МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра прикладної математики та моделювання складних систем

Допущено до захисту

Завідувач кафедри ПМ та МСС

_____ Коплик І.В.
(підпис)

«__» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «магістр»

спеціальність 113 «Прикладна математика»

освітньо–професійна програма «Наука про дані та моделювання складних систем»

тема роботи **«МОДЕЛЮВАННЯ ТА ПРОГНОЗУВАННЯ ВЕЛИКИХ НАБОРІВ ДАНИХ ЗАСОБАМИ МАШИННОГО НАВЧАННЯ»**

Виконавець

студент факультету ЕлІТ

Кіншаков Е.В _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник

доцент, к.е.н.
(науковий ступінь, вчене звання)

Маринич Т.О. _____
(прізвище, ім'я, по батькові) `(підпис)

Суми – 2020

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет **електроніки та інформаційних технологій**

Кафедра **прикладної математики та моделювання складних систем**

Рівень вищої освіти другий

(перший (бакалавр) або другий (магістр)

Галузь знань **11 Математика та статистика**

Спеціальність **113 Прикладна математика**

Освітня програма **освітньо–професійна «**Наука про дані та моделювання складних систем**»**

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМтаМСС
Коплик І.В.
«___» _____20__р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Кіншакова Едуарда Віталійовича.
(прізвище, ім'я, по батькові)

1. **Тема роботи** Моделювання та прогнозування великих наборів даних засобами машинного навчання

Керівник роботи доцент, к.е.н. Маринич Т.О
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)

затверджено наказом по факультету ЕлІТ від «__» ___20__р. № _____

2. **Термін подання роботи студентом** «___»_____20__р.

3. **Вихідні данні до роботи** модель машинного навчання для передбачення цільових змінних

4. **Зміст розрахунково–пояснювальної записки (перелік питань, що їх належить розробити)** теоретико – методологічні та прикладні аспекти великих даних, методологічні аспекти аналізу великих наборів даних , практична частина

5. Перелік графічного матеріалу

Рис. 1.1 – Різноманітність даних в області Big Data

Рис. 1.2 – Схема роботи PySpark

Рис. 2.1– Приклад роботи дерева рішень

Рис. 2.2 – Принцип роботи bootstrap

Рис. 2.3 – Приклад генерування нового набору даних

Рис. 2.4 – Результат згенерованого датасету

Рис. 2.5 – Розподіл ознак

Рис. 2.6 – Кореляційна матриця

Рис. 2. 7 – Графіки розподілу пояснювальних змінних

Рис. 2.8 – Аналіз згладжування

Рис. 2.9 – Новий великий датасет

Рис. 2.10 – Генерування великого набору

Рис. 2.11 – Масив Dask

Рис. 2.12 – Фрейм даних dask

Рис. 2.13 – Принцип розпаралелювання

Рис. 2.14 – Масштабування алгоритмів

Рис. 3.1 – Результат підключення до бібліотеки

Рис. 3.2 – Побудова нового набору даних

Рис. 3.3 – Перевірка пропущених значень вибірки

Рис. 3.4 – Нестандартні пропущені значення

Рис. 3.5 – Графік щільності розподілу цільових змінних

Рис. 3.6 – Діаграми розсіювання залежних та пояснювальних змінних

Рис. 3.7 – Гістограми розподілу даних

Рис. 3.8 – Кореляційна карта – матриця

Рис. 3.9 – Дані після видалення не інформативних змінних

Рис. 3.10 – Стандартизовані дані

Рис. 3.14 – Дерево для цільової змінної (target1)

Рис. 3.15 – Дерево для цільової змінної (target2)

Рис. 3.16 – Результати прогнозних та фактичних даних

Рис. 3.17 – Блок схема роботи моделі

6. **Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх**

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Специфікація великих даних	доцент, к.е.н. Маринич Т.О		
Лінійна регресія	доцент, к.е.н. Маринич Т.О		
Дерево рішень	доцент, к.е.н. Маринич Т.О		
Статистичний аналіз великого набору даних	доцент, к.е.н. Маринич Т.О		
Результати моделей машинного навчання	доцент, к.е.н. Маринич Т.О		

7. Дата видачі завдання «__»_____20__р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання роботи	Примітка
1	Специфіка великих даних	02.09.20 – 15.09.20	
2	Інструменти та характеристики Big Data	15.09.20 – 30.09.20	
3	Аналіз існуючих даних	30.09.20 – 09.10.20	
4	Обробка даних	09.10.20 – 20.10.20	
5	Паралелізація за допомогою бібліотеки Dask	20.10.20 – 31.10.20	
6	Побудова моделей	31.10.20 – 19.11.20	
7	Формування результатів	19.11.20 – 15.12.20	

Здобувач вищої освіти _____ Кіншаков Е.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ доцент, к.е.н. Маринич Т.О
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота: 73с., 31 – рисунків, 4 – таблиці, 29 джерел.

Мета роботи: розробка функціональної моделі машинного навчання для прогнозування неперервних чисельних ознак для великого набору даних.

Об'єкт дослідження: великі вибірки даних кількісних та якісних ознак в умовах невизначеності предметної галузі та характеру змінних.

Предмет дослідження: пристосування великих даних для машинної обробки та прогнозування.

Методи навчання: рівняння математичної фізики, інтегральні рівняння, моделювання статистичних методів та алгоритмів машинного навчання для великого набору даних.

Для отримання задовільного результату роботи алгоритмів машинного навчання, був проведений аналіз великих даних. За допомогою бібліотеки dask був проведений статистичний аналіз та побудовані моделі. Бібліотека dask масштабувала всі методи для аналізу та навчання моделі. Швидкість навчання збільшилась, а також швидкість виконування певних статичних функцій. Після навчання моделей, обиралась найкраща, за критеріями оцінки якості моделі, а також за оцінками прогнозної якості.

Ключові слова: DASK, PYTHON, SKLEARN, СТАТИСТИКА, МАШИНЕ НАВЧАННЯ, АНАЛІЗ ДАНИХ, РЕГРЕСІЯ, ВИПАДКОВИЙ ЛІС, ДЕРЕВА РІШЕНЬ, ЦІЛЬОВА ЗМІННА, PANDAS.

ЗМІСТ

ВСТУП 7
1 ТЕОРЕТИКО–МЕТОДОЛОГІЧНІ ТА ПРИКЛАДНІ АСПЕКТИ ВЕЛИКИХ ДАНИХ 8
1.1 Специфіка великих даних 8
1.2 Інструменти та характеристики Big Data 16
2.1 Інструменти машинного навчання для Big Data 20
2.3 Паралелізація методом Dask 41
3 ПРАКТИЧНА ЧАСТИНА 48
3.1 Статистичний аналіз великого набору даних 48
3.2 Результати моделей машинного навчання 57
ПЕРЕЛІК ДЖЕРЕЛ ІНФОРМАЦІЇ 64

ВСТУП

Сьогодні все більше компаній, підприємств та організацій збирають та аналізують великі набори даних для оптимізації бізнес процесів, підвищення ефективності діяльності, залучення нових та утримання старих клієнтів. При цьому часто компанії стикаються з проблемами надійного збереження даних, їх обробки та забезпечення вимог приватності даних.

В роботі розглянуто техніки та прийоми підготовки та аналізу великоформатних вибірок в умовах прихованої інформації про походження даних та відсутності інфраструктурного забезпечення Big Data. У якості емпіричних даних використано проектне завдання та дані із сервісу UpWork, де більшість вибірок даних є анонімізовані через необхідність збереження комерційної таємниці.

Метою роботи є розробка функціональної моделі машинного навчання для прогнозування неперервних чисельних ознак для великого набору даних.

Об’єкт дослідження: Великі вибірки даних кількісних та якісних ознак в умовах невизначеності предметної галузі та характеру змінних.

Предмет дослідження: Пристосування великих даних для машинної обробки та прогнозування.

Практична новизна: Розробка функціональної робочої моделі прогнозування знеособлених великих даних за обмежених технічних ресурсів.

1 ТЕОРЕТИКО–МЕТОДОЛОГІЧНІ ТА ПРИКЛАДНІ АСПЕКТИ ВЕЛИКИХ ДАНИХ

1.1 Специфіка великих даних

Термін «Великі дані» (Big Data) використовується для позначення структурованих і неструктурованих даних великих обсягів різних форматів.

