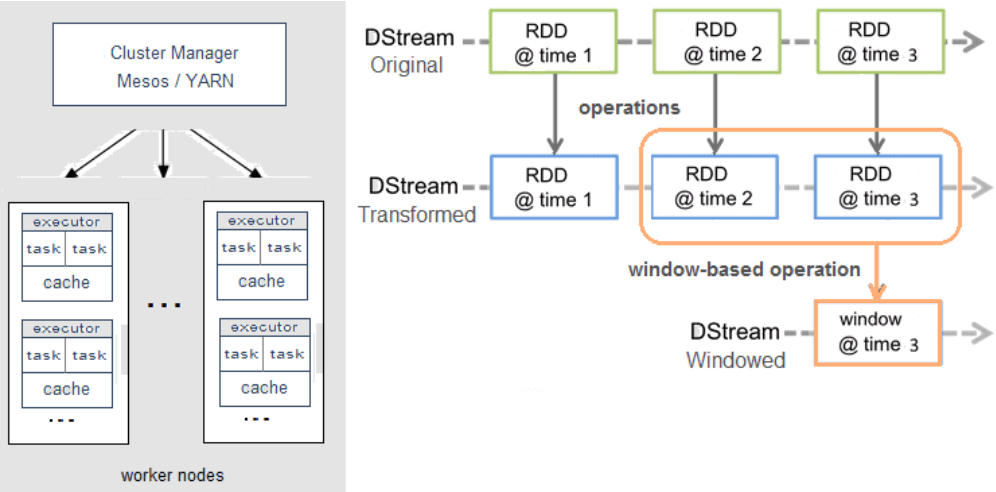


Рисунок 2.2 - Принцип роботи Apache Spark

- Elasticsearch.

Elasticsearch - це розподілена система пошуку та аналізу текстових даних з відкритим вихідним кодом. Вона базується на Apache Lucene, потужній бібліотеці для повнотекстового пошуку. Elasticsearch надає можливості швидкого, масштабованого та розподіленого пошуку, аналізу та зберігання даних. ES є ядром ELK-стеку (Elastic Stack), до складу якого, крім Elasticsearch, входять такі продукти:

Logstash - інструмент збору, перетворення та збереження в загальному сховищі подій з різних джерел (файли, бази даних, логи та ін) в реальному часі;

Kibana є візуалізаційним інтерфейсом для Elasticsearch, який дозволяє швидко та легко відображати, аналізувати та інтерактивно взаємодіяти з даними, збереженими у Elasticsearch;

FileBeat - легкий агент на серверах, який призначений для збору, відправки та різних типів оперативних даних в ES.

У масштабних Big Data системах кілька копій Elasticsearch об’єднуються в кластер. Пошукові індекси можна розділити на сегменти, реплікувавши кожен із яких кілька разів. Це забезпечує відмовостійкість системи. На вузлі ES кластера може розміщуватися кілька сегментів. Кожен вузол кластера діє як координатор для делегування операцій правильному сегменту з автоматичним перебалансуванням та маршрутизацією.

Пов’язані дані часто зберігаються в тому самому індексі з одного або декількох первинних сегментів і декількох реплік. Після створення індексу кількість первинних сегментів не можна змінити. Довгострокове зберігання індексу забезпечує шлюз, дозволяючи відновлювати індекс при збої сервера (рис. 2.3)

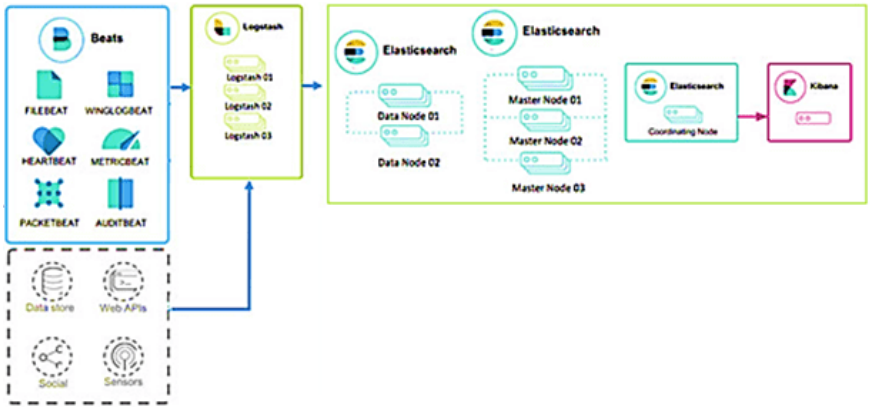


Рисунок 2.3 - Принцип роботи ELK-стеку

Завдяки широкому набору функціональних можливостей, особливо повнотекстового пошуку за багатьма мовами та аналітикою в реальному часі, Elasticsearch активно застосовується в різних Big Data системах великих та середніх компаній по всьому світу. З найбільш відомих зарубіжних користувачів варто відзначити корпорації Netflix, IBM, Facebook, Amazon, GitHub, Wikimedia, CERN, Mozilla, Adobe.

- Hive.

Hive – це система управління базами даних (СУБД) у рамках платформи Hadoop для зберігання та обробки великих даних у розподіленому середовищі. Хайв дозволяє проектувати структури Big Data (таблиці, партиції, бакети) за допомогою SQL-подібної мови, що називається HiveQL.

Apache Hive – це SQL інтерфейс доступу до даних для платформи Apache Hadoop. Hive дозволяє виконувати запити, агрегувати та аналізувати дані використовуючи SQL синтаксис. Для даних у файловій системі HDFS використовується схема доступу читання, що дозволяє поводитися з даними, як із звичайної таблицею чи реляційної СУБД. Запити HiveQL транслюються до Java-коду завдань MapReduce (рис. 1.8).

Запити Hive створюються мовою запитів HiveQL, яка базується на мові SQL, але не має повної підтримки стандарту SQL-92. Однак, ця мова дозволяє програмістам використовувати власні запити, коли незручно або неефективно використовувати можливості HiveQL. HiveQL може бути розширений за допомогою скалярних функцій користувача (UDF), агрегацій (UDAF кодів), і табличних функцій (UDTF).

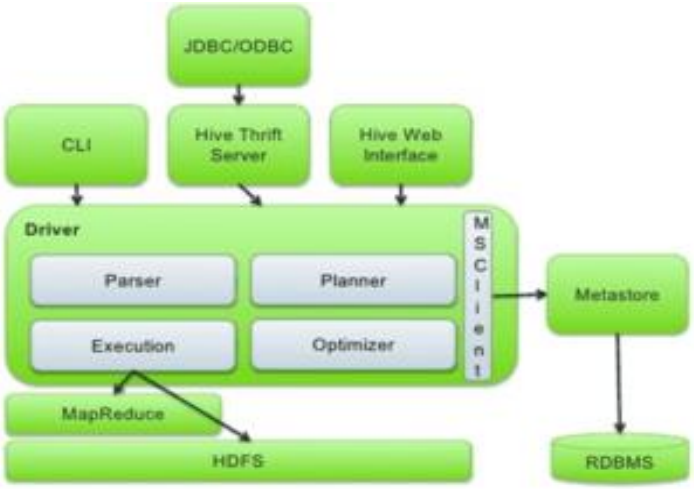


Рисунок 2.4 – Архітектура Hive

Apache Hive – це СУБД для зберігання та обробки великих даних у розподіленому кластері екосистеми Hadoop.

СУБД Apache Hive включає такі елементи:

- HCatalog – це компонент Хайв, який відповідає за керування таблицями та сховищами. Цей компонент також забезпечує користувачів різними інструментами обробки Big Data (включаючи MapReduce та Pig) для більш простого читання та запису даних.
- WebHCat – це компонент, що дозволяє виконувати базові операції (читання, запис та видалення) з метаданими в мережі Хайв за допомогою використання інтерфейсу HTTP (Hyper Text Transfer Protocol). WebHCat також використовується як сервер для розгортання середовища Хайва.
- Dask.

Dask - це бібліотека з відкритим кодом, призначена для забезпечення паралелізму з існуючим стеком Python. Він забезпечує інтеграцію з бібліотеками Python, такими як NumPy Arrays, Pandas DataFrames та scikit-learn, щоб забезпечити

паралельне виконання на кількох ядрах, процесорах та комп'ютерах без необхідності вивчення нових бібліотек чи мов.

Dask складається з двох частин:

1. API колекцій для паралельних списків, масивів та кадрів даних для природного масштабування Numpy, NumPy, Pandas та scikit-learn для роботи в середовищах з великим обсягом пам'яті або розподілених середовищах. Колекції Dask є паралельними колекціями з базової бібліотеки (наприклад, масив Dask складається з масивів Numpy) і запускаються поверх планувальника завдань.

Планувальник завдань для побудови графіків задач та координації, планування та моніторингу завдань, оптимізований для інтерактивних робочих навантажень між ядрами ЦП та машинами (рис.2.5).

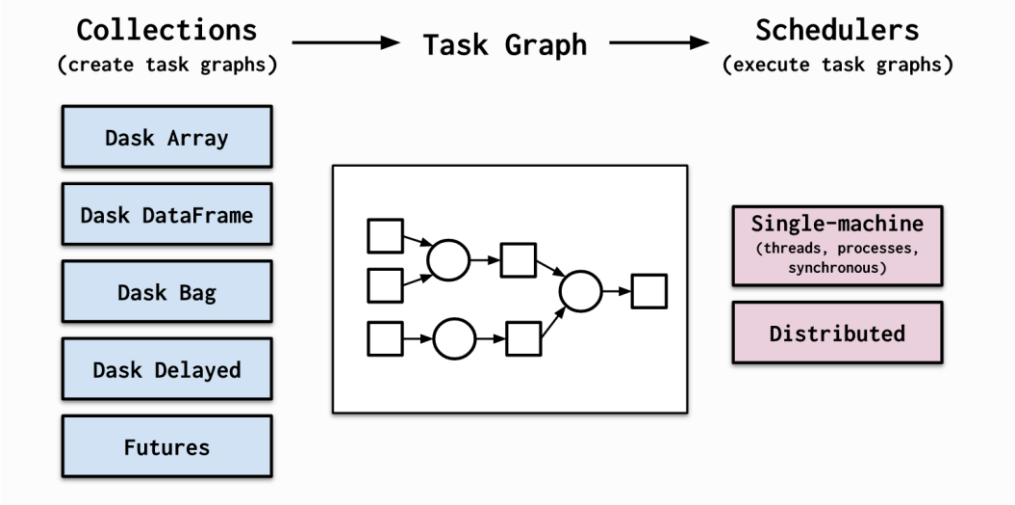


Рисунок 2.5 – Архітектура Dask

Кожна з трьох паралельних колекцій Dask - DataFrames, Bags і Arrays - може автоматично використовувати дані, розділені між ОЗП і диском, розподілені кількома вузлами в кластері, залежно від доступності ресурсів. Для завдань, які можна розпаралелити, але які погано вписуються в абстракції високого рівня, такі як

Dask Arrays або DataFrames, існує відкладена функція, яка використовує декоратори Python для зміни функцій, щоб вони працювали ліниво. Це означає, що виконання затримується, а функція та її аргументи поміщаються у граф задач.

Планувальник завдань Dask може масштабуватися до кластерів із тисячі вузлів, а його алгоритми були протестовані на деяких із найбільших у світі суперкомп'ютерів. Його інтерфейс планування завдань можна налаштувати для конкретних завдань. Dask забезпечує мінімальні накладні витрати, малу затримку та мінімальну серіалізацію, необхідну для швидкості.

Зручна для користувача високорівнева мова програмування Python та бібліотеки Python, такі як NumPy, Pandas та scikit-learn, набули широкого поширення серед фахівців за даними.

Розроблені доти, як випадки використання великих даних стали настільки поширеними, ці бібліотеки мали сильного рішення для паралелізму. Python був кращим вибором для одноядерних обчислень, але користувачі змушені були шукати інші рішення для багатоядерного або багатомашинного паралелізму. Це викликало розчарування та перерву у роботі користувачів.

Ця зростаюча потреба у масштабуванні робочих навантажень у Python призвела до природного зростання Dask за останні п'ять років.

Dask - це спосіб, що легко встановлюється і швидко налаштовується, прискорити аналіз даних у Python, який не вимагає від розробників оновлення апаратної інфраструктури або переходу на іншу мову програмування. Синтаксис, що використовується для запуску завдань Dask, такий самий, як і для інших операцій Python, тому його можна інтегрувати з невеликою доробкою коду.

Також популярний серед веб-розробників Python має надійний мережевий стек, який Dask використовує для створення гнучкої, продуктивної розподіленої обчислювальної системи, здатної масштабувати широкий спектр робочих навантажень. Гнучкість Dask допомагає йому виділитися серед інших рішень для

роботи з великими даними, як Hadoop або Apache Spark, а підтримка нативного коду робить його особливо зручним для користувачів Python і розробників C/C++/CUDA. Dask був швидко прийнятий спільнотою розробників Python і виріс разом із популярністю Numpy та Pandas, які надають цінні розширення Python для вирішення спеціальної аналітики та математичних обчислень.

Він також набагато краще масштабується, ніж Pandas, і особливо добре працює із завданнями, які легко розпаралелити, наприклад, із сортуванням даних у тисячах електронних таблиць. Прискорювач може завантажувати в пам’ять сотні кадрів даних Pandas та координувати їх за допомогою єдиної абстракції.

Сьогодні Dask управляється спільнотою розробників, яка охоплює десятки установ та проектів PyData, таких як Pandas, Jupyter та Scikit-Learn. Інтеграція Dask із цими популярними інструментами привела до швидкого поширення: близько 20% серед розробників, яким потрібні інструменти Pythonic для роботи з великими

Використовуючи Pandas DataFrames, Dask може включати програми для аналізу часових рядів, бізнес-аналітики та підготовки даних. Dask-ML, бібліотеку для розподіленого та паралельного машинного навчання, можна використовувати з Scikit-Learn та XGBoost для створення масштабованого навчання та прогнозування на великих моделях та наборах даних. Розробники можуть використовувати стандартні робочі процеси Dask для підготовки та налаштування даних, а потім передавати дані в XGBoost або Tensorflow.