Big data є постійним потоком величезних обсягів інформації, безперервно надходжених з різних джерел. Як відзначається в багатьох публікаціях, до категорії big data відноситься більшість потоків даних понад 100 Гб в день. Великі дані відрізняються за такими параметрами як:

– об`єм. Сам по собі термін Big Data пов'язаний з великим розміром. Розмір даних – найважливіший показник при здобутті можливої цінності. Щодня 6 мільйонів людей використовують цифрові медіа, що за попередніми оцінками генерує 2.5 квінтильйони байт даних.

– різноманітність – наступний аспект. Він посилається на гетерогенні джерела і природу даних, які можуть бути як структурованими, так і неструктурованими. Раніше електронні таблиці і бази даних були єдиними джерелами інформації, розглянутими в більшості додатків. Сьогодні ж дані в формі електронних листів, фото, відео, PDF файлів, аудіо теж розглядаються в аналітичних додатках. Така різноманітність неструктурованих даних призводить до проблем в зберіганні, видобутку і аналізі: 27% компаній не впевнені, що працюють з відповідними даними;

– швидкість генерації. Тобто наскільки швидко дані накопичуються і обробляються для задоволення вимог, визначає потенціал. Швидкість визначає швидкість припливу інформації з джерел – бізнес процесів, додатків, сайтів, соціальних мереж і медіа, сенсорів, мобільних пристроїв. Потік даних величезний і безперервний в часі;

– мінливість описує мінливість даних в деякі моменти часу, яке ускладнює обробку і управління. Так, наприклад, велика частина даних неструктурованих за своєю природою.

На початку 2000 – х років стрімкого розвитку набули спеціальні масштабовані програмні інструменти та підходи для ефективної обробки великих даних, як альтернатива традиційним системам управління базами даних і рішень класу Business Intelligence [1, 2].

Business intelligence (BI) – програмне забезпечення, створене для допомоги менеджеру в аналізі інформації про свою компанію і її оточенні. Більшість інструментів Business intelligence застосовуються кінцевими користувачами для доступу, аналізу і генерації звітів за даними, які найчастіше розташовуються в сховищі, вітринах даних або оперативних складах даних. Бізнес–аналіз як діяльність складається з кількох пов'язаних між собою процесів:

– інтелектуальний аналіз даних;
– аналітичну обробку в реальному часі;
– отримання інформації з баз даних;
– складання звітів .

Розробники додатків використовують BI–платформи для створення і впровадження BI–додатків, які не розглядаються як BI–інструменти. Існують такі інструменти BI як:

1. Tableau – робить аналіз даних інтуїтивно зрозумілим і простим у використанні. За допомогою функції перетягування можна редагувати і створювати свої аналітичні діаграми без глибокого розуміння структур даних. Це дозволяє людям візуалізувати ідеї і ділитися ними зі своїми командами;

2. QlikView – це продукт для самообслуговування, що дозволяє користувачам шукати і досліджувати аналітичні програми. За допомогою

механізму асоціативного індексування даних QlikView користувачі можуть легко генерувати ідеї, комбінуючи різні бази даних в одне натискання;

3. SAP Business Objects – це звітність і аналіз BI. Підключається до фонових ресурсів, які містять безліч додатків для звітів. Це дозволяє користувачам створювати і впроваджувати моделі прогнозної аналітики для генерації інформації і прогнозування тенденцій маркетингу;

4. IBM Cognos Analytics – це аналітична платформа самообслуговування, яка включає розширену аналітику. Інтерактивні інформаційні панелі спрощують аналіз даних. Він також має широкий спектр функцій аналізу, розширений аналіз, аналітичну звітність і аналіз тенденцій. Крім того, Cognos Analytics дозволяє користувачам взаємодіяти з інформацією зі своїх мобільних пристроїв;

5. WebFOCUS – це генератор інформації. Це дозволяє користувачам створювати панель моніторингу на основі даних, зібраних в різних форматах. Він також дозволяє користувачам переглядати дані, отримувати корисну інформацію, створювати звіти і обмінюватися результатами з співавторами.

Аналіз великих даних дозволяє приймати більш ефективні рішення та розробляти зважене стратегічне планування. В наші дні компанії все частіше використовують великі дані, щоб перевершити своїх конкурентів. У більшості галузей як існуючі конкуренти, так і нові учасники будуть використовувати стратегії, отримані на основі проаналізованих даних, для конкуренції, впровадження інновацій та отримання прибутку.

Великі дані допомагають організаціям створювати нові можливості для зростання і зовсім нові категорії компаній, які можуть об'єднувати і аналізувати галузеві дані. Ці компанії мають у своєму розпорядженні великий обсяг інформації про продукти і послуги, покупців і постачальників, перевагах споживачів, які можна фіксувати і аналізувати.

Організації збирають дані з різних джерел, включаючи бізнес–транзакції, соціальні мережі та інформацію з датчиків або обмін даних між

машинами. Раніше його зберігання було проблемою, але нові технології (наприклад, Hadoop) полегшили цей тягар.

Hadoop є проектом верхнього рівня організації Apache Software Foundation, тому основним дистрибутивом і центральним репозиторієм для всіх напрацювань вважається саме Apache Hadoop.

Якщо налаштовувати інфраструктуру для Big Data проектів (з нуля), взявши за основу класичний дистрибутив проекту Hadoop, розгорнути екосистему для великих даних буде досить трудомістким і тривалим процесом, з яким впорається не кожен системний адміністратор. Як правило, щоб скоротити час розгортання і складність адміністрування, використовують готові рішення на основі Hadoop: Cloudera, Hortonworks, MapReduce або HDInsight [3].

Ці продукти вже містять в собі не тільки 4 основних модуля hadoop(HDFS, MapReduce, Yarn і Hadoop Common) від Apache Software Foundation. Також в них присутні додаткові інструменти для обробки різноформатних даних та інтелектуального аналізу інформації (Data Mining), в тому числі з використанням машинного навчання [4]:

– рішення для управління потоками даних (Flume, Sqoop);

– фреймворки для розподіленої і потокової обробки, а також брокери повідомлень (Spark, Storm, Kafka);

– нереляційні СУБД і SQL – движки для Big Data аналітики (HBase, Hive, Impala, Shark, Drill);

– координатори і планувальники завдань (Zookeeper, Hue, Oozie, Azbakan);

– засоби Machine Learning (Mahout, Spark MLlib).

У готових рішеннях всі ці інструменти вже інтегровані між собою і супроводжуються повним пакетом програмної документації, що полегшує процеси адміністрування та підтримки інфраструктури для великих даних.

Але необхідно підкреслити, що за ці зручні фреймворки необхідно добре заплатити.

Дані надходять у всіх типах форматів від структурованих наборів даних, числових даних в традиційних базах даних до неструктурованих текстових документів, електронної пошти, відео, аудіо, даних біржових котирувань і фінансових транзакцій. Раніше електронні таблиці і бази даних були єдиними джерелами даних, які розглядалися більшістю додатків. У наші дні в додатках для аналізу також враховуються дані у вигляді електронних листів, фотографій, відео, пристроїв моніторингу, PDF – файлів, аудіо. Така різноманітність неструктурованих даних створює певні проблеми для зберігання, видобутку і аналізу даних. Різноманітність даних зображено на рис.1.2 [5].