2.2. Підходи до створення ефективної системи навчання для прогнозування неперервних чисельних ознак для великого набору даних

У даній роботі розглянуті класичні статистичні моделі, алгоритми машинного навчання з учителем, ансамблеві моделі та їх застосування в умовах обмеженості технічних та інформаційних ресурсів.

• Лінійна регресія.

Лінійна регресія є одним з найпростіших та найпоширеніших методів машинного навчання для вирішення задачі прогнозування. Вона використовується для встановлення лінійного зв'язку між вхідними ознаками (змінними) і вихідним значенням (залежною змінною) (рис. 2.6).

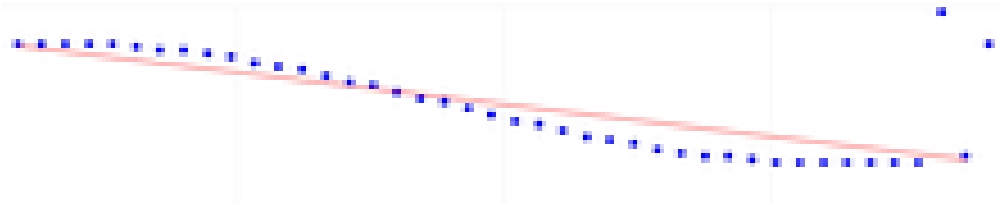


Рис. 2.6 - Приклад лінії (червона), побудованої з використанням лінійної регресії

Лінійна регресія була першим типом регресійного аналізу, який був ретельно вивчений та широко використовувався у практичних додатках. Це з тим, що моделі, які лінійно залежить від своїх невідомих параметрів, легше підібрати, ніж моделі, які нелінійно пов’язані зі своїми параметрами, і навіть оскільки статистичні властивості отриманих оцінок легше визначити.

Лінійна регресія має багато практичних застосувань. Більшість додатків потрапляють до однієї з наступних двох широких категорій:

Якщо метою є зменшення помилок у прогнозуванні або прогнозуванні , можна використовувати лінійну регресію, щоб підігнати прогностичну модель до набору даних значень відгуку і незалежних змінних. Якщо після розробки такої моделі збираються додаткові значення незалежних змінних без супутнього значення відгуку, підібрану модель можна використовувати для прогнозування відгуку.

Якщо мета полягає в тому, щоб пояснити зміну змінної відгуку, яка може бути пов’язана зі зміною змінних, що пояснюють, можна застосувати лінійний регресійний аналіз для кількісної оцінки сили взаємозв’язку між відгуком і

пояснювальними змінними і, зокрема, для визначення того, чи є деякі незалежні змінні можуть взагалі не мати лінійного зв’язку з відповіддю або визначати, які підмножини незалежних змінних можуть містити надмірну інформацію про відповідь.

У лінійній регресії передбачається, що спостереження (червоний) є результатом випадкових відхилень (зелений) від основного зв’язку (синій) між залежною змінною *(y)* та незалежною змінною *(x)*.

- Дерево рішень.

Дерево рішень - це ієрархічна модель підтримки прийняття рішень, в якій використовується деревоподібна модель рішень та їх можливих наслідків, включаючи результати випадкових подій, витрати ресурсів та корисність. Це один із способів відображення алгоритму , який містить лише оператори умовного керування.

Дерева рішень зазвичай використовуються в дослідженнях операцій, особливо в аналізі рішень, щоб допомогти визначити стратегію, яка з найбільшою ймовірністю призведе до мети, але вони також є популярним інструментом у машинному навчанні (рис.2.7).

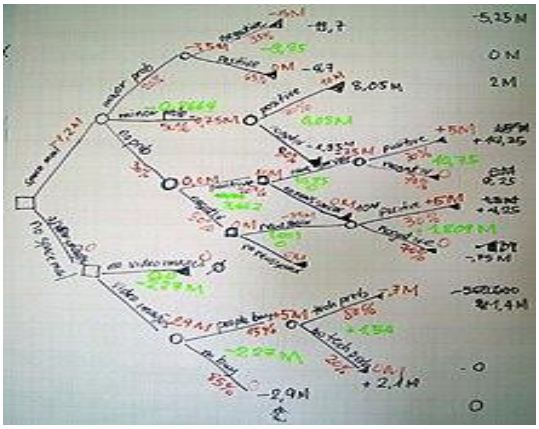


Рисунок 2.7 – Приклад ручного дерева рішень

Дерево рішень є структурою, подібною до блок-схеми , в якій кожен внутрішній вузол є «перевіркою» атрибута (наприклад, чи випадає при підкиданні монети орел або решка), кожна гілка є результатом перевірки, а кожен кінцевий вузол є результатом перевірки. . мітка класу (рішення приймається після обчислення всіх атрибутів). Шляхи від кореня до листа є правилами класифікації.

В аналізі рішень дерево рішень і тісно пов’язана з ним діаграма впливу використовуються як візуальний та аналітичний інструмент підтримки прийняття рішень, де розраховуються очікувані значення (або очікувана корисність) конкуруючих альтернатив.

Дерево рішень складається з трьох типів вузлів:

– Вузли рішень – зазвичай представлені квадратами.
– Вузли шансів – зазвичай представлені гуртками.
– Кінцеві вузли – зазвичай представлені трикутниками.

Дерева рішень зазвичай використовуються в дослідженнях операцій та управлінні операціями. Якщо на практиці рішення повинні прийматися в режимі онлайн без відкликання при неповних знаннях, дерево рішень має бути паралельно з моделлю ймовірності як модель найкращого вибору або алгоритм моделі онлайн-вибору.

Інше використання дерев рішень як описовий засіб для розрахунку умовних ймовірностей.

Дерева рішень, діаграми впливу, функції корисності та інші інструменти та методи аналізу рішень викладаються студентам бакалаврату в школах бізнесу, економіки охорони здоров’я та суспільної охорони здоров’я та є прикладами методів дослідження операцій чи науки управління (рис.2.8).

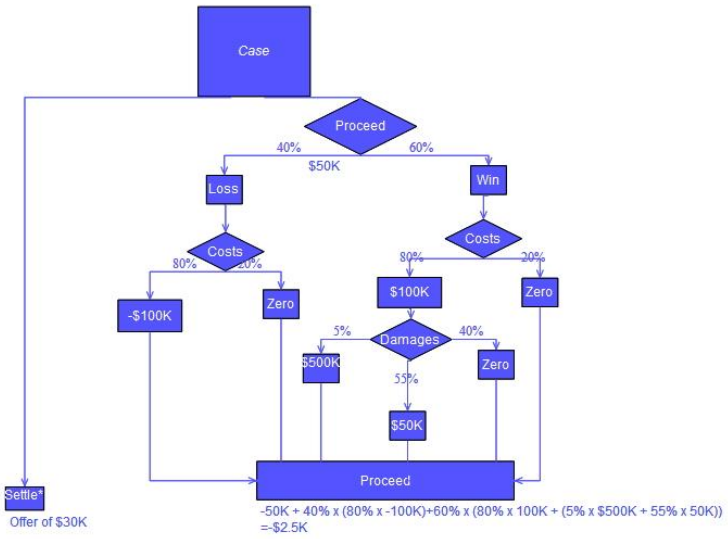


Рисунок 2.8 - Приклад дерева рішень з використанням символів блок-схеми

Серед інструментів підтримки прийняття рішень дерева рішень (і діаграми впливу) мають низку переваг.

– Вони прості для розуміння та інтерпретації. Люди можуть зрозуміти моделі дерев рішень після короткого пояснення.

– Важливі ідеї можуть бути отримані на основі опису експертами ситуації (її альтернатив, ймовірностей та витрат) та їх переваг щодо результатів.

– Може поєднуватись з іншими методами прийняття рішень.

– Можна враховувати дії більш ніж однієї особи, яка приймає рішення.

Недоліки дерев рішень:

– Вони нестабільні, а це означає, що невелика зміна даних може призвести до великої зміни структури оптимального рішення дерева.

– Часто вони щодо неточні. Багато інших предикторів працюють краще з аналогічними даними. Це можна виправити, замінивши одне дерево рішень випадковим лісом дерев рішень, але випадковий ліс не так легко інтерпретувати як одне дерево рішень.

– Для даних, що включають категоріальні змінні з різною кількістю рівнів, приріст інформації на деревах рішень зміщується на користь атрибутів з великою кількістю рівнів.

– Розрахунки можуть стати дуже складними, особливо якщо багато значень є невизначеними та/або якщо багато результатів пов’язані між собою (рис. 2.9).

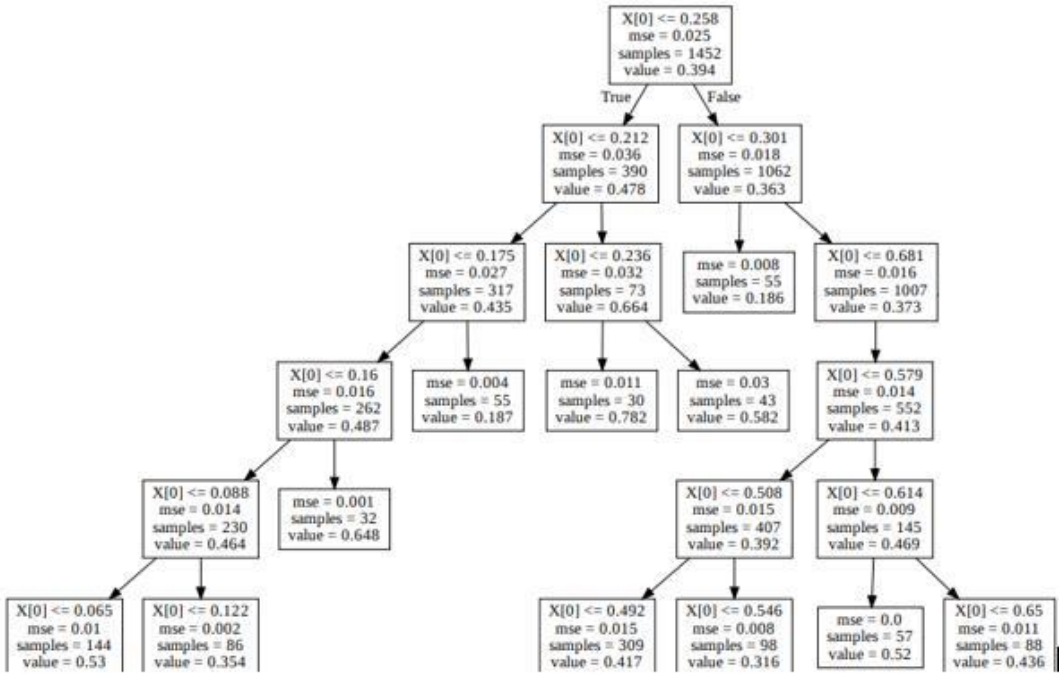


Рисунок 2.9 – Принцип роботи дерева рішень

Побудова дерева здійснюється в 4 етапи:

– вибір атрибут для здійснення розбиття в вузлі;

– визначення критерію зупинки навчання;

– вибір методу відсікання гілок;

– оцінка точності побудованого дерева.

Процес розбиття атрибутів у дереві рішень триває до побудови повного дерева або до досягнення заданої умови зупинки. Результатом є дерево, де кожна вершина

відповідає атрибуту, а кожне ребро представляє значення атрибуту, яке веде до наступної вершини або листової вершини, яка містить прогнозоване значення.Ансамблеві методи.

Ансамблі (ensembles) – це методи, які використовуються для поєднання прогнозів кількох моделей з метою отримання кращих та більш надійних результатів. Замість використання окремої моделі, ансамбль використовує ансамбль моделей для зроблення прогнозів або прийняття рішень. Існує багато моделей машинного навчання, але найпопулярніші ансамблеві методи включають в себе випадковий ліс (Random Forest) та градієнтний бустінг (Gradient Boosting) (рис.2.10).

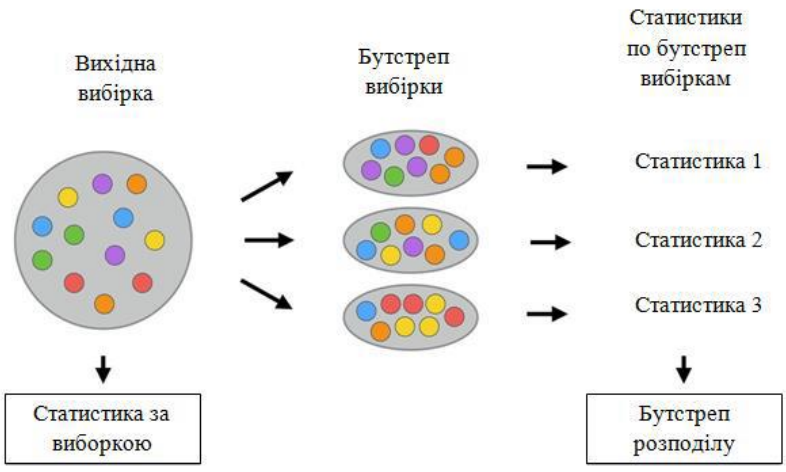


Рисунок 2.10 – Принцип роботи bootstrap

Випадковий ліс – складається з набору дерев рішень. Кожне дерево будується незалежно від інших за допомогою вибірки даних з повторенням (bootstrap sample) та вибору випадкового підмножини ознак.

Випадкові ліси (Random Forests) є потужним ансамблевим методом в машинному навчанні, який комбінує декілька рішень дерев рішень для вирішення задач класифікації та регресії. Вони широко використовуються завдяки своїй

ефективності, надійності та здатності працювати з різноманітними типами даних (рис. 2.11).