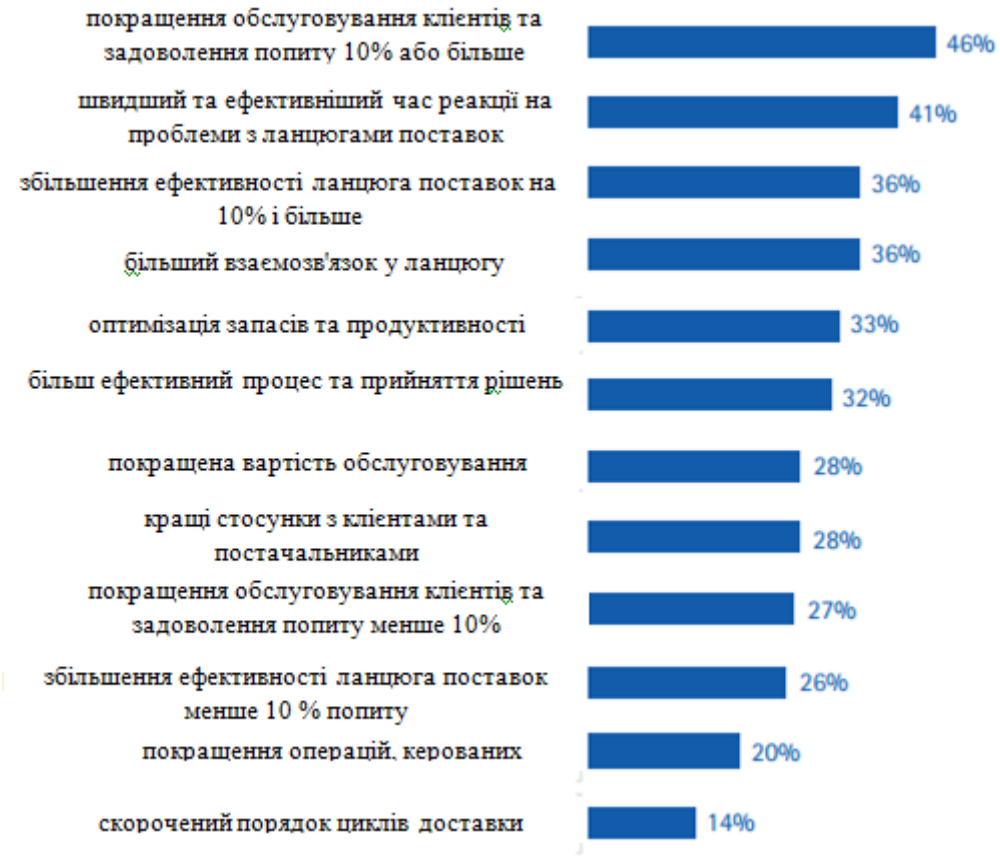

Рис. 1.1 – Різноманітність даних в області Big Data[5].

Важливість великих даних залежить не від того, скільки даних у компанії, а від того, як компанія використовує зібрані дані. Кожна компанія використовує дані по– своєму, чим ефективніше компанія використовує свої дані, тим більше у неї можливостей для зростання. Перевагами збору та аналізу великих даних є:

– економія витрат: деякі інструменти великих даних, такі як Hadoop, можуть принести бізнесу економічну вигоду, коли необхідно зберігати великі обсяги даних, і ці інструменти також допомагають у визначенні більш ефективних способів ведення бізнесу

– скорочення часу: висока швидкість таких інструментів, як Hadoop і аналітика в пам'яті, дозволяє легко визначати нові джерела даних, що допомагає підприємствам негайно аналізувати дані і приймати швидкі рішення на основі отриманих знань;

– розуміння ринкових умов: аналізуючи великі дані, можна краще зрозуміти поточні ринкові умови. Наприклад, прослідкувавши купівельну поведінку клієнтів, компанія може визначити продукти, які продаються найбільше, і виробляти продукти відповідно до цієї тенденцією. Тим самим він може випередити своїх конкурентів;

– контроль репутації в Інтернеті: інструменти для роботи з великими даними можуть виконувати аналіз настроїв. Таким чином, можна отримати зворотній зв'язокпро те, хто що говорить про вашу компанію[5].

Великі дані створюють характерні особливості, які не розділяються традиційними наборами даних. Ці особливості створюють значні проблеми для аналізу даних і мотивують розробку нових статистичних методів. На відміну від традиційних наборів даних, де обсяг вибірки, як правило, більше, ніж вимір, великі дані характеризуються величезним розміром вибірки і високою розмірністю. Тому існує значний ряд проблем роботи з великим набором даних:

1) знаходження сигналу в шумі: після того, як ми проаналізували дані, іноді нам доводиться повертатися і говорити, що ми просто не виміряли це правильно чи виміряли неправильні змінні, тому що тут ми нічого не можемо виявити. Таким чином, одна з найбільших проблем, з якими стикаються підприємства при роботі з великими даними, – це класична проблема голки в стозі сіна. Далі необхідно зазначити, що в необробленому вигляді великі дані виглядають як клубок вовни, і необхідний науковий підхід до даних;

2) розрізнені сховища даних: це причина, по якій вам потрібно обробляти цифри, щоб складати щомісячний звіт про продажі. Вони є причиною того, що рішення вищого керівництва приймаються повільними темпами. Вони – причина того, що відділи продажів і маркетингу просто не ладнають. Вони є причиною того, що ваші клієнти шукають можливості для свого бізнесу в іншому місці, тому що вони не відчувають, що їхні потреби задовольняються, а менша, гнучкіша компанія пропонує щось краще;

3) неточні дані: збірники даних не тільки неефективні на операційному рівні, вони також є родючим грунтом для найбільшої проблеми з даними: неточних даних;

4) технології розвиваються занадто швидко: більш великі корпорації з більшою ймовірністю стануть жертвами розрізнених даних з тих причин, що вони вважають за краще зберігати свої бази даних локально, а також тому, що прийняття рішень про нові технології часто відбувається повільно;

5) відсутність кваліфікованих робітників: звіт CapGemini показав, що 37% компаній не можуть знайти кваліфікованих аналітиків, які могли б використовувати їх дані. Найкраще для них сформувати одну загальну команду з аналізу даних для компанії, або перепрофілювавши ваших нинішніх співробітників, або найнявши нових співробітників, що спеціалізуються на великих даних [6] ;

6) проблема безпеки: важливість безпеки даних не може залишитися непоміченою. Однак у міру впровадження рішень не завжди легко зосередитися на безпеці даних за допомогою безлічі рухомих частин. Дані також необхідно правильно зберігати, що починається з шифрування і постійного резервного копіювання;

7) доступність: Іноді компанії передають дані одній людині або одному відділу. Це не тільки покладає величезну відповідальність на деяких обраних, але також створює недолік доступності для всієї організації в відділах, де дані можуть бути корисні для надання позитивного впливу. Розрізнені сховища даних в першу чергу безпосередньо обмежують переваги збору даних;

8) висока вартість інформаційних рішень: зрозумівши, які переваги принесе вашому бізнесу впровадження рішень для обробки даних, можна виявити, що покупка і обслуговування необхідних компонентів може бути дорогим. Поряд з обладнанням, таким як сервери і сховище для програмного забезпечення, також потрібні людські ресурси і час;

9) нерозуміння: компанії можуть використовувати дані для підвищення продуктивності в багатьох областях. Деякі з кращих варіантів використання даних: скорочення витрат, створення інновацій, запуск нових продуктів, збільшення прибутку і підвищення ефективності, і це лише деякі з них. Незважаючи на переваги, компанії не поспішали впроваджувати технології обробки даних або розробляли план створення культури, орієнтованої на дані [7].

1.2 Інструменти та характеристики Big Data

Дані безглузді, поки не перетворюються в корисну інформацію і знання, які можуть допомогти керівництву в прийнятті рішень. Для цього є ряд програмного забезпечення для роботи з великими даними, доступних на ринку. Це програмне забезпечення допомагає зберігати, аналізувати, складати звіти і багато іншого[8]. 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 – один з потужних інструментів аналізу великих даних з відкритим вихідним кодом. Він пропонує більше 80 операторів високого рівня, які спрощують створення паралельних програм. Це один з інструментів аналізу даних з відкритим вихідним кодом, який використовується в самих різних організаціях для обробки великих наборів даних.

Особливості Apache Spark:

– Він допомагає запускати додаток в кластері Hadoop до 100 разів швидше в пам'яті і в десять разів швидше на диску;

– це один з інструментів аналізу даних з відкритим вихідним кодом, який пропонує швидку обробку;

– підтримка складної аналітики;

– можливість інтеграції з Hadoop і існуючими даними Hadoop;

– це один з інструментів аналізу великих даних з відкритим вихідним кодом, який надає вбудовані API– інтерфейси на Java, Scala або Python [8].

Elasticsearch – це система пошуку і аналізу великих даних на основі JSON. Це розподілена система пошуку і аналітики RESTful для вирішення безлічі варіантів використання. Це один з інструментів аналізу великих

даних, який пропонує горизонтальну масштабованість, максимальну надійність і просте управління.

Особливості Elasticsearch:

– Він дозволяє комбінувати багато типів пошуку, такі як структурований, неструктурований, географічний, метричний.

– інтуїтивно зрозумілі API для моніторингу та управління забезпечують повну видимість і контроль;

– він використовує стандартні API RESTful і JSON. Він також створює і підтримує клієнтів на багатьох мовах, таких як Java, Python, NET і Groovy;

– функції пошуку та аналітики в реальному часі для роботи з великими даними за допомогою Elasticsearch – Hadoop;

– він розширює можливості функцій безпеки, моніторингу, звітності та машинного навчання [8].

Hive – великі дані з відкритим вихідним кодом. Це дозволяє програмістам аналізувати великі набори даних на Hadoop. Це допомагає швидко запитувати і управляти великими наборами даних.