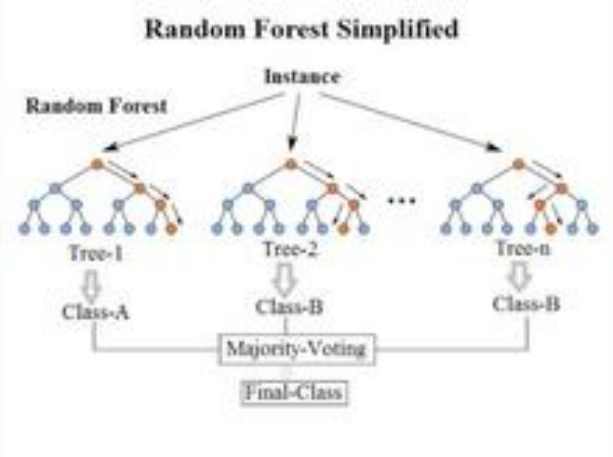


Рисунок 2.11 - Схема випадкового лісу рішень

Хоча випадкові ліси часто забезпечують більш високу точність, ніж одиночне дерево рішень, вони жертвують внутрішньою інтерпретованістю, властивою деревам рішень. Дерева рішень відносяться до досить невеликого сімейства моделей машинного навчання, які легко інтерпретуються поряд з лінійними моделями, моделями на основі правил та моделями на основі уваги. Ця інтерпретованість є одним із найбільш бажаних якостей дерев рішень. Це дозволяє розробникам підтвердити, що модель отримала реалістичну інформацію з даних, і дозволяє кінцевим користувачам довіряти і довіряти рішенням, прийнятим моделлю.

Наприклад, простежити шлях, яким дерево рішень приймає рішення, досить, тривіально, але простежити шлях десятків чи сотень дерев набагато складніше. Для досягнення як продуктивності, так і інтерпретації деякі методи стиснення моделей дозволяють перетворити випадковий ліс у мінімальне «відроджене» дерево рішень, яке точно відтворює ту саму функцію прийняття рішень.

- Статистичний аналіз існуючих даних.

Статистична база даних - це база даних, що використовується для цілей статистичного аналізу. Це система OLAP (онлайн-аналітична обробка), а не OLTP (онлайн-обробка транзакцій). Сучасні рішення та класичні статистичні бази даних часто ближчі до реляційної моделі, ніж багатовимірна модель, яка зазвичай використовується сьогодні в системах OLAP.

Статистичні бази даних зазвичай містять дані параметрів та виміряні дані для цих параметрів. Наприклад, дані параметрів складаються з різних значень різних умов експерименту (наприклад, температури, часу). Виміряні дані (або змінні) - це вимірювання, проведені в ході експерименту в різних умовах.

Багато статистичних баз даних розріджені і містять безліч нульових або нульових значень. Статистична база даних часто буває розрідженою на 40-50%. Є два варіанти боротьби з розрідженістю: (1) залишити там нульові значення та використовувати методи стиснення, щоб видавити їх, або (2) видалити записи, які мають лише нульові значення.

Статистичні бази даних часто включають підтримку передових методів статистичного аналізу, таких як кореляції, які виходять за рамки SQL. Вони також створюють унікальні проблеми безпеки, які були в центрі уваги багатьох досліджень, особливо наприкінці 1970-х і на початку-середині 1980-х років.

Дані були отримані в результаті роботи в онлайн ресурсі Upwork. Головною задачею є передбачення залежних змінних. Унікальність цього проекту є те, що ми маємо невідомий контекст даних. Перед тим як робити аналіз даних, необхідно на них подивитися, в якому вигляді вони представлені.

Першочергово цей набір даних не розглядався, як великій набір даних, оскільки першим варіантом об’єднання даних був інший датасет з середнім значенням кожної колонки. Приклад генерування нового набору даних зображено на рисунку 2.12. Таким чином було отримано новий датасет з розмірністю 182 стрічки та 16 колонок. Результат датасету зображено на рисунку 2.13.

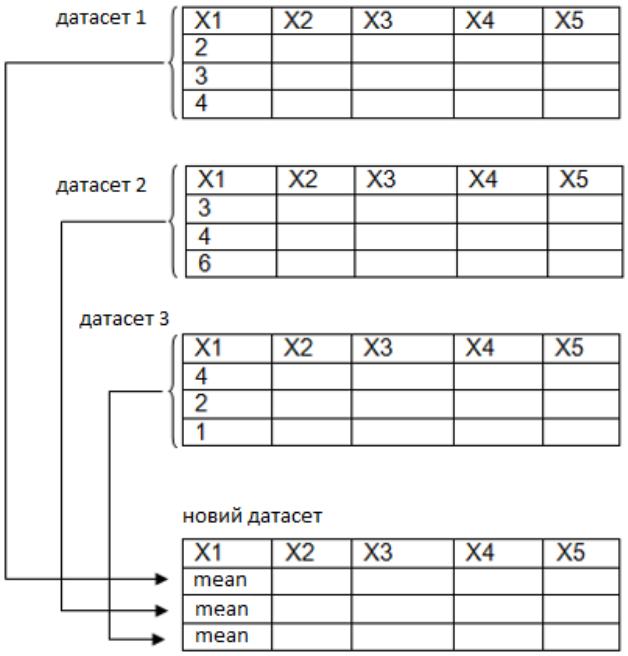


Рисунок 2.12 – Приклад генерування нового набору даних

	0	1	2	3	4
x1	16444.1	19257.4	17690.9	17708.8	18998.3
x2	9310.71	9618.68	12149.7	12453.7	14450.3
x3	0.607377	0.590536	0.621691	0.618072	0.636217
x4	0.591938	0.561094	0.620786	0.617728	0.629457
x5	16.6746	15.6734	24.3281	23.9563	17.8548
x6	7.64276	6.06902	7.91388	8.0901	6.59926
x7	0.587855	0.574968	0.60522	0.600999	0.619322
x8	0.474367	0.4461	0.499742	0.497528	0.505592
x9	16.6335	15.6332	24.2857	23.9142	17.8114
x10	3.95212	2.60032	2.52535	2.78399	2.64725
x11	119.589	119.654	118.685	119.455	118.783
x12	119.667	119.585	118.976	120.036	119.519
x13	365.906	417.493	377.458	379.743	476.132
x14	368.315	416.904	375.419	380.437	478.085
x15	0.074432	0.0731156	0.0748449	0.0734592	0.0736191
x16	0.0703404	0.0820819	0.0564553	0.0549475	0.0557161
target1	106	102	116	106	108
target2	105	102	111	108	109

(181, 18)

Рисунок 2.13 – Результат згенерованого датасету

Після того, як дані були оброблені був проведений певний статистичний аналіз. В першу чергу, датасет був перевірений на наявність:

- пустих комірок;

- нетипових даних – викидів;

- неінформативних даних – дублікатів.

Для аналізу лінійної залежності між кількісними змінними використовуємо діаграми розсіювання, діаграма зображена на рис. 2.14. За допомогою цього графіка ми можемо спостерігати розподіл ознак. Результат цього графіку полягає в тому, щоб зрозуміти, наскільки дані залежні один від одного. Взагалі, між ознаками лінійних стохастичних та нелінійних, залежностей не спостерігається. Щоб переконатися в цьому, буде побудована карта кореляції на рис. 2.15.

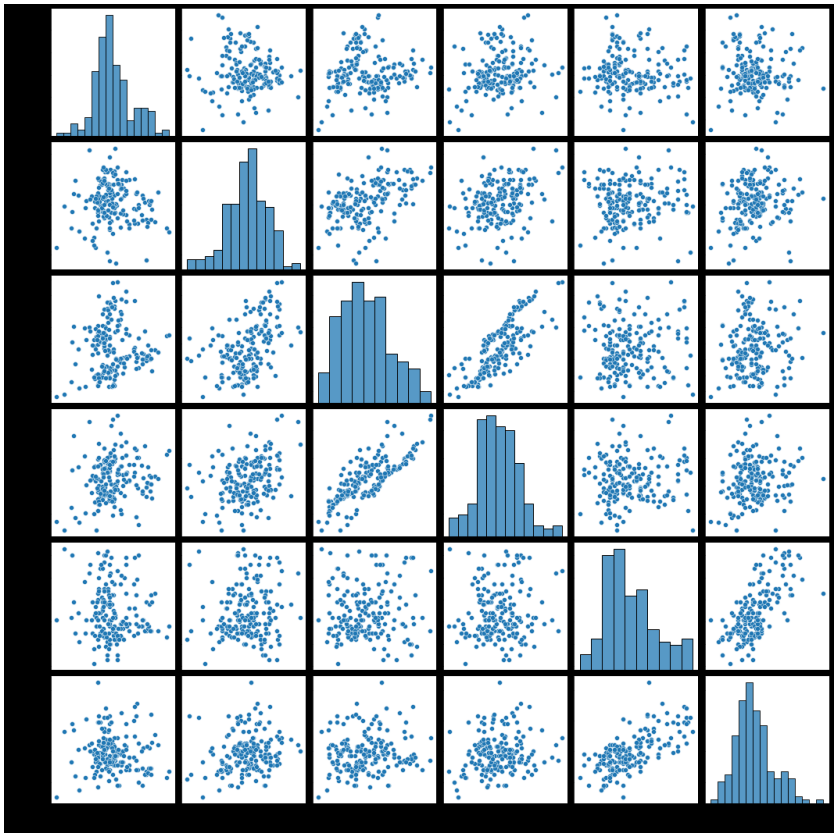


Рисунок 2.14 – Розподіл ознак

Карта кореляції. Тут ми можемо спостерігати, як параметри (особливості) залежать один від одного. Виходячи з попередніх графіків, наші припущення правильні, де 1 на попередньому графіку був розподілом по периметру.

Кореляційна матриця зображена на рис. 2.15.

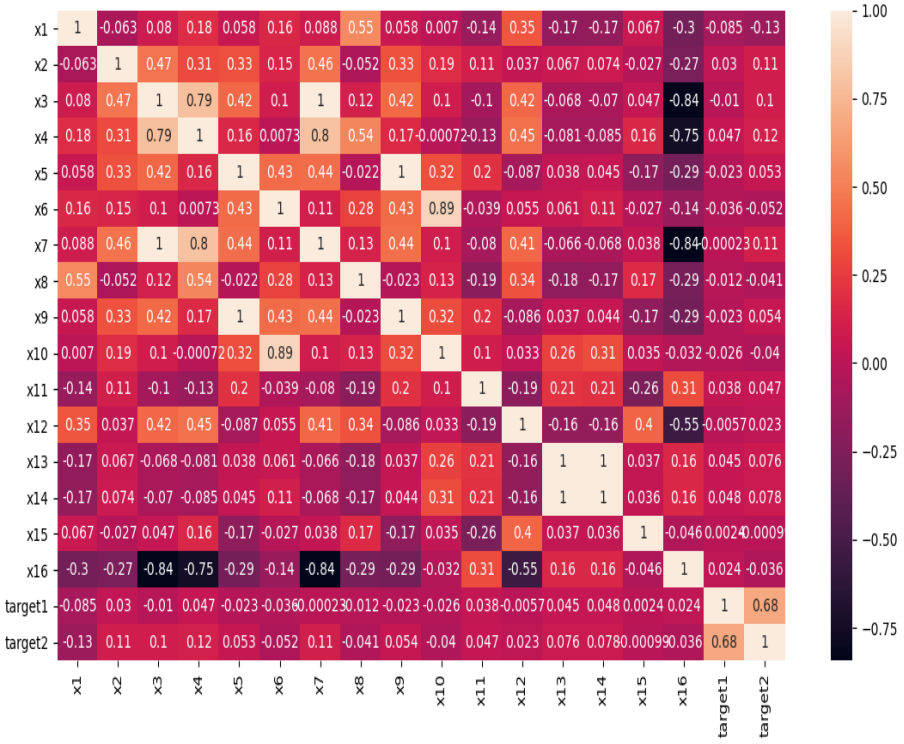


Рисунок 2.15 – Кореляційна матриця

Оскільки попередні варіанти, які ми використовували, не дають нам позитивних результатів, наступним кроком було проаналізувати кожен стовпець, точніше, побачити, як вони розподіляються між собою без урахування змінної. Графіки розподілу зображено на рис. 2.16.

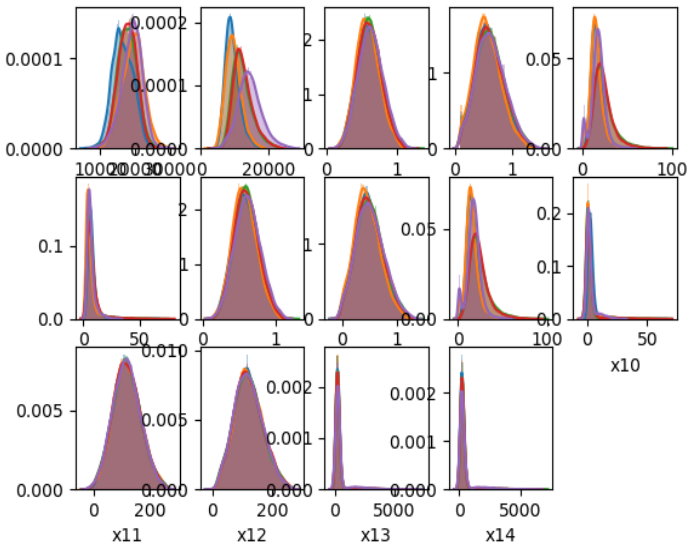


Рисунок 2. 16 – Графіки розподілу пояснювальних змінних

Увагу привернуло згладжування або інакше їх називають хвости. Ці хвости можуть нести за собою різну інформацію, або це викиди або шуми. Згладжування показано червоним кольором. На основі аналізу було вирішено, їх необхідно видалити. Згладжування зображено на рис. 2.17.

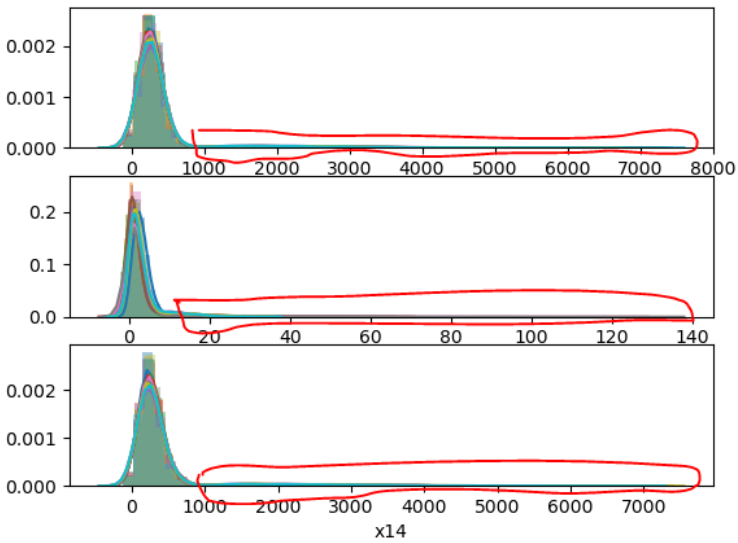


Рисунок 2.17 – Аналіз згладжування

Після тестування було прийняте рішення згруповувати дані не по середньому значенню, а просто кожний файл додавати до одного, таким чином будем мати доволі велику вибірку, приклад генерування великого дата сету зображено на рис.2.14.

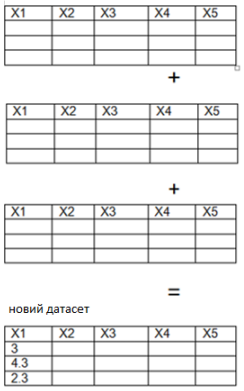


Рисунок 2.18 – Новий великий датасет

Результат великого датасету зображено на рис. 2.19. Також слід зазначити його розмір (1815696 – строк, 18 – стовбців).

	0	1	2	3	4	5	6	7	2
x1	14136.000000	21895.000000	16245.000000	21811.000000	20322.000000	20125.000000	22128.000000	20848.000000	
x2	8773.000000	10789.000000	7822.000000	14270.000000	12093.000000	7661.000000	9260.000000	9338.000000	
x3	0.937134	0.526545	0.422646	0.755819	0.670338	0.393966	0.531751	0.724615	
x4	0.929526	0.434938	0.774747	0.822779	0.536706	0.494276	0.624500	0.703542	
x5	15.771608	11.431559	10.652850	10.800585	15.789350	15.705252	35.385727	9.546813	
x6	3.125166	3.127802	3.445335	2.178386	4.792278	4.405985	14.888521	4.845100	
x7	0.922892	0.518313	0.405828	0.742631	0.658859	0.383159	0.503491	0.706342	
x8	0.744948	0.331276	0.693582	0.674253	0.404934	0.417644	0.523801	0.562274	
x9	15.707006	11.395277	10.624442	10.748601	15.743230	15.678431	35.350483	9.497369	
x10	-0.359916	0.599410	1.087976	-0.206522	1.299158	0.927243	7.044930	2.737817	
x11	230.160848	103.589098	69.569617	173.912968	147.385963	63.332245	95.961712	159.703267	
x12	181.955516	96.973361	170.712982	167.314556	112.300623	114.961427	135.112770	143.571559	
x13	74.270311	283.590717	253.445205	146.779673	214.344640	199.272807	584.604310	389.802436	
x14	479.626285	226.649362	255.879338	377.384050	296.904523	313.747044	10.712542	137.268228	
x15	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
x16	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
target1	111.000000	111.000000	111.000000	111.000000	111.000000	111.000000	111.000000	111.000000	
target2	121.000000	121.000000	121.000000	121.000000	121.000000	121.000000	121.000000	121.000000	

18 rows × 1815696 columns

Рисунок 2.19 – Генерування великого набору

Оскільки маємо дуже великий набір даних і зрозуміло, що оперувати такою кількістю даних спроможні лише технології Big Data. Таким чином було прийняте рішення розглядати цю задачу, як з великим набором даних, але за допомогою методів data science.

2.3 Висновки за розділом

Підсумовуючи другий розділ, можемо зробити такі висновки:

1. Розглянуто сучасних моделей машинного навчання, побудованих на Big Data.
2. Окреслено підходи до створення ефективної системи навчання для прогнозування неперервних чисельних ознак для великого набору даних
3. Обгрунтовано вибір алгоритму машинного навчання для прогнозування неперервних чисельних ознак для великого набору даних.

3 ОСОБЛИВОСТІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СЕРВІСУ ДЛЯ ПРОГНОЗУВАННЯ НА BIG DATA МОВОЮ PYTHON

3.1. Розробка алгоритму прогнозування методом Dask

Аналіз та побудова моделей машинного навчання справа не із легких, якщо у вас оперативна пам’ять не перевищує 4 Гб. Це звичайна проблема, з якою стикаються фахівці з даними при роботі з обмеженими обчислювальними ресурсами. З такою задачею ми зіштовхнулись при розробці дипломного проекту.

Використовувати технології Big data, не є актуальною у даному випадку, адже багато факторів вказують на те, що задачу можна виконати завдяки паралелізації, а саме бібліотеки котра працює на основі паралелізації обчислень. Коли було побудовано новий датасет, майже відразу стало зрозумілим, що існуючі бібліотеки мають певні обмеження, коли справа доходить до обробки великих наборів даних. Pandas і Numpy – відмінні бібліотеки, але вони не завжди ефективні з обчислювальної точки зору, особливо коли є гігабайти даних, якими потрібно маніпулювати.

Dask - це відкрита бібліотека для розподіленого обчислення в масштабі даних, яка надає можливості паралельного обчислення і обробки даних. Вона побудована на основі Python і призначена для роботи з великими обсягами даних, які не поміщаються в пам'ять одного комп'ютера.

В даній роботі ми використовуємо загальні бібліотеки Python, такі як pandas, numpy і scikit – learn, для попередньої обробки даних і побудови моделей. Ці бібліотеки зазвичай добре працюють, якщо набір даних підходить до існуючої оперативної пам’яті. Але ці бібліотеки не масштабуються і працюють на одному процесорі. Однак dask може масштабуватися до кластеру комп’ютера.

Dask може ефективно виконувати паралельні обчислення на одній машині з використанням багатоядерних процесорів. Щоб використовувати менший обсяг пам’яті під час обчислень, Dask зберігає повні дані на диску і використовує для обробки фрагменти даних (менші частини, а не всі дані) з диска. Під час обробки згенеровано проміжні значення (якщо такі є) якомога швидше відкидаються, щоб скоротити споживання пам’яті.

Таким чином, dask може працювати на кластері машин для ефективної обробки даних, оскільки він використовує всі ядра підключених машин. Цікавим є той факт, що необов’язково, щоб на всіх машинах була однакова кількість ядер. Якщо одна система має 2 ядра, а інша – 4 ядра, dask може обробляти ці варіанти внутрішньо.

Dask надає кілька користувальницьких інтерфейсів, кожен з яких має свій набір паралельних алгоритмів для розподілених обчислень. Існують такі важливі інтерфейси масштабування numpy, pandas і scikit–learn:

- масиви: паралельний Numpy;
- датафрейми: паралельний Pandas;
- машинне навчання: паралельний Scikit– Learn.

Стосовно масивів numpy то вони діляться на більш дрібні масиви, які при групуванні разом утворюють масив dask. Простіше кажучи, масиви dask – це розподілені масиви numpy. Кожна операція з масивом dask запускає операції з меншими масивами numpy, кожен з яких використовує ядро на машині. Таким чином, всі доступні ядра використовуються одночасно, що дозволяє виконувати обчислення на масивах, розмір яких перевищує розмір пам’яті. На рис.3.1 наведено зображення, яке демонструє як виглядає масив.

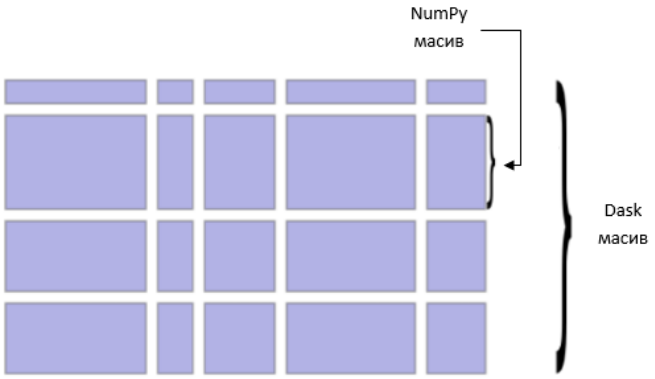


Рисунок 3.1 – Масив Dask

Як можна побачити, кілька масивів numpy організовані в сітки, щоб сформувати масив dask. При створенні масиву dask можна вказати розмір блоку, який визначає розмір масивів numpy. Таким чином, наведено кілька важливих функцій масивів dask:

– паралелізація: масиви dask використовують всі ядра системи;

– більше ніж існуюча пам’ять: дозволяє працювати з наборами даних, розмір яких перевищує обсяг доступної в системі пам’яті;

– заблоковані алгоритми: виконувати великі обчислення, виконуючи безліч менших обчислень.

Подібно масиву dask, фрейм даних dask складається з декількох менших фреймів даних pandas. Фрейм даних великий pandas розбивається по рядках, щоб сформувати кілька менших фреймів даних. Ці менші фрейми даних присутні на диску однієї або декількох машин (що дозволяє зберігати набори даних розміром більше пам’яті). Кожне обчислення в фреймі даних dask розпаралелює операції з існуючими кадрами даних pandas. На рис. 3.2 зображено, як представляє структуру фрейму даних dask.

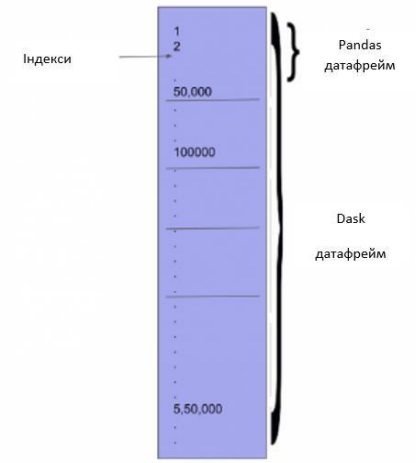


Рисунок 3.2 – Фрейм даних dask

В таблиці 3.1 зображені порівняльний час відкриття даних використовуючи pandas та dask . Розмір даних (1815696 – строк, 18 – стовбців).

Таблиця 3.1 - Порівняльна характеристика швидкості зчитування

	dask	pandas
CPU times	0,0202 сек	5.76 сек
Wall time	0,0277 сек	6.66 сек

Dask добре адаптований під машинне навчання, надає масштабовані алгоритми на Python, які сумісні з scikit–learn. Користувач може виконувати паралельні обчислення з допомогою scikit–learn (на одній машині), задавши будь – який параметр. Scikit–learn використовує Joblib для виконання цих паралельних обчислень. Joblib – це бібліотека на Python, яка забезпечує підтримку

розпаралелювання. Joblib розподіляє завдання за доступними ядрами. Принцип розпаралелювання зображений на рис.3.3.

Рисунок 3.3 – Принцип розпаралелювання

Як висновок можна сказати, що dask є доволі зручний та не складний у налагодженні, необхідно лише встановити деякі бібліотеки для функціонування. Оптимізує швидкість роботи алгоритмів та аналізу великих наборів даних.

3.2 Статичний аналіз великого набору даних

Було прийняте рішення зібрати дані іншим чином, в результаті отримали великий набір даних 1815696 – строк. Спершу необхідно підключитися до бібліотеки dask та виділити об’єм пам’яті котрий необхідний, в даному випадку це 8 Гб. Результати зображено на рис 3.5. Дані зображені на рисунку 3.6.

```
client = Client(n_workers=2, threads_per_worker=1, memory_limit='4GB',processes=False)
client
```

Client
- **Scheduler:** inproc://172.28.0.2/55/22

Cluster
- **Workers:** 2
- **Cores:** 2
- **Memory:** 8.00 GB

Рисунок 3.5 – Результат підключення до бібліотеки

	0	1	2	3	4
x1	14136.000000	21895.000000	16245.000000	21811.000000	20322.000000
x2	8773.000000	10789.000000	7822.000000	14270.000000	12093.000000
x3	0.937134	0.526545	0.422646	0.755819	0.670338
x4	0.929526	0.434938	0.774747	0.822779	0.536706
x5	15.771608	11.431559	10.652850	10.800585	15.789350
x6	3.125166	3.127802	3.445335	2.178386	4.792278
x7	0.922892	0.518313	0.405828	0.742631	0.658859
x8	0.744948	0.331276	0.693582	0.674253	0.404934
x9	15.707006	11.395277	10.624442	10.748601	15.743230
x10	-0.359916	0.599410	1.087976	-0.206522	1.299158
x11	230.160848	103.589098	69.569617	173.912968	147.385963
x12	181.955516	96.973361	170.712982	167.314556	112.300623
x13	74.270311	283.590717	253.445205	146.779673	214.344640
x14	479.626285	226.649362	255.879338	377.384050	296.904523
x15	0.000000	0.000000	0.000000	0.000000	0.000000
x16	0.000000	0.000000	0.000000	0.000000	0.000000
target1	111.000000	111.000000	111.000000	111.000000	111.000000
target2	121.000000	121.000000	121.000000	121.000000	121.000000

Рисунок 3.6 – Побудова нового набору даних

Після того, як дані були зібрані та відображені, наступним кроком був первинний аналіз, тобто таким чином зрозуміти, що дані не мають пропусків або інших символів. Спочатку у процентному відношенні перевіримо список часткою відсутніх записів для кожної ознаки. Результат зображено в рис.3.7.

```
x1 - 0.0%
x2 - 0.0%
x3 - 0.0%
x4 - 0.0%
x5 - 0.0%
x6 - 0.0%
x7 - 0.0%
x8 - 0.0%
x9 - 0.0%
x10 - 0.0%
x11 - 0.0%
x12 - 0.0%
x13 - 0.0%
x14 - 0.0%
x15 - 0.0%
x16 - 0.0%
target1 - 0.0%
target2 - 0.0%
```

Рисунок 3.7 – Перевірка пропущених значень вибірки

Далі переглянемо на наявність текстових значень. Результат виконання зображено на рисунок 3.8.

```
x1         0
x2         0
x3         0
x4         0
x5         0
x6         0
x7         0
x8         0
x9         0
x10        0
x11        0
x12        0
x13        0
x14        0
x15        0
x16        0
target1    0
target2    0
```

Рисунок 3.8 – Нестандартні пропущені значення

Наступним кроком створили рафік розподілу цільових зразків синій – тренування, помаранчевий – тестування, зелений загальний. Наскільки цільові змінні подібні одна до одної. Графік щільності зображений на рисунку 3.9.