Особливості:

– підтримує SQL–подібна мова запитів для взаємодії і моделювання даних;

– він компілює мову з двома основними картками завдань і редуктором;

– це дозволяє визначати ці завдання з використанням Java або Python;

– hive призначений для управління і запиту тільки структурованих даних;

– мова Hive, заснований на SQL, відокремлює користувача від складності програмування Map Reduce;

– він пропонує інтерфейс Java Database Connectivity (JDBC).

Нещодавно я знайшов час і відшукав кращий інструмент, який змусив мене перетворити процес обробки даних. Я використовую цей інструмент для складної обробки – наприклад, для читання декількох файлів з 10 гігабайтами даних, застосування до них фільтрів і агрегування.

Dask – надає розширений паралелізм для аналітики, забезпечуючи діяльність на рівні інструментів, які часто використовують для аналізу великого набору. Dask включає в себе numpy, pandas і sklearn. Це бібліотека з відкритим вихідним кодом, що знаходиться у вільному доступі. У ній використовуються існуючі API – інтерфейси Python і структури даних, щоб спростити перемикання між еквівалентами на основі Dask.

Особливості Dask :

– реалізований та підтримується тільки python;

– масштабований під фреймворк Sklearn;

– паралелізує процеси обробки та побудови моделі машинного навчання;

– dask використовує pandas API.

Всі перелічені інструменти для великих даних реалізуються лише мовою python. Це дуже важлива складова, оскільки не витрачається час на вивчення та розуміння іншої мови, також не витрачається час на інсталяції та налагодження середовища для великих даних.

Технологія hive гарний інструмент, який допомагає реалізувати аналіз даних за допомогою мови python, але працює по більшій частині з базами даних SQL, оскільки цей фреймворк на цій мові і заснований. Тому даний інструмент не буде ефективним для даної задачі.

Apache Spark чудовий інструмент для обробки та роботи з великим потоком або кластерами даних, з великим потоком даних. Доволі популярний та багатофункціональний, але він орієнтований на більш потужне апаратне забезпечення, що є проблемою в вирішенні даної задачі.

На рис. 1.2 зображений вигляд роботи фреймворку PySpark. Насамперед необхідно підкреслити, що технологія використовується дуже активно в залежності від поставленої задачі.

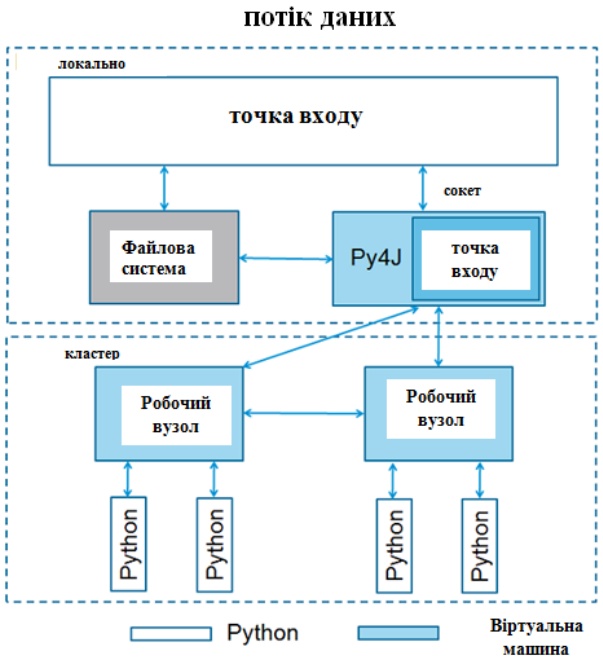

Рис. 1.2 – Схема роботи PySpark [9]

Оскільки задача була поставлена лише в передбачені цільових змінних, і не ґрунтувалась на великій обробці першочергово, таким чином було прийняте рішення засобами Data Science досягнути певних результатів. Обробка та аналіз цих даних будуть наведені у наступних частинах пояснювальної записки.

Для даної задачі буде доречний фреймворк Dask. Він включає в себе такі інструменти аналізу та побудови алгоритмів, як numpy, pandas і sklearn. Необхідно лише встановити декілька бібліотек для функціонування фреймворку. Паралелізація допоможе нам зменшити час на обробку та аналіз. Також буде легко зробити функціональну модель для замовника, ніж це робити іншими інструментами великих даних.

2 МЕТОДОЛОГІЧНІ АСПЕКТИ АНАЛІЗУ ВЕЛИКИХ НАБОРІВ ДАНИХ

2.1 Інструменти машинного навчання для Big Data

Оскільки існуючий інструментарій обробки та роботи з великими даними використовуватись не будуть, тому що не вистачає потужностей апаратної частини і це не розглядається з точки зору замовника. Для того щоб зробити зручну та функціональну модель, розглядатись будуть принципи роботи Data Science. Головною проблематикою є вирішення задачі великих наборів за допомогою класичного машинного навчання та статистичного аналізу.

Машинне навчання полягає в добуванні знань з даних. Це наукова область, що знаходиться на перетині статистики, штучного інтелекту та комп'ютерних наук і також відома як прогнозна аналітика або статистичне навчання. В останні роки застосування методів машинного навчання в повсякденному житті стало повсякденним явищем. Сучасні веб – сайти і пристрої використовують алгоритми машинного навчання, починаючи з автоматичних рекомендацій по перегляду фільмів, замовлення їжі або покупки продуктів, і закінчуючи персоналізованими онлайн– радіотрансляції і розпізнаванням друзів на фотографіях. Коли бачимо складний сайт типу Facebook, Amazon або Netflix, то вельми ймовірно, що кожен розділ сайту містить кілька моделей машинного навчання[10].

Найбільш успішні алгоритми машинного навчання – це ті, які автоматизують процеси прийняття рішень шляхом узагальнення відомих прикладів. У цих методах, відомих як навчання з учителем або контрольоване навчання (Supervised Learning), користувач надає алгоритму пари об'єкт– відповідь, а алгоритм знаходить спосіб отримання відповіді по об'єкту. Зокрема, алгоритм здатний видати відповідь для об'єкта, якого він

ніколи не бачив раніше, без будь – якої допомоги людини. Якщо повернутися до прикладу класифікації спаму з використанням машинного навчання, користувач пред'являє алгоритму велика кількість листів (об'єкти) разом з інформацією про те, чи є лист спамом чи ні (відповіді). для нового електронного листа алгоритм визначить ймовірність, з якою це лист можна віднести до спаму[10].

Алгоритми навчання без вчителя або неконтрольованого навчання (Unsupervised Algorithms) – це ще один вид алгоритмів машинного навчання.. В алгоритмах навчання без учителя відомі тільки об'єкти, а відповідей немає. Хоча є багато успішних сфер застосування цих методів, їх, як правило, важче інтерпретувати і оцінити.

Вирішуючи завдання машинного навчання з учителем і без, важливо представити вхідні дані в форматі, зрозумілі комп'ютеру. Часто дані представляють у вигляді таблиці. Кожна точка даних, яку необхідно дослідити (кожен електронний лист, кожен клієнт, кожна транзакція) є рядком, а кожна властивість, яка описує цю точку даних (скажімо, вік клієнта, сума або місце здійснення транзакції), є стовпцем. Можна описати користувачів за віком, статтю, датою створення облікового запису і частоті покупок в інтернет–магазині.

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

2.1.1 Лінійна регресія

Лінійна регресія – це керований алгоритм навчання, який використовується, коли цільова або залежна змінна продовжує дійсне число. Він встановлює взаємозв'язок між залежною змінною y та однією або декількома незалежними змінними x, використовуючи лінію, що найкраще

підходить. Це працює за принципом звичайної найменшої квадратичної (OLS) – середньоквадратичної помилки (MSE). У статистиці цей метод оцінює параметри лінійної регресійної функції, мінімізуючи суму квадратної різниці між спостережуваною залежною змінною у наборі даних та прогнозованою за допомогою функціонування лінійної регресії [11].

Представлення гіпотези.Ми будемо використовувати x_i для позначення незалежної змінної та y_i для позначення залежної змінної. Пару (x_i , y_i) називають навчальним прикладом. Ознака i в позначенні просто вказується на навчальному наборі. Таким чином маємо навчання m, тоді $i = 1,2,3, \dots m$.

Метою контрольованого навчання є вивчення функції гіпотези h для навчального набору, який можна використовувати для оцінки y на основі x. Отже, функціонування гіпотези представлена у вигляді:

$$h_\theta(x_i) = \theta_0 + \theta_1 x_i, \tag{2.1}$$

де θ_0, θ_1– параметр гіпотези, це рівняння для простої одновимірної лінійної регресії.