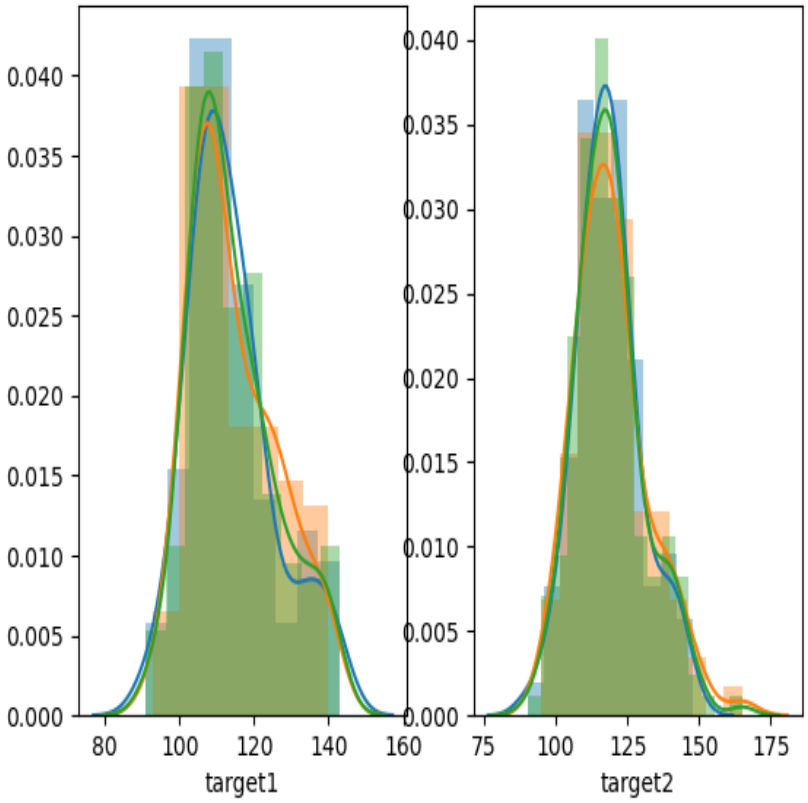


Рисунок 3.9 – Графік щільності розподілу цільових змінних

Графік розсіювання котрий зображений на рисунку 3.10, демонструє нам наскільки данні залежні один від одного, а також розсіювання даних. Завдяки цьому графіку можна упевнятися в тому чи потрібно чистити дані іншими методами, або застосовувати інший підхід до даних.

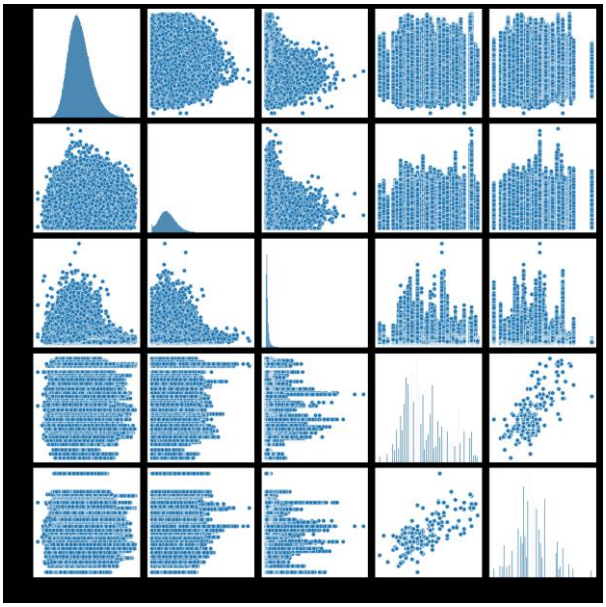


Рисунок 3.10 – Діаграми розсіювання залежних та пояснювальних змінних

Для того, щоб впевнитись в тому, наскільки часто зустрічаються ті чи інші значення X, будуємо діаграму частот. Кожен стовпець гістограми показує частоту потрапляння значення вибірки в інтервал значень – чим вище стовпчик, тим імовірніше відповідні значення показника. Гістограма зображена на рис 3.11.

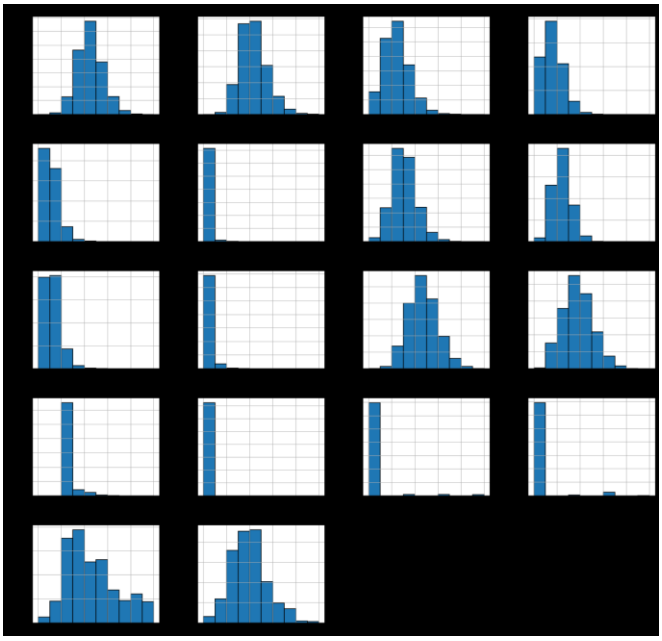


Рисунок 3.11 – Гістограми розподілу даних

Наступним кроком використовували карту кореляції. Тут ми можемо спостерігати, як параметри (особливості) залежать один від одного. Виходячи з попередніх графіків, наші припущення правильні, де 1 на попередньому графіку був розподілом по периметру. Карта зображена на рисунку 3.12.

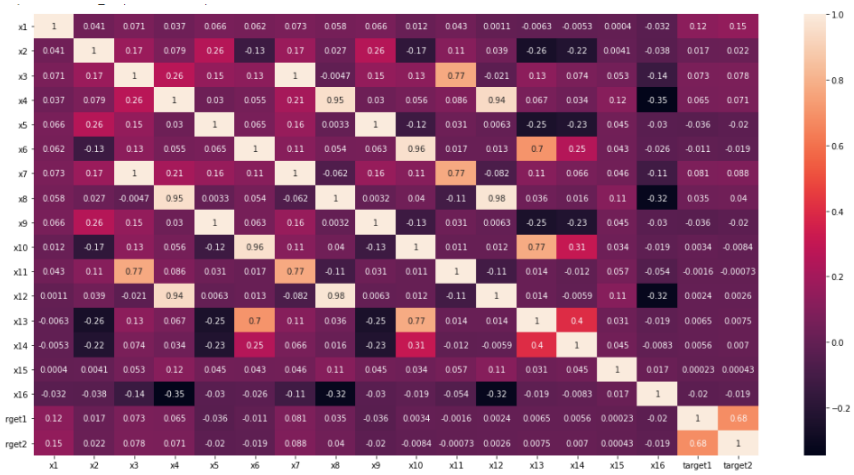


Рисунок 3.12 – Кореляційна карта – матриця

Після кореляційної карти було прийнято рішення видалити ті змінні, які мають дуже високу кореляцію, тобто ніякої інформативності вони не несуть. Були видалені такі змінні як: x7, x5, x8, x10, x14, x15, x16. Після видалення даних ми отримали вибірку такого вигляду, як зображено на рис. 3.13.

	x2	x3	x4	x6	x9	x11	x12	x13	target1	target2
0	8773	0.937134	0.929526	3.125166	15.707006	230.160848	181.955516	74.270311	111	121
1	10789	0.526545	0.434938	3.127802	11.395277	103.589098	96.973361	283.590717	111	121
2	7822	0.422646	0.774747	3.445335	10.624442	69.569617	170.712982	253.445205	111	121
3	14270	0.755819	0.822779	2.178386	10.748601	173.912968	167.314556	146.779673	111	121
4	12093	0.670338	0.536706	4.792278	15.743230	147.385963	112.300623	214.344640	111	121
...	...	...	...	...	...	...	...	...	...	...
1815691	13855	0.444579	0.944804	16.943530	9.929977	61.825721	158.095613	3942.412923	106	105
1815692	8183	0.771922	0.674117	8.652344	2.028285	156.530470	106.108712	2434.390274	106	105
1815693	11037	0.626786	0.886155	7.217694	14.590242	114.204442	141.366510	1036.541148	106	105
1815694	8055	0.493443	0.724412	13.149114	8.804448	80.185968	122.784254	3032.202558	106	105
1815695	7353	0.353857	1.341039	9.910459	3.079820	29.159254	217.796809	2656.283451	106	105

1815696 rows × 10 columns

Рисунок 3.13 – Дані після видалення не інформативних змінних

Дані мають два типи, такі як, цілочисельні, а також числа з плаваючою точкою. Для того, щоб вони мали один вигляд була застосована стандартизація. Ідея стандартизації полягає в тому, що він перетворює дані так, що їх розподіл буде мати середнє значення 0 і стандартне відхилення 1. З огляду на розподіл даних, кожне значення в наборі даних буде відніматися із середнього значення вибірки, а потім ділитися на стандартне відхилення всього набору даних. Стандартизовані дані зображені на рисунку 3.14.

x1	0.375985	0.641868	0.448256	0.638990	0.587965	0.581214	0.649853	0.605990
x2	0.267968	0.329546	0.238920	0.435872	0.369376	0.234002	0.282843	0.285226
x3	0.355502	0.181139	0.137017	0.278504	0.242203	0.124837	0.183350	0.265253
x4	0.253043	0.102171	0.205828	0.220480	0.133215	0.120272	0.159996	0.184107
x5	0.086409	0.062479	0.058186	0.059000	0.086507	0.086044	0.194558	0.052087
x6	0.009036	0.009044	0.009992	0.006208	0.014015	0.012862	0.044172	0.014173
x7	0.418574	0.257907	0.213236	0.346988	0.313721	0.204234	0.252021	0.332577
x8	0.321932	0.212488	0.308343	0.303229	0.231976	0.235338	0.263424	0.273603
x9	0.096200	0.072690	0.068487	0.069164	0.096397	0.096044	0.203307	0.062341
x10	0.057737	0.060438	0.061813	0.058169	0.062408	0.061361	0.078584	0.066458
x11	0.741563	0.429539	0.345674	0.602901	0.537507	0.330298	0.410736	0.567871
x12	0.535364	0.321671	0.507093	0.498548	0.360212	0.366903	0.417575	0.438845
x13	0.252685	0.265898	0.263995	0.257262	0.261527	0.260576	0.284900	0.272603
x14	0.002307	0.001779	0.001840	0.002094	0.001926	0.001961	0.001328	0.001592
x15	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
x16	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
target1	0.384615	0.384615	0.384615	0.384615	0.384615	0.384615	0.384615	0.384615
target2	0.413333	0.413333	0.413333	0.413333	0.413333	0.413333	0.413333	0.413333

Рисунок 3.14 – Стандартизовані дані

3.3 Аналіз результатів машинного навчання

Далі побудуємо моделі машинного навчання. Оскільки ми прийшли до того, що маємо задачу регресії, адже головною задачею є передбачення цільових змінних. Будувати моделі будемо ті ж самі які будували на минулих даних. Оскільки не задоволені результати дали зрозуміти, що даних мало, і моделі перенавчаються. Тому було прийняте використати такі методи, як: лінійна регресія, випадковий ліс,

дерево рішень і визначимо найкращу прогнозуючу модель. Всі методи будуть розглядатися з точки зору регресії. Результати похибки прогнозів використаних методів представимо на рис. 3.15.

Метод	Похибка прогнозної якості	Target1 (цільова змінна)		Target2 (цільова змінна)	
		train	test	train	test
Лінійна регресія	MSE	0.02	0.04	0.01	0.02
	MAE	0.13	0.17	0.9	0.11
	MAPE	14.20	16.67	14.28	16.41
	SCORE	0.13	0.10	0.13	0.13
Дерево рішень	MSE	0.01	0.02	0.0	0.01
	MAE	0.06	0.11	0.03	0.07
	MAPE	7.49	10.63	3.15	6.55
	SCORE	0.76	0.53	0.75	0.58
Випадковий ліс	MSE	0.02	0.04	0.01	0.03
	MAE	0.01	0.03	0.02	0.04
	MAPE	2.54	3.46	1.0	1.96
	SCORE	0.90	0.88	0.93	0.91

Рисунок 3.15 – Результати похибки прогнозів використаних методів

Окрім візуалізації прогнозної якості були побудовані у графічному вигляді дерева регресії для обох цільових змінних. Саме графічно можна зрозуміти, яким чином йде розподіл. Дерево для target1 зображено на рисунку 3.16. Дерево для target 2 зображено на рисунку 3.17. Як можна побачити, ми беремо підмножину даних і вирішуємо, як найкраще розділити підмножину.

Нашою початковою підмножиною був весь набір даних, і ми розділили його відповідно до правила $X <= 0.258$. Потім для кожної підмножини ми виконали додаткове розбиття до тих пір, поки не змогли правильно передбачити цільову змінну, не дотримуючись обмеження стосовно глибини дерева.

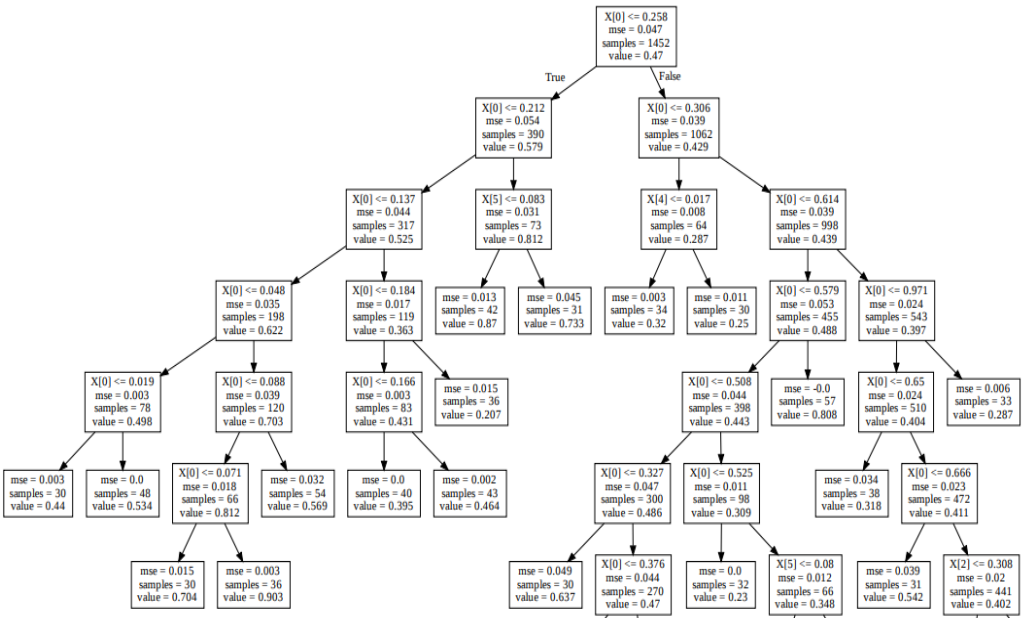


Рисунок 3.16 – Дерево для цільової змінної (target1)

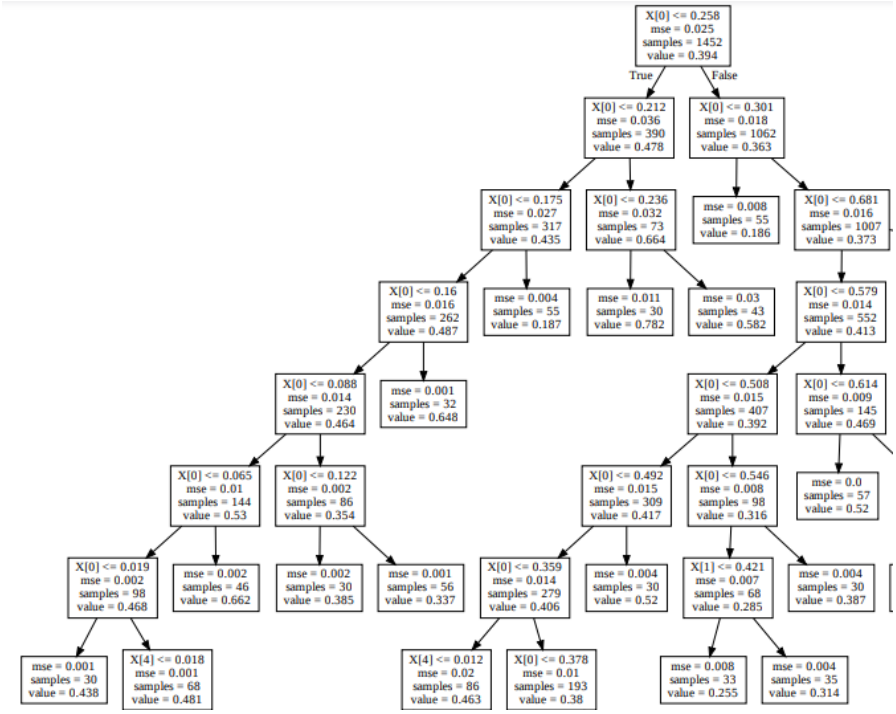


Рисунок 3.17 – Дерево для цільової змінної (target2)

На рисунку 3.18 візуалізовано результати методів. На графіках зображено фактичні і прогнозні дані. Можна побачити, що модель випадкового лісу набагато краще прогнозує цільові змінні, оскільки прогнозні значення приближені до фактичних вздовж бісектриси. Дані результати влаштовують потреби замовника і будуть передані йому.

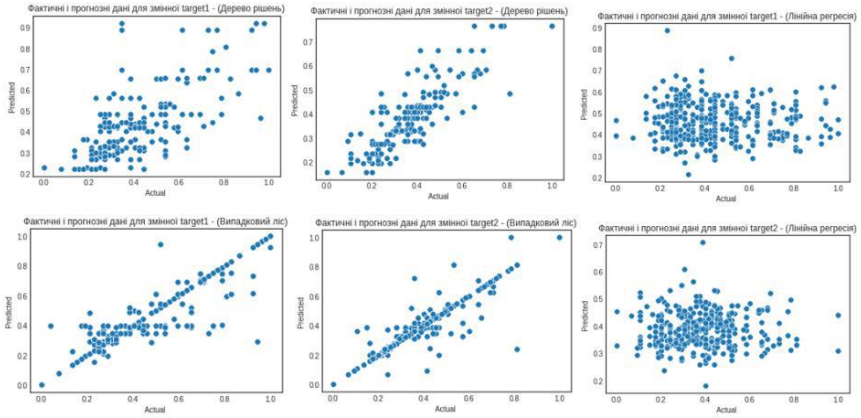


Рисунок 3.18 – Результати прогнозних та фактичних даних

Для того, щоб узагальнити весь функціонал данної розробки, на рис. 3.19 зображена узагальнена блок схема роботи даної моделі.

Всі результати котрі були продемонстровані, вони говорять про те що найбільш ефективною виявилась модель машинного навчання випадковий ліс. Після чого розробився зручний програмний код, щоб замовник з легкістю міг оперувати данною системою. Для того, щоб спрогнозувати інші дані, необхідно у кореневий каталог дані завантажити, після чого через командну стрічку операційної системи запустити програмний код. Всі результати будуть надходити та зберігатись у табличному редакторі excel файл в такому вигляді, як зображено на рисунку 3.20.

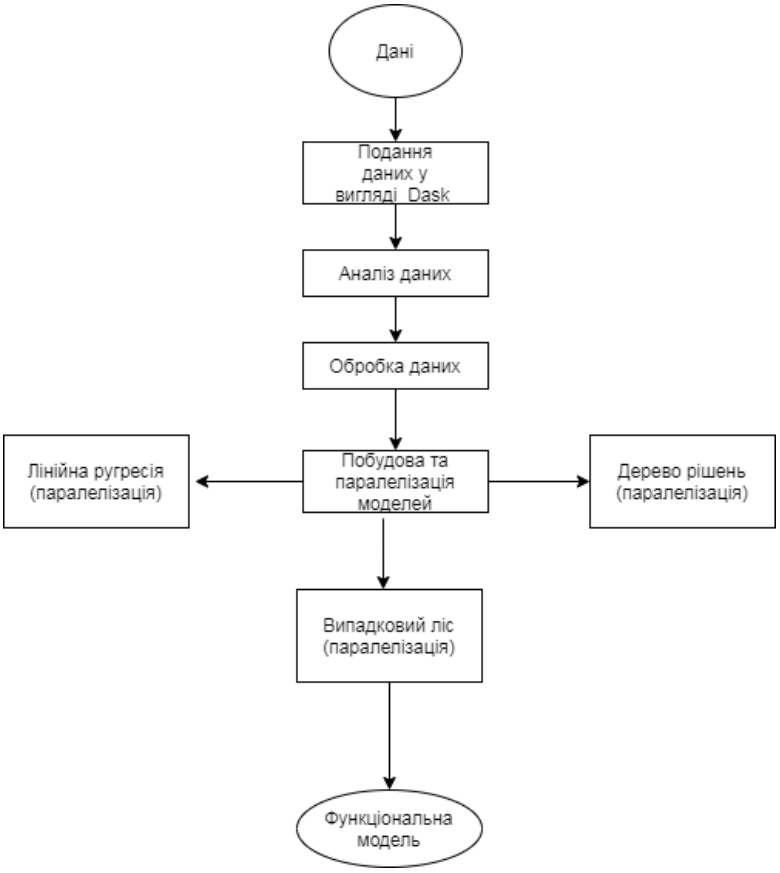


Рисунок 3.19 – Блок схема роботи моделі

A	B	C	D
target1	target2	pred_tar_1	pred_tar_2
106	105	0,2404	0,3753
102	102	0,6273	0,3805
116	111	0,3092	0,3641
106	108	0,3114	0,3801
108	109	0,4808	0,3727
111	122	0,486	0,3853
119	107	0,4448	0,4105
106	105	0,4846	0,4313
107	106	0,4308	0,3845
103	103	0,4229	0,4903
140	125	0,2837	0,3288
106	105	0,4297	0,3395

Риссунок 3.20 – Отриманий результат

3.4 Моделювання вимог до сервісу

Моделювання вимог до сервісу є важливим кроком у процесі розробки. Це допомагає зрозуміти, які функціональні та нефункціональні вимоги має задовільняти сервіс, а також визначити, які функціональності повинні бути реалізовані.

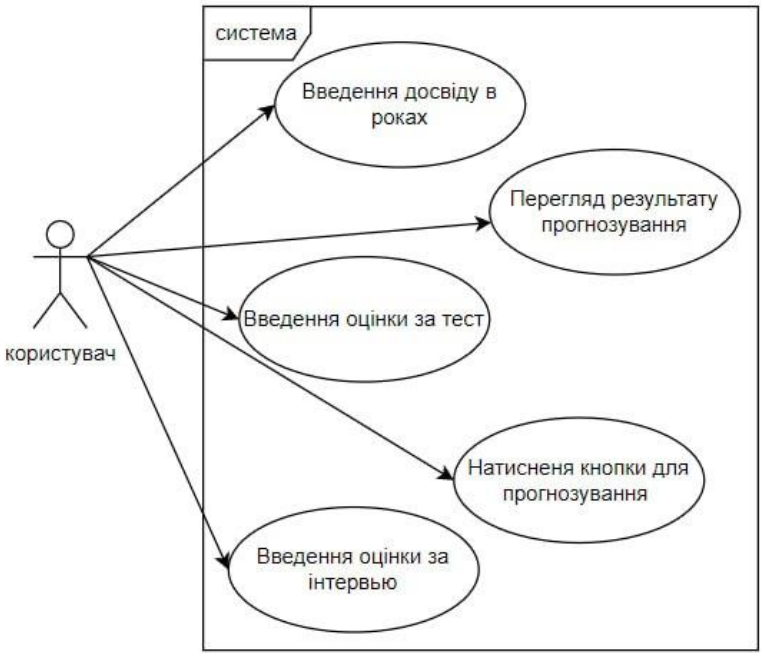
Функціональні вимоги:

1. Можливість завантажувати великі набори даних і проводити їх попередню обробку, таку як очищення, нормалізацію та видалення відсутніх значень.
2. Можливість вибору та навчання різних моделей машинного навчання на основі вхідних даних.
3. Використання навчених моделей для прогнозування на нових наборах даних.

Не функціональні вимоги:

1. Час відповіді на запит має бути мінімальним, а прогнозування займає прийнятий час.
2. Забезпечення високого рівня безпеки для захисту конфіденційності та цільності даних.
3. Комфортний та зрозумілий інтерфейс користувача.

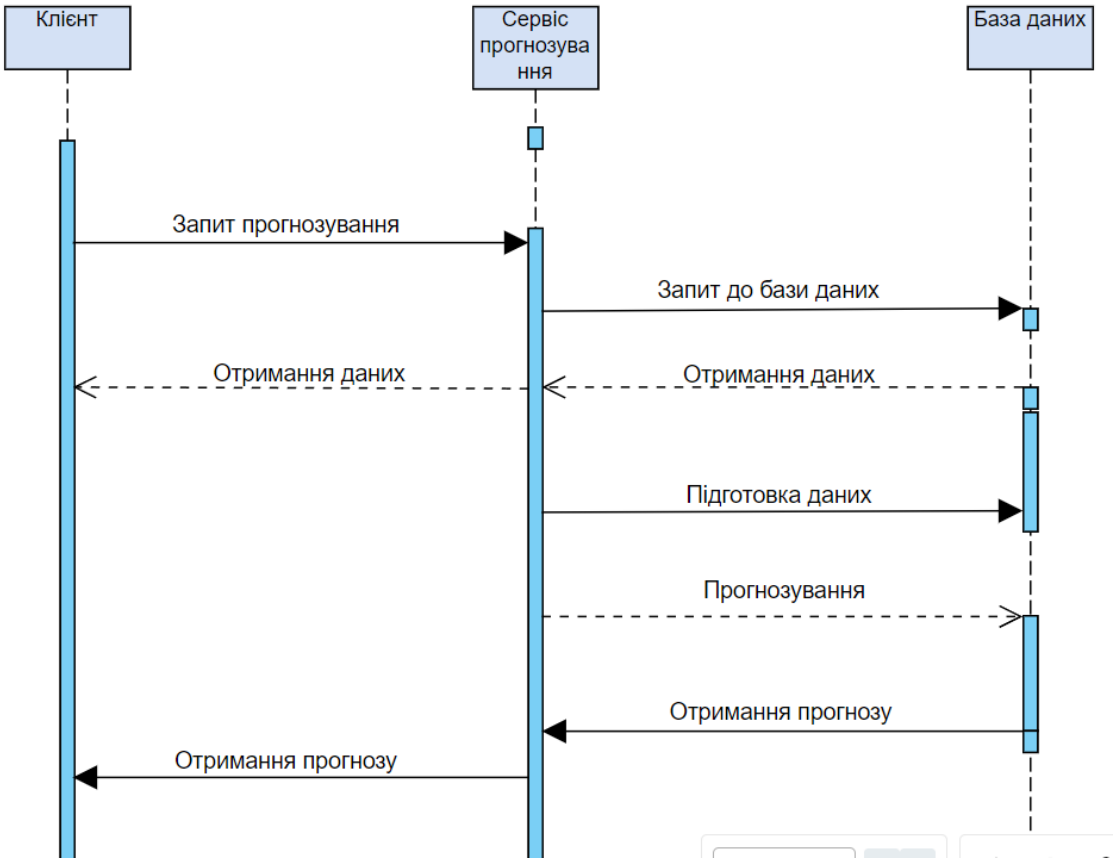
На рисунку 3.21 наведено діаграму варіантів варіантів використання сервісу. Основні прецеденти користувача включають наступні дії : введення досвіду в роках, введення оцінки за тест, введення оцінки за інтерв’ю, перегляд результату прогнозування, натиснення кнопки для прогнозування.