Для множинної лінійної регресії більше одного незалежного виходу змінної, тоді необхідновикористовувати x_{ij} для позначення незалежної змінної та y_i для позначення залежної змінної. Маємо n незалежних змінних, тоді $j = 1,2,3, \dots n$. Функція гіпотези представлена у вигляді [11].

$$y \approx h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots\dots\dots + \theta_j x_j \tag{2.2}$$

$\theta = \{\theta_0, \theta_1, \dots . \theta_j \dots . \theta_n\}$ є параметром гіпотези, m кількість тренувальних прикладів, n кількість незалежних змінних. Регресори $x = \{x_1, x_2, \dots . x_n\}$.

Середня квадратична помилка (MSE): середня квадратична помилка є найбільш поширеною функцією втрат. Щоб розрахувати MSE, треба взяти різницю між передбаченими значеннями і істинними, звести її в квадрат і усереднити по всьому набору даних [12]. Реалізація даної моделі буде здійснюватись за допомогою імпортування бібліотеки [13].

$$MSE(X, h_\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 \quad (2.3)$$

де $y^{(i)}$ – фактичний очікуваний результат, а $x^{(i)}$ – прогноз моделі.

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i| \quad (2.4)$$

де α – гіперпараметри регресії

2.1.2 Дерево рішень

Дерево рішень – це інструмент для прийняття рішень, який використовує деревоподібну структуру у вигляді блок – схеми або є модель рішень і всіх їх можливих результатів, включаючи результати, витрати на введення і корисність.

Алгоритм дерева рішень відноситься до категорії алгоритмів навчання з учителем. Він працює як для безперервних, так і для категоріальних вихідних змінних.

Регресія дерева рішень спостерігає за особливостями об'єкта і навчає модель в структурі дерева передбачати дані в майбутньому для отримання значимого безперервного висновку [14]. Безперервний висновок означає, що результат не є дискретним, тобто він не представлений тільки дискретним відомим набором чисел або значень. На рис.2.1. зображено дерево регресійне дерево рішень на існуючих даних. [15].

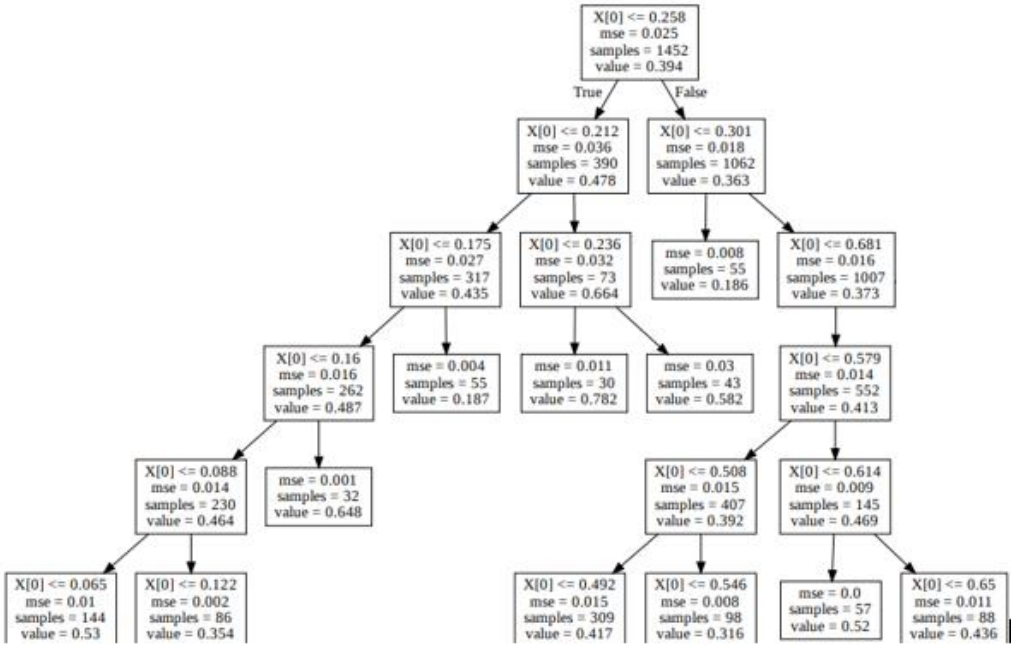

Рис. 2.1– Приклад роботи дерева рішень

Побудова дерева здійснюється в 4 етапи:

- вибрати атрибут для здійснення розбиття в вузлі;
- визначити критерій зупинки навчання;
- вибрати метод відсікання гілок;
- оцінити точність побудованого дерева.

Розбиття атрибутів має здійснюватися за певним правилом, для якого і вибирають атрибут. Причому обраний атрибут повинен розбити безліч спостережень в вузлі так, щоб результуючі підмножини містили приклади з однаковими мітками класу або були максимально наближені до цього. Іншими словами – кількість об'єктів з інших класів в кожному з цих множин має бути якомога менше. Критеріїв існує багато, але найбільшою популярністю користуються теоретико–інформаційний та статистичний.

В основі теоретико–інформаційного критерію лежить (ентропія Шеннона) [16]:

$$S = -\sum_{i=1}^{N} p_i log_2 p_i \quad (2.5)$$

де ρ_{i} – ймовірність знаходження системи в і – ому стані. Це дуже важливе поняття, яке використовується в фізиці, теорії інформації та інших областях. Зазначимо, що, інтуїтивно, ентропія відповідає ступеню хаосу в системі. Чим вище ентропія, тим менше впорядкована система і навпаки [16].

Ентропія розглядається як міра неоднорідності підмножини за представленими в ньому класів. І навіть якщо класи представлені в рівних частках, а невизначеність класифікації найбільша, то ентропія теж максимальна. Логарифм від одиниці буде звертати ентропію в нуль, якщо всі приклади вузла належать до одного класу.

Якщо обраний атрибут розбиття забезпечує максимальне зниження ентропії результуючої підмножини відносно батьківського, його можна вважати найкращим.

Оскільки ентропія – по суті ступінь хаосу (або невизначеності) в системі, зменшення ентропії називають приростом інформації. Формально приріст інформації (information gain, IG) при розбитті вибірки за ознакою визначається як [17]:

$$IG(Q) = S_o - \sum_{i=1}^{q} \frac{N_i}{N} S_i \quad (2.6)$$

де q – число груп після розбиття, N_i– число елементів вибірки, у яких ознака Q має i – е значення [18].

Задача вибору атрибута в такій ситуації полягає в максимізації величини $IG(Q)$, яку називають приростом інформації. Тому теоретико–інформаційний підхід також відомий під назвою «критерій приросту інформації».

Якщо говорити про статистичний підхід, то в основі цього методу лежить використання індексу Джині. Він показує, як часто випадково

обраний приклад навчальної вибірки буде розпізнано неправильно. Важлива умова – цільові значення повинні братися з певного статистичного розподілу. Якщо говорити простіше, то індекс Джині показує відстань між розподілами цільових значень і передбаченнями моделі. Мінімальне значення показника говорить про гарну роботу моделі.

Індекс Джині розраховується за формулою[18]:

$$Gini(G) = 1 - \sum_{i=1}^{n} p_i^2 \tag{2.7}$$

де Q – результуюче безліч, n – число класів в ньому, pi – ймовірність i–го класу (виражена як відносна частота прикладів відповідного класу).

Значення показника змінюється від 0 до 1. Якщо індекс дорівнює 0, значить, всі приклади результуючої безлічі відносяться до одного класу. Якщо дорівнює 1, значить, класи представлені в рівних пропорціях і різновірогідні. Оптимальним вважають те розбиття, для якого значення індексу Джині мінімально.

Критерій якості при прогнозуванні кількісної ознаки:

$$D = \frac{1}{\ell}\sum_{i=1}^{\ell}(y_i - \frac{1}{\ell}\sum_{i=1}^{\ell} y_i)^2 \tag{2.7}$$

В основі популярних алгоритмів побудови дерева рішень, існує принцип жадібної [18] максимізації приросту інформації на кожному кроці обирається та ознака, при поділі за яким приріст інформації виявляється найбільшим. Далі процедура повторюється рекурсивно, поки ентропія не опиниться рівною нулю або деякій малій величині (якщо дерево не підходить ідеально під навчальну вибірку щоб уникнути перенавчання) у різних алгоритмах застосовуються різні евристики для "ранньої зупинки" або "відсікання", щоб уникнути побудови перенавчання [19].

Переваги та недоліки дерева рішень.