3.21 - UML діаграма варіантів використання

3.5 Моделювання роботи сервісу

Для того , щоб проаналізувати логіку взаємодії між компонентами системи було вирішено побудувати діаграму послідовності. Вона допомагає зрозуміти, які об’єкти спілкуються між собою, які повідомлення або запити передаються та в якому порядку. У цій діаграмі послідовності (рис. 3.22) зображено взаємодію між клієнтом, сервісом прогнозування та базою даних.



3.22 – Діаграма послідовності

3.6 Реалізація сервісу прогнозування зарплат

Для реалізації сервісу ми використовуємо бібліотеку pandas для завантаження даних з CSV-файлу і подальшої обробки. З допомогою бібліотеки scikit-learn ми розділяємо дані на навчальний та тестовий набори, навчаємо модель лінійної регресії на навчальних даних та прогнозуємо зарплату для тестових даних. Використовується середньоквадратична помилка (Mean Squared Error) для оцінки точності моделі. Нарешті, навчена модель зберігається на диск за допомогою модуля pickle.

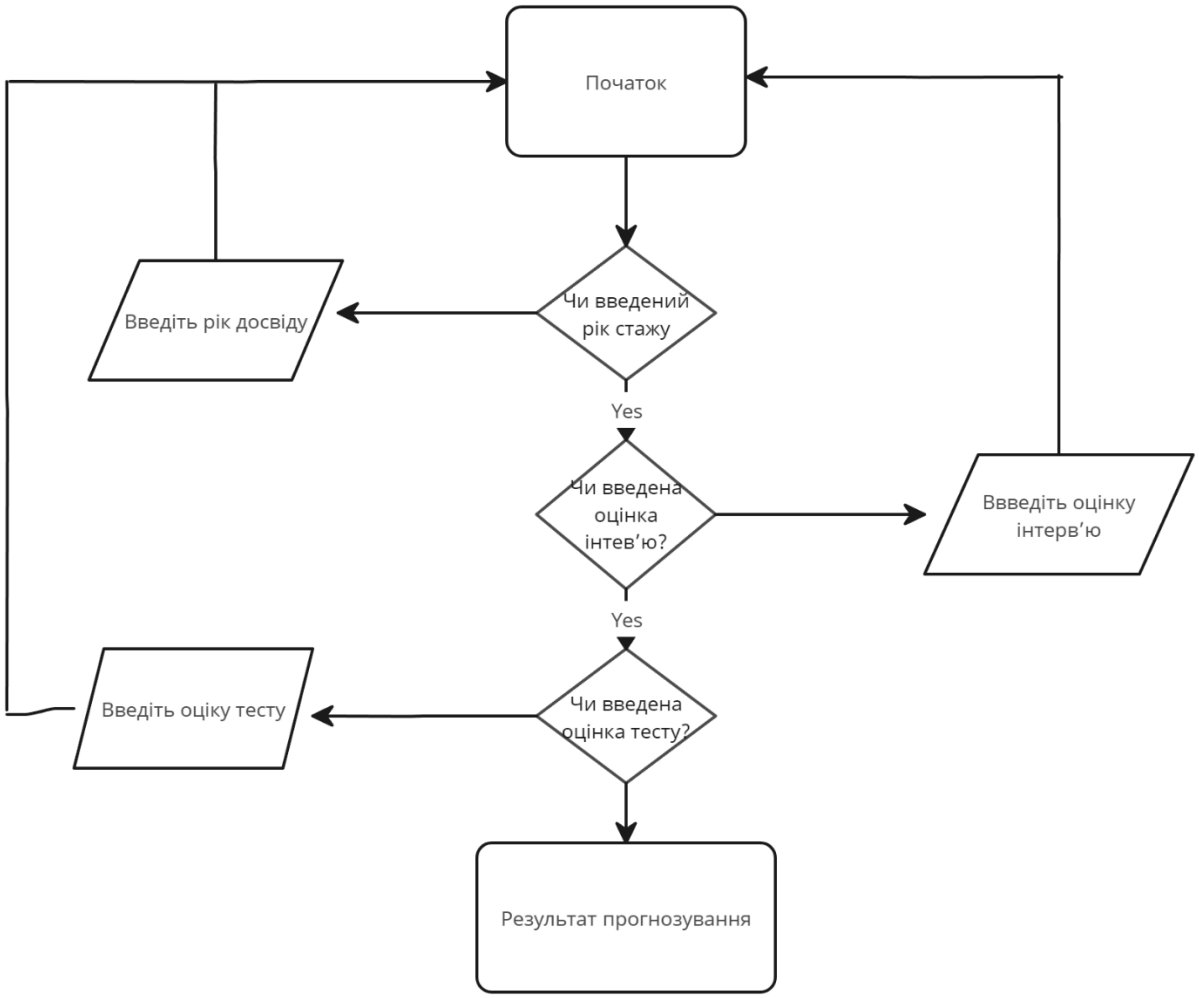
main.py

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import pickle

data = pd.read_csv('salary_data.csv')
X = data['YearsExperience'].values.reshape(-1, 1)
y = data['Salary'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Навчання моделі лінійної регресії
model = LinearRegression()
model.fit(X_train, y_train)

# Прогнозування зарплати
y_pred = model.predict(X_test)

# Оцінка точності моделі
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")

# Збереження моделі на диск
with open('salary_prediction_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

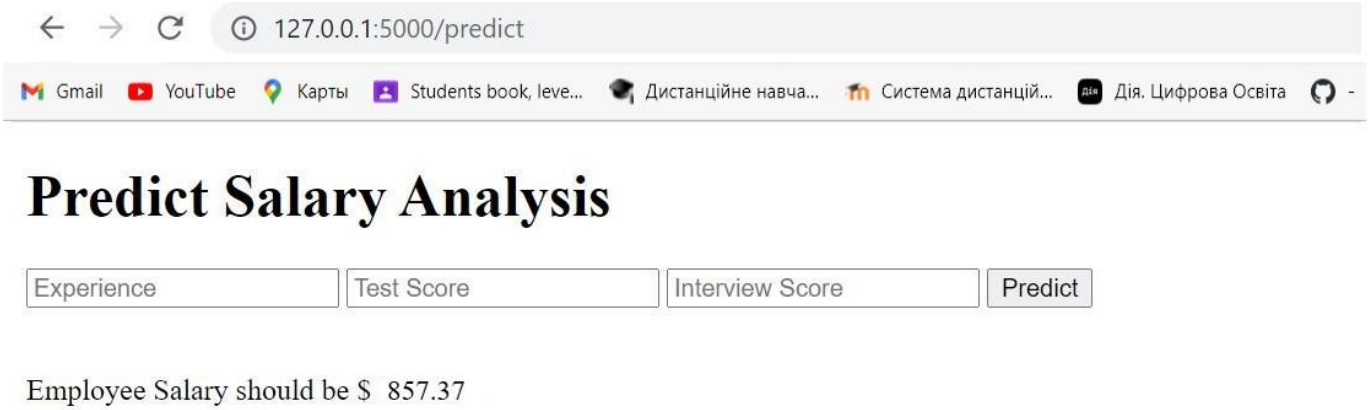
3.23 - Вигляд коду реалізації сервісу для прогнозування

3.7 - Розробка інтерфейсу користувача

Для розробки інтефейсу користувача використовували HTML та Flask , де користувач може вводити дані про рік досвіду, оцінку тестування та оцінку інтерв'ю. Для того, щоб побачити алгоритм роботи побудовано блок – схему (рис.3.24).



3.24 – Блок- схема роботи сервісу



3.25 – Вигляд сторінки сервісу

ВИСНОВКИ

Підсумовуючи загальний зміст дипломного проекту, можемо констатувати наступне:

1. Визначено, що великі дані в першу чергу відносяться до наборів даних, які надто великі або складні для обробки за допомогою традиційного програмного забезпечення для обробки даних. Дані з великою кількістю записів (рядків) мають більшу статистичну силу, а дані з більш високою складністю (більше атрибутів або стовпців) можуть призвести до більш високого рівня помилкових виявлень. Хоча інтерпретація, яка іноді використовується вільно, частково через відсутність формального визначення, здається, що найкраще описує великі дані, яка пов'язана з великим обсягом інформації, яку ми не можемо зрозуміти, якщо використовувати тільки в менших кількостях.

2. З'ясовано, що проблеми аналізу великих даних включають збір даних, зберігання даних, аналіз даних, пошук, спільне використання, передачу, візуалізацію, запити, оновлення, конфіденційність інформації та джерело даних.

3. Досліджено, що поява платформ на основі відкритого коду, таких як Hadoop та пізніше Spark, відіграла значну роль у поширенні великих даних, оскільки ці інструменти спрощують обробку великих даних та знижують вартість зберігання. За минулі роки обсяги високих даних зросли на порядки. Величезні обсяги даних з'являються в результаті діяльності користувачів, але тепер не лише їх.

4. Проаналізовано, що Dask - це бібліотека з відкритим кодом, призначена для забезпечення паралелізму з існуючим стеком Python. Він забезпечує інтеграцію з бібліотеками Python, такими як NumPy Arrays, Pandas DataFrames та scikit-learn, щоб забезпечити паралельне виконання на кількох ядрах, процесорах та комп'ютерах без необхідності вивчення нових бібліотек чи мов. Кожна з трьох паралельних колекцій Dask - DataFrames, Bags і Arrays - може автоматично використовувати дані, розділені

між ОЗП і диском, розподілені кількома вузлами в кластері, залежно від доступності ресурсів. Для завдань, які можна розпаралелити, але які погано вписуються в абстракції високого рівня, такі як Dask Arrays або DataFrames, існує відкладена функція, яка використовує декоратори Python для зміни функцій, щоб вони працювали ліниво. Це означає, що виконання затримується, а функція та її аргументи поміщаються у граф задач.

5. В результаті визначено альтернативний метод використання засобів машинного навчання в процесі розробки сервісу для прогнозування на великих наборах даних. На основі результатів виконаних досліджень розроблено модель швидкої адаптації великих даних для машинної обробки та прогнозування методом Dask.

Упровадження розробленої методики дозволяє прискорити процес навчання, а також виконання певних статичних функцій. Після навчання моделей, обиралась найкраща, за критеріями оцінки якості моделі, а також за оцінками прогнозної якості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Береза А. М., Основи створення інформаційних систем. Навчальний посібник. 2-ге видання. – К.: КНЕУ, 2001р. – 156 с.

2. Вороновський Г. К., Махотило К. В., Петрашев С. Н., Сергєєв С. А. Генетичні алгоритми, штучні нейроні мережі і проблеми віртуальної реальності. - Замовне. - Х.: ОСНОВА, 1997. - С. 100-112.

3. Глушков В. М., Амосов М. М., Артеменко І. О. Енциклопедія кібернетики. Том 2. Київ, - 1974. – С. 33-54.

4. Головач Ю. Складні мережі / Ю. Головач, О. Олємской, К. фон Фербер та ін. // Журнал фізичних досліджень. - 2006. - 10, № 4. - С. 247-289.

5. Дейч А. М. Методи ідентифікації динамічних об’єктів. – М: Енергія, 1979 – 240 с.

6. Інтернет-ресурс. – Нейромережеве відображення дійсності. – Режим доступу: http://studies.in.ua/mpd_seminar/1313-neyrmerezh.html .

7. Комашинський В. І. Смирнов Д. О. Ведення у нейро-інформаційні технології. / В. І. Комашинский, Д. А. Смирнов - СПб, 1999. – С. 33-48.

8. Николенко С. Базові принципи машинного навчання на прикладі лінійної регрессії. Habr. URL: https://habr.com/ru/company/ods/blog/322076.

9. Соколов В. Ю. Інформаційні системи і технології: Навчальний посібник – Київ ДУІКТ. - 2010. – С. 33 - 49.

10. Растригін Л. А., Ейдук Я. Ю. Адаптивнні методи багатокритеріальної оптимизації // Автоматика і телемеханіка. 1985. - № 1. - С. 5-26.

11. Річардсон Я. Відеокодування. H.264 и MPEG-4 – стандарти нового покоління. М.: Техносфера, 2005. - 368 с.

12. Тимчук М. І. Особливості використання Oracle GoldenGate для розробки відмовостійкої архітектури баз даних. Інформаційні моделі, системи та технології:

Праці VIII наук.-техн. конф. (Тернопіль, 09-10 грудня 2020 р.) Тернопіль, 2020. – С. 121.

13. Федько В. В. Організація баз даних та знань : навч.-прак. посібн. / В. В. Федько, О. В. Тарасов, М. Ю. Лосєв. – Х.: Вид. ХНЕУ, 2013. – 200 с.

14. Устінова Г. М. Інформаційні системи менеджмента. / Навч. посіб. – СПб.: «ДіаСофтЮП», 2000. – 204 с.

15. Уосермен Ф. Нейрокомп'ютерна техніка: Теорія і практика. Переклад українською І. Ю. Юрчак, 2001 р. – С. 3-15.

16. Шапіро Л., Стокман Д. Комп'ютерний зір // М.: Біном. Лабораторія знань. - 2006. - Т. 752. – С. 66-69.

17. Augspurger T. Dask for Machine Learning. Dask. URL: https://examples.dask.org/machine–learning.html.

18. Brownlee J. Why Use Ensemble Learning?. Machine Learning Mastery. URL: https://machinelearningmastery.com/why–use–ensemble–learning.

19. Dask is a flexible library for parallel computing in Python. Dask. URL: https://docs.dask.org/en/latest.

20. Daubechies I. The wavelet transform, time-frequency localization and signal analysis / I. Daubechies // IEEE Transactions Information Theory. - 1990. - Vol. 36. - P. 961–1005.

21. Image and Video Coding – Emerging Standards and Beyond. // IEEE Transactions on Circuits and Systems for Video Technology. 1997. V. 8. № 7. P. 814–837.

22. Hank Tullis, Paul Elbow, Chaitanya R. Geddam. Beginning Oracle GoldenGate. Apress. 2015. - 400 p.

23. Vettigli G. Cambridgespark. From simple regression to multiple regression with decision trees. URL: https://info.cambridgespark.com/latest/from–simple–regression–to–multiple–regression–with–decision–trees.

24. Kurzweil R. 15 Big Data Problems You Need to Solve. SolveXia. URL: https://www.solvexia.com/blog/15–big–data–problems–you–need–to–solve.

25. Lees K. Image compression using wavelets / K. Lees // 1999. ISIE’ 99. − 2002. – P. 3-12.

26. Loeffler C., Ligtenberg A., Moschytz G. Practical Fast 1-D DCT Algorithms with 11 Multiplications. Proc. Int’l. Conf. on Acoustics, Speech, and Signal Processing 1989 (ICASSP ‘89). – 991p.

27. Machhi C. PySpark – SparkContext. tutorialspoint. URL: https://www.tutorialspoint.com/pyspark/pyspark_sparkcontext.htm

28. Mallat S. Understanding image transform codes / S. Mallat, F. Falzon // Proc. SPIE Aerospace Conf. – 1997. – P. 56-62.

29. Salamon J. A Dataset and Taxonomy for Urban Sound Research / J. Salamon, C. Jacoby, J. Bello. // 22nd ACM International Conference on Multimedia,Orlando USA. – 2014 – P.13-19.

30. Singh A. Ultimate guide to handle Big Datasets for Machine Learning using Dask (in Python). Analytics Vidhya. URL: https://www.analyticsvidhya.com /blog/2018/08/dask–big–datasets– machine_learning–python.

31. Sen J. A Robust Mechanism for Defending Distributed Denial OF Service Attacks on Web Servers / J. Sen // International Journal of Network Security & Its Applications (IJNSA). - 2011, March. - Vol. 3, N 2. - P. 162–179.

32. RIEDEL M. Big Data – Definition, Importance, Examples & Tools. Research Data Alliance. URL: https://www.rd–alliance.org/group/big–data–ig–data–development–ig/wiki/big–data–definition–importance–examples–tools.

33. Tsvetovat M. top 5 problems with big data – and how to solve them. Piesync. URL: https://www.piesync.com/blog/top–5–problems–with–big–data–and–how–to–solve–them/

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО -НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

"Розробка сервісу для прогнозування на великих наборах даних методами машинного навчання мовою Python"

Виконав студент 4 курсу
групи ПД - 44
ПоночовнийОлександр Вікторович
Керівникроботи
к.ф.-м.н., доц., доценткафедриІПЗ СадовенкоВолодимир Сергійович

Київ – 2023

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи –** прогнозування майбутніх тенденцій, розвитку або подій на основі великих наборів даних.
- **Об'єктом дослідження** є прогнозування на великих наборах даних методами машинного навчання.
- **Предметом дослідження** є методи та алгоритми машинного навчання для прогнозування на великих наборах даних.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. розглянути теоретичні аспекти застосування методів обробки великих даних;
2. охарактеризувати сучасні стандарти та технології Big Data;
3. визначити особливості розробки сервісу для прогнозування на великих наборах даних в умовах невизначеності предметної галузі та характеру змінних;
4. проаналізувати підходи до створення ефективної системи навчання для прогнозування неперервних чисельних ознак для великого набору даних.
5. розробити сервіс для прогнозування на великих наборах даних в умовах невизначеності предметної галузі та характеру змінних методом Dask.

3

АНАЛІЗ АНАЛОГІВ

	MS Excel	Amazon Forecast	Predict Salary Analysis
Точність прогнозів на великих наборах даних	-	+	+
Моделі прогнозування	Стандартні моделі	Великий вибір моделей	Декілька навчених моделей
Доступність	Безкоштовно	Платна	Безкоштовно
Продуктивність на великих наборах даних	Обмежена	Не обмежена	Обмежена

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

1. Можливість завантажувати великі набори даних і проводити їх попередню обробку, таку як очищення, нормалізацію та видалення відсутніх значень.
2. Можливість вибору та навчання різних моделей машинного навчання на основі вхідних даних.
3. Використання навчених моделей для прогнозування на нових наборах даних.

Нефункціональні вимоги

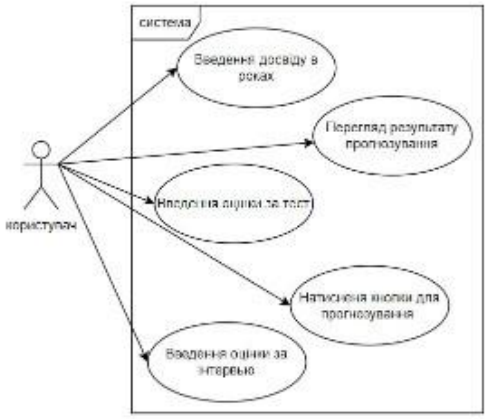
1. Час відповіді на запит має бути мінімальним, а прогнозування займає прийнятий час.
2. Забезпечення високого рівня безпеки для захисту конфіденційності та цільності даних.
3. Комфортний та зрозумілий інтерфейс користувача.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

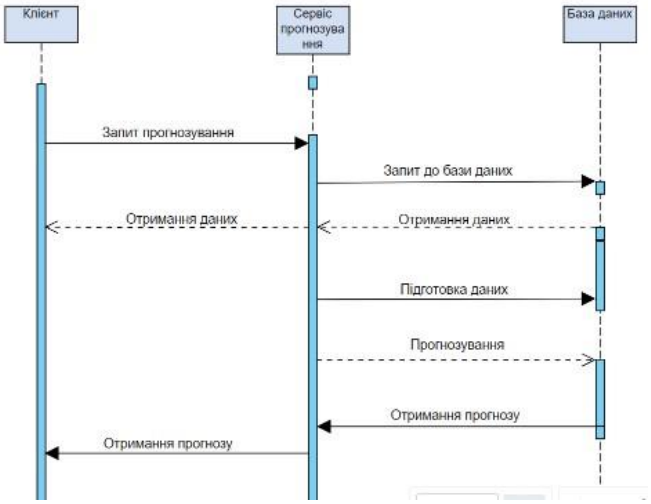
6

Діаграма варіантів використання



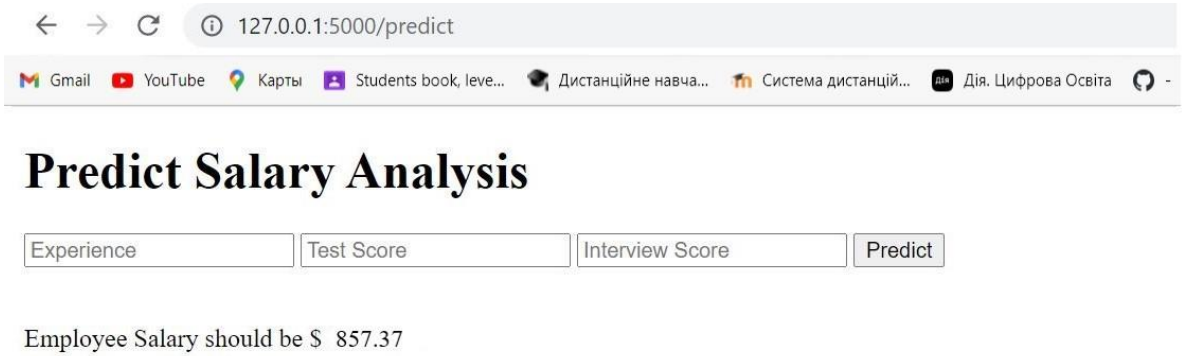
7

Діаграма послідовності



8

Головний екран сервісу



9

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Поночовний О.В. Використання алгоритму Supervised Learning для прогнозування на великих наборах даних / Поночовний О.В., В.С. Садовенко // Всеукраїнська науково -технічна конференція «Застосування програмного забезпечення в ІКТ» Збірник тез. 20.04.2023 ДУТ, м. Київ — К.: ДУТ, 2023. — С. 110.

10

ВИСНОВКИ

1. Розглянуто теоретичні аспекти застосування методів обробки великих даних.
2. Охарактеризовано сучасні стандарти та технології Big Data.
3. Визначено особливості розробки сервісу для прогнозування на великих наборах даних в умовах невизначеності предметної галузі та характери змінних.
4. Проаналізовано підходи до створення ефективної системи навчання для прогнозування неперервних чисельних ознак для великого набору даних.
5. Розроблено сервіс для прогнозування на великих наборах даних в умовах невизначеності предметної галузі та характери змінних методом Dask.

Додаток Б

Лістинг коду

only_function.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from new_data.pars_data import pars, pars_target
from new_data.preprocessing import drop_feature, drop_tails, mean, scale
from new_data.train_data import model_train
from new_data.test_data import model_test
pd.set_option('display.width', 400)
pd.set_option('display.max_columns', 10)
# variables for function
filesdir = ('D:\learning\predict_value\\new_data\Data_Andrew_181_Files\\')
NUMBER_OF_LETTERS = –12
filenames, fn, target = pars(filesdir, 'D:\learning\predict_value\\new_data\Andrew_181_Files – копия.xlsx')
data_features, data_target = pars_target(fn, NUMBER_OF_LETTERS, filesdir, target)
data_features = drop_feature(data_features)
data_features = drop_tails(data_features)
data_mean = mean(data_features, data_target)
data_mean = scale(data_mean)
print(data_mean)
# split data train – test
train, test = train_test_split(data_mean, test_size=0.2, random_state=47)
model_train(train.drop(['target1','target2'],1),train['target1'], 'target1')
model_train(train.drop(['target1','target2'],1),train['target2'], 'target2')
# save result only fit (target – 1)in file redicted_target1.txt
f_target1 = open('predicted_target1.txt', 'w')
result = model_test(test.drop(['target1','target2'],1), 'target1')
for i in range(len(result[0])):
f_target1.write(result[0][i]+":\n")
for value in result[1][i]:
f_target1.write(str(value)+"\n")
f_target1.write("\n")
f_target1.close()
# save result only fit (target – 2) in file redicted_target2.txt
f_target2 = open('predicted_target2.txt', 'w')
result = model_test(test.drop(['target1','target2'],1), 'target2')
for i in range(len(result[0])):
f_target2.write(result[0][i]+":\n")
for value in result[1][i]:
f_target2.write(str(value)+"\n")
```

```
f_target2.write("\n")
pars_data.py
import pandas as pd
import os
import dask.dataframe as dd
pd.set_option('display.width', 400)
pd.set_option('display.max_columns', 10)
def pars(files, file_target):
"""
:param files: les that are in the folder
:param file_target: this file which contains targets
:return: returns everything that was in the directory
"""
NUMBER_OF_LETTERS = –12
target = None
filenames = []
for (_, _, fn) in os.walk(files):
for file in fn:
filenames.append(file[:NUMBER_OF_LETTERS])
target = pd.read_excel(file_target, index_col=False)
return filenames, fn, target
# Preprocessing:
files_1 = ('D:\learning\predict_value\new_data')
def dataframe_in_dict(fn, NUMBER_OF_LETTERS, root):
"""
:param fn: file name
:param NUMBER_OF_LETTERS:quantity letters in dataset
:param root:root directory
:return: dictionary with dataset
"""
dict_data = {}
for f in fn:
if f[:NUMBER_OF_LETTERS] not in dict_data:
dict_data.update({f[:NUMBER_OF_LETTERS]: [dd.read_csv(root + f)]})
else:
dict_data[f[:NUMBER_OF_LETTERS]].append(dd.read_csv(root + f))
return dict_data
def pars_target(fn, NUMBER_OF_LETTERS, root, target):
data_features = []
data_target = []
proc = dataframe_in_dict(fn, NUMBER_OF_LETTERS, root)
for key, val in proc.items():
data_features.append(pd.concat(val, ignore_index=True))
for targetIndex in range(target.shape[0]):
if key[9:] == target.loc[targetIndex, 'ID'][9:]:
data_target.append(target.loc[targetIndex])
data_target = pd.concat(data_target, 1).T
return data_features, data_target
```

train.py

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from dask.distributed import Client, progress
from dask_ml.model_selection import train_test_split
import dask.dataframe as dd
from sklearn.linear_model import LinearRegression
import pickle
import os
def model_train(x,y, filename2):
#Plain random forest>
model_rf1 = RandomForestRegressor()
with joblib.parallel_backend('dask'):
model_rf1.fit(train.drop(['target1','target2'],1),train['target1'])
Lin_1 = LinearRegression()
with joblib.parallel_backend('dask'):
Lin_1.fit(train.drop(['target1','target2'],1),train['target1'])
#Plain DecisionTreeRegressor
Dt_1 = DecisionTreeRegressor (max_depth=10, min_samples_leaf=30,random_state=42)
with joblib.parallel_backend('dask'):
Dt_1.fit(train.drop(['target1','target2'],1),train['target1'])
# save result method in file
filename = '/'.join([models_dir,'random_forest.xls'])
pickle.dump(model_rf1, open(filename, 'wb'))
filename = '/'.join([models_dir,' LinearRegression.xls'])
pickle.dump( Lin_1, open(filename, 'wb'))
filename = '/'.join([models_dir,'decision_tree.xls'])
pickle.dump(Dt_1, open(filename, 'wb'))
```

test.py

```
import os
import pickle
def model_test(x, filename2):
"""
target = []
modelnames = []
for root, d, filenames in os.walk(filename2+'_models'):
for f in filenames:
loaded_model = pickle.load(open('/'.join([root,f]), 'rb'))
modelnames.append(f[:–4])
target.append(loaded_model.predict(x))
return modelnames, target
```