Переваги:

– формують чіткі і зрозумілі правила класифікації та регресії;

– здатні генерувати правила в областях, де фахівцеві важко формалізувати свої знання;

– легко візуалізуються, тобто можуть «інтерпретуватися» не тільки як модель в цілому, але і як прогноз для окремого тестового суб'єкта (шлях в дереві);

– швидко навчаються і прогнозують;

– не потрібно багато параметрів моделі;

– підтримують як числові, так і категоріальні ознаки.

Недоліки:

– дерева рішень чутливі до шумів у вхідних даних. Невеликі зміни навчальної вибірки можуть привести до глобальних коректувань моделі, що позначиться на зміні правил класифікації і інтерпретується моделі.

– розділяюча межа має певні обмеження, через що дерево рішень за якістю класифікації поступається іншим методам.

– можливе перенавчання дерева рішень, через що доводиться вдаватися до методу «відсікання гілок», установці мінімального числа елементів в листі дерева або максимальної глибини дерева.

– складний пошук оптимального дерева рішень: це призводить до необхідності використання евристики типу жодного пошуку ознаки з максимальним приростом інформації, які в кінцевому підсумку не дають 100–відсоткової гарантії знаходження оптимального дерева.

– дерево рішень робить константний прогноз для об'єктів, що перебувають у просторі ознак поза паралелепіпеда, який охоплює не всі об'єкти навчальної вибірки. Реалізація методу, буде здійснюватися за допомогою імпорту бібліотеки [20].

2.1.3 Ансамблеві методи

Ансамблі (ensembles) – це методи, які поєднують в собі безліч моделей машинного навчання, щоб у підсумку отримати більш потужну модель. Існує багато моделей машинного навчання, які належать до цієї категорії, але є дві ансамблевих моделі, які довели свою ефективність на самих різних наборах даних для задач класифікації і регресії, обидві використовують дерева рішень в якості будівельних блоків: випадковий ліс дерев рішень і градієнтний бустінг дерев рішень.

Метод bootstrap [21] полягає в наступному. Нехай є вибірка X розміру N. Рівномірно вибрати з вибірки N об'єктів з поверненням. Це означає, що ми будемо N раз обирати довільний об'єкт вибірки (вважаємо, що кожен об'єкт «дістається» з однаковою ймовірністю $\frac{1}{N}$), причому кожен раз обраємо з усіх вихідних N об'єктів. Відзначимо, що через повернення серед них виявляться повтори. Позначимо нову вибірку через X_1. Повторюючи процедуру M раз, згенеруємо M підвибірок $X_{1,..,}X_m$. Тепер маємо досить велику кількість вибірок і можемо оцінювати різні статистики вихідного розподілу рис.2.2 [21].

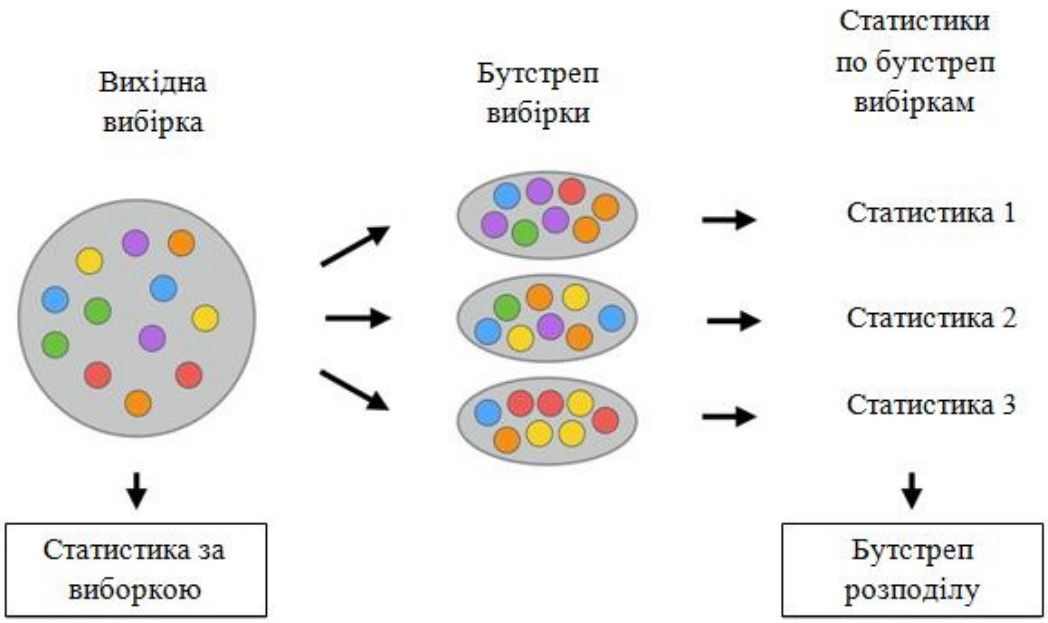

Рис. 2.2 – Принцип роботи bootstrap [20]

Беггінг (bagging) [22] . Нехай є навчальна вибірка X. За допомогою бутстрепа згенеруємо з неї вибірки $X_{1,..,}X_m$. Тепер на кожній вибірці навчимо свій класифікатор $a_i(x)$. Підсумковий класифікатор буде усереднювати відповіді всіх цих алгоритмів (в разі класифікації це відповідає голосуванню):

$$a(x) = \frac{1}{M}\sum_{i=1}^{M} a_i(x) \qquad (2.8)$$

Випадковий ліс – це набір дерев рішень, де кожне дерево трохи відрізняється від інших. Ідея випадкового лісу полягає в тому, що кожне дерево може досить добре прогнозувати. Випадковий ліс є хорошим сімейством базових класифікаторів для беггінга (bagging) [22] , оскільки вони досить складні і можуть досягати нульової помилки на будь– якій вибірці.

Ансамбль моделей, що використовують метод випадкового підпростору, можна побудувати, використовуючи наступний алгоритм:

1. Нехай кількість об'єктів для навчання рівне *N*, а кількість ознак *D*.

2. Виберіть *L* як число окремих моделей в ансамблі.

3. Для кожної окремої моделі *l* обрати *dl(dl<D)* як число ознак для *l*. Зазвичай для всіх моделей використовується тільки одне значення *dl*.

4. Для кожної окремої моделі *l* створіть навчальну вибірку, вибравши *dl* ознак з *D*, і навчіть модель [22].

5. Тепер, щоб застосувати модель ансамблю до нового об'єкту, об'єднавши результати окремих *L* моделей мажоритарних голосувань або шляхом комбінування апостеріорної вірогідності.

Алгоритм побудови випадкового лісу, що складається з *N* дерев, виглядає наступним чином:

1. Для кожного *n = 1,....N;*

2. Згенерувати вибірку X_n за допомогою бутстрепа;

3. побудувати вирішальне деревоb_n по вибірці X_n:

- по заданому критерію ми вибираємо кращий ознака, робимо розбиття в дереві по ньому і так до вичерпання вибірки;

- дерево будується, поки в кожному аркуші не більше n_{min} об'єктів або поки не досягнемо певної висоти дерева;

- при кожному розбитті спочатку вибирається *m* випадкових ознак з *n* вихідних, с і оптимальний розподіл вибірки шукається тільки серед них.

Підсумковий класифікатор:

$$a(x) = \frac{1}{N}\sum_{i=1}^{N} b_i(x) \qquad (2.9)$$

простими словами – для завдання класифікації обираємо рішення голосуванням за більшістю, а в задачі регресії – середнім.

Рекомендується в задачах класифікації брати $m = \sqrt{n}$, а в задачах регресії – $m = \frac{n}{3}$, , де n – число ознак. Також рекомендується в задачах

класифікації будувати кожне дерево до тих пір, поки в кожному листі не виявиться по одному об'єкту, а в задачах регресії – поки в кожному листі не виявиться по п'ять об'єктів [22].

Таким чином, випадковий ліс – це беггінг над вирішальними деревами, при навчанні яких для кожного розбиття ознаки вибираються з деякої випадкової підмножини ознак.

Переваги випадкового лісу:

– має високу точність передбачення, на більшості завдань буде краще лінійних алгоритмів; точність порівнянна з точністю бустінга;

– практично не чутливий до викидів в даних через випадкове семпліювання;

– не чутливий до масштабування (і взагалі до будь– яких монотонним перетворенням) значень ознак, пов'язане з вибором випадкових підпросторів;

– не вимагає ретельної настройки параметрів, добре працює «з коробки». За допомогою «тюнінгу» параметрів можна досягти приросту від 0.5 до 3% точності в залежності від завдання і даних;

– здатний ефективно обробляти дані з великим числом ознак і класів;

– однаково добре обробляє як безперервні, так і дискретні ознаки;

– рідко перенавчати, на практиці додавання дерев майже завжди тільки покращує композицію, але на валідації, після досягнення певної кількості дерев, крива навчання виходить на асимптоту;

– для випадкового лісу існують методи оцінювання значущості окремих ознак в моделі;

– обчислює близькість між парами об'єктів, які можуть використовуватися при кластеризації, виявлення викидів або (шляхом масштабування) дають цікаві уявлення даних [23];

– можливості, описані вище, можуть бути розширені до нерозмічену даних, що призводить до можливості робити кластеризацію і візуалізацію даних, виявляти викиди.

Недоліки випадкового лісу:

– на відміну від одного дерева, результати випадкового лісу складніше інтерпретувати

– немає формальних висновків (p – values), доступних для оцінки важливості змінних

– алгоритм працює гірше багатьох лінійних методів, коли у вибірці дуже багато розріджених ознак (тексти, Bag of words)

– випадковий ліс не вміє екстраполювати, на відміну від тієї ж лінійної регресії (але це можна вважати і плюсом, тому що не буде екстремальних значень в разі потрапляння викиду)

– для даних, що включають категоріальні змінні з різною кількістю рівнів, випадкові лісу упереджені на користь ознак з великою кількістю рівнів: коли у ознаки багато рівнів, дерево буде сильніше підлаштовуватися саме під ці ознаки, так як на них можна отримати більш високе значення оптимізується функціоналу (типу приросту інформації)

– якщо дані містять групи корельованих ознак, що мають схожу значимість для міток, то перевага віддається невеликим групам перед великими [24]. Реалізація методу, буде здійснюватися за допомогою імпорту бібліотеки [25].

2.2 Статистичні відомості існуючих даних

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 рис. 2.3. Таким чином було отримано новий датасет з розмірністю 182 стрічки та 16 колонок. Результат датасету зображено на рис. 2.4.

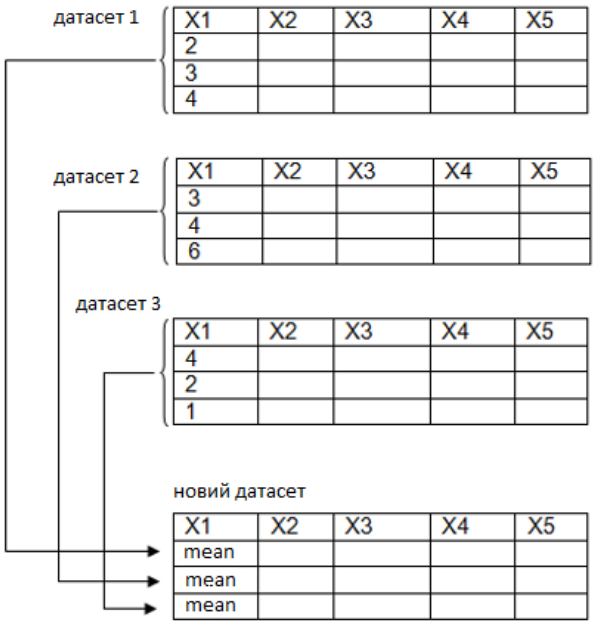

Рис. 2.3 – Приклад генерування нового набору даних

	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)

Рис. 2.4 – 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. 2.6. 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. 2.5.

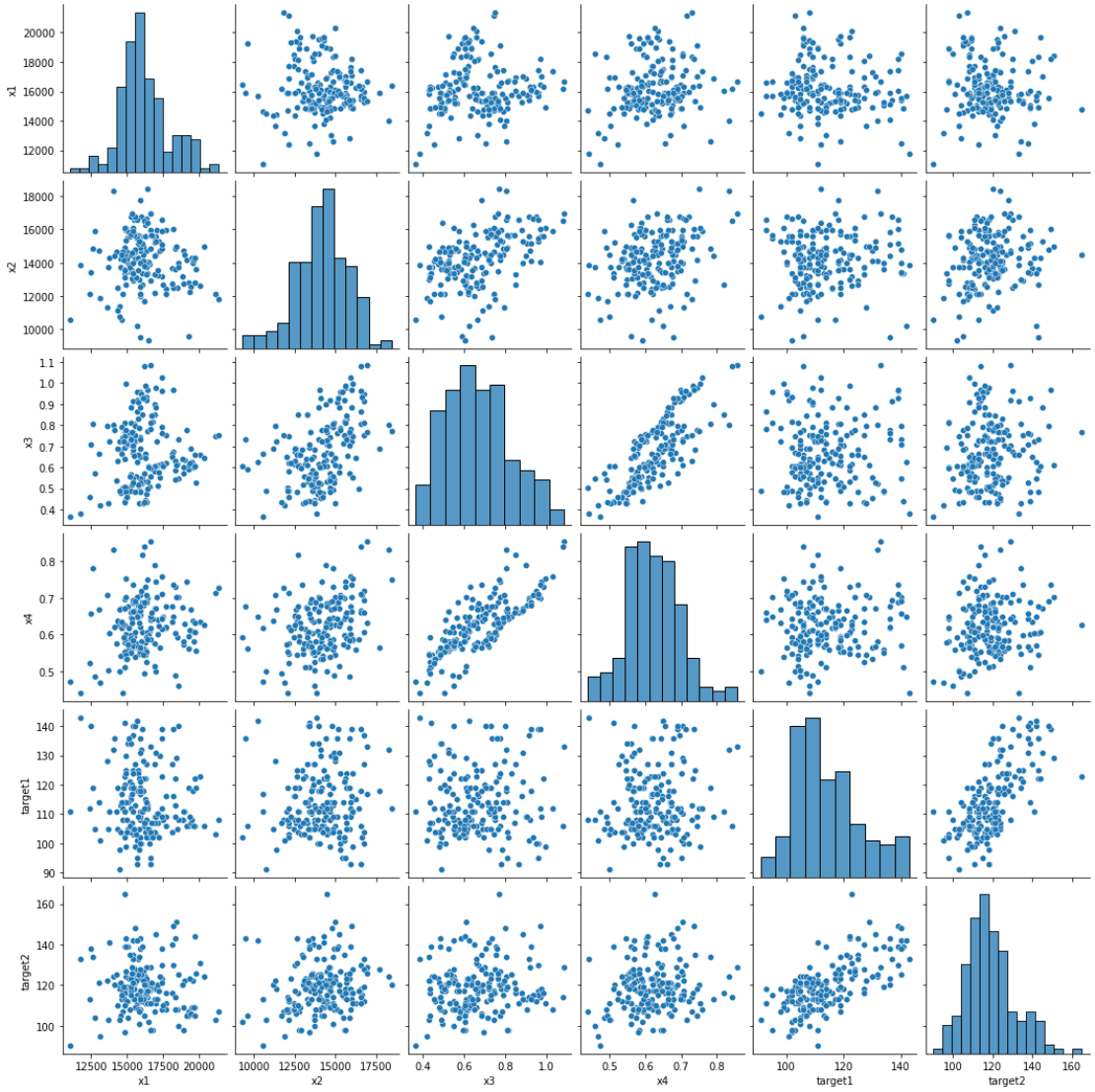

Рис. 2.5 – Розподіл ознак

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.2.6.

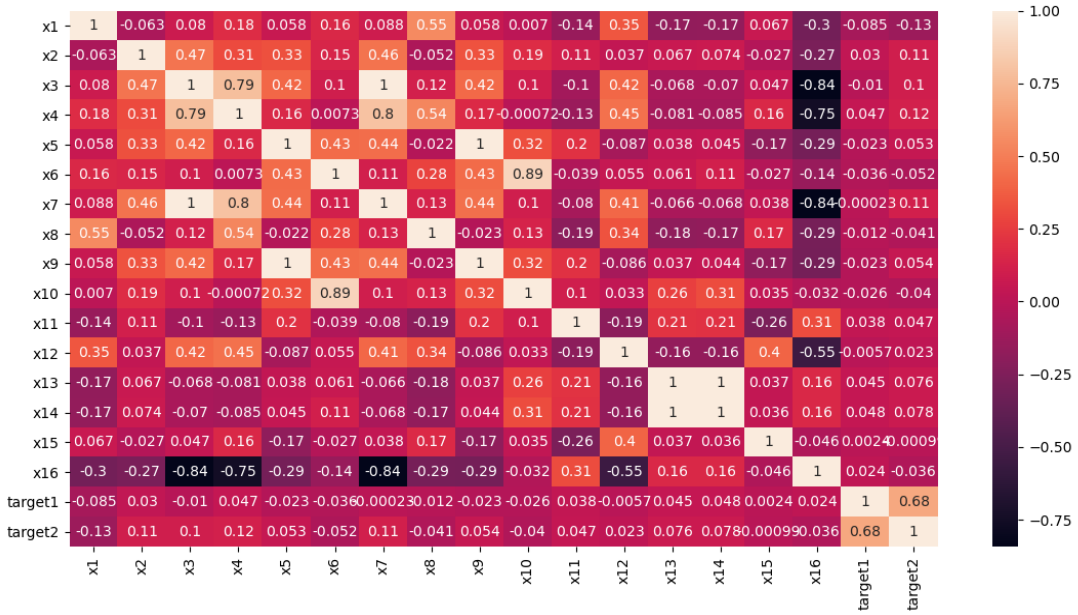

Рис. 2.6 – Кореляційна матриця

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. 2.7.

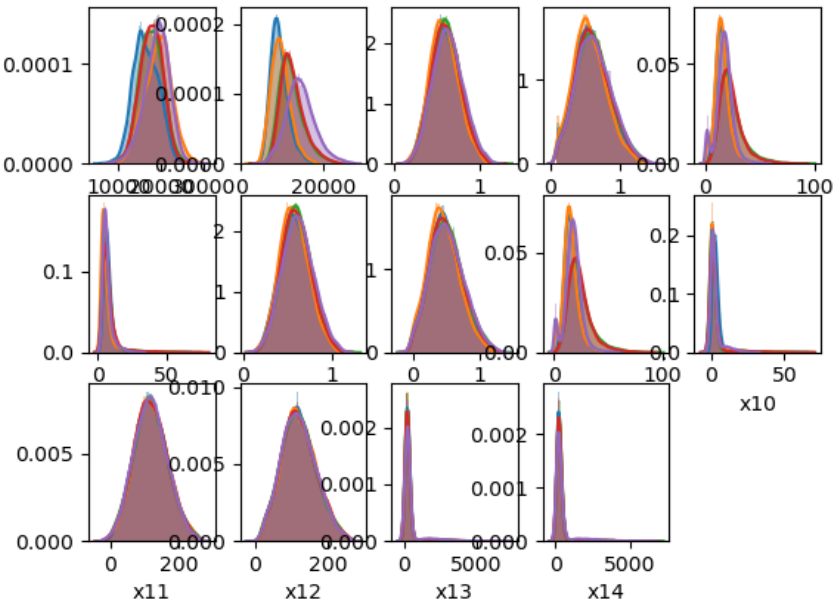

Рис. 2. 7 – Графіки розподілу пояснювальних змінних

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. 2.8.

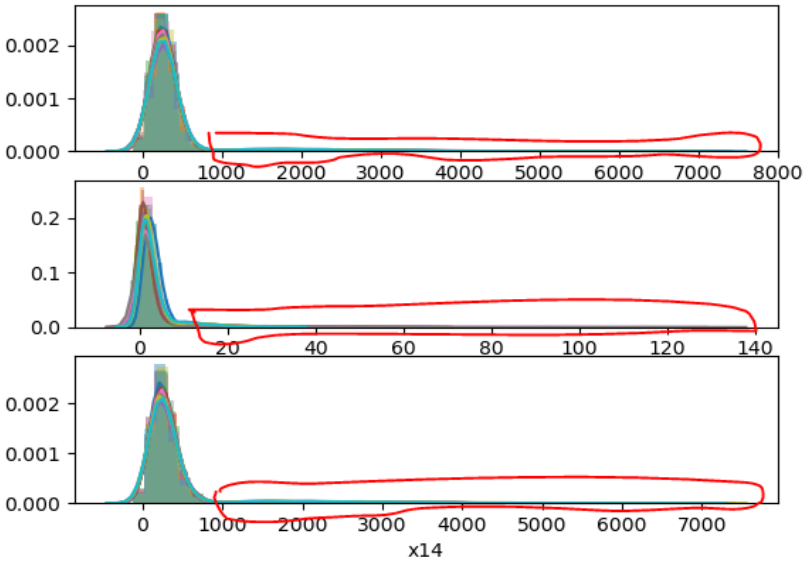

Рис. 2.8 – Аналіз згладжування

Результати роботи алгоритмів на тренувальних та тестових даних зображено в таблиця. 2.1. В цьому випадку були використані 2 алгоритма машиного навчання і статистичний метод як, лінійна регресія, випадковий ліс, дерево рішень. Дані результати були отримані за допомогою коефіцієнту детермінації. Найкраща можлива оцінка – 1.0, і вона може бути негативною (оскільки модель може бути довільно гіршою) [15]. Метрики прогнозної якості як:

– коефіцієнт

$$R^2 = 1 - \frac{u}{v} = (y - y')^2 \qquad (2.10)$$

де u– залишкова сума квадратів

v– загальна сума квадратів

Таблиця 2.1

Результати методів на тренувальних та тестових даних

Метод	Target1(train)	Target2(train)
лінійна регресія	0.2026429564680046	0.1794397919947847
випадковий ліс	0.8803117563095045	0.8909629386996096
дерево рішень	0.8773405067875056	0.8806227130503516
регуляризація Лассо	0.35646734637617206	0.3756643474661721
	Target1(test)	Target2(test)
лінійна регресія	–0.1226429564680046	–0.0794397919947847
випадковий ліс	–0.9833322166197314	–0.6137050129557848
дерево рішень	0.27346376466712197	0.18381144990546183
регуляризація Лассо	0.44927274484	0.5962660452977463

Необхідно звернути увагу на результати. Як уже згадувалося раніше, ми маємо досить хороший результат на вибірці тренувань. Тобто модель добре вчиться. Але ми бачимо поганий навіть негативний результат на тесті. Це так звана перенавчання (overfitting). Як висновок ми маємо дуже потужну модель, агресивні показники lenin rates, засновану на цьому, і поганий тест. Потрібно більше даних. Наступним кроком є дослідження та створення нових функцій. А саме збагачення дата сету даними.

Таким чином 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.2.9.

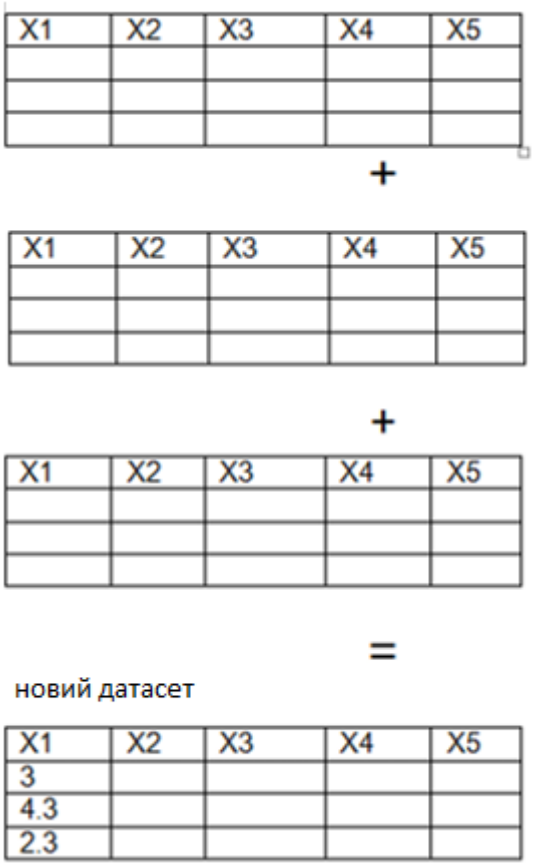

Рис. 2.9 – Новий великий датасет

Результат великого датасету зображено на рис. 2.10. Також слід зазначити його розмір (1815696 – строк, 18 – стовбців).

	0	1	2	3	4	5	6	7
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

8 rows × 1815696 columns

Рис. 2.10 – 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. Проблематика буде висвітлена у дипломному проекті.

2.3 Паралелізація 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 [26] і Numpy [27] – відмінні бібліотеки, але вони не завжди ефективні з обчислювальної точки зору, особливо коли є гігабайти даних, якими потрібно маніпулювати.

Dask – це бібліотека Python, яка може обробляти великі набори даних на одному процесорі з використанням декількох ядер машин або кластера машин (розподілені обчислення).

Dask працює з фреймами даних Pandas і структурами даних Numpy, щоб допомогти виконувати обробку даних і побудову моделей з використанням великих наборів даних на не дуже потужних машинах.

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.2.11 наведено зображення, яке демонструє як виглядає масив.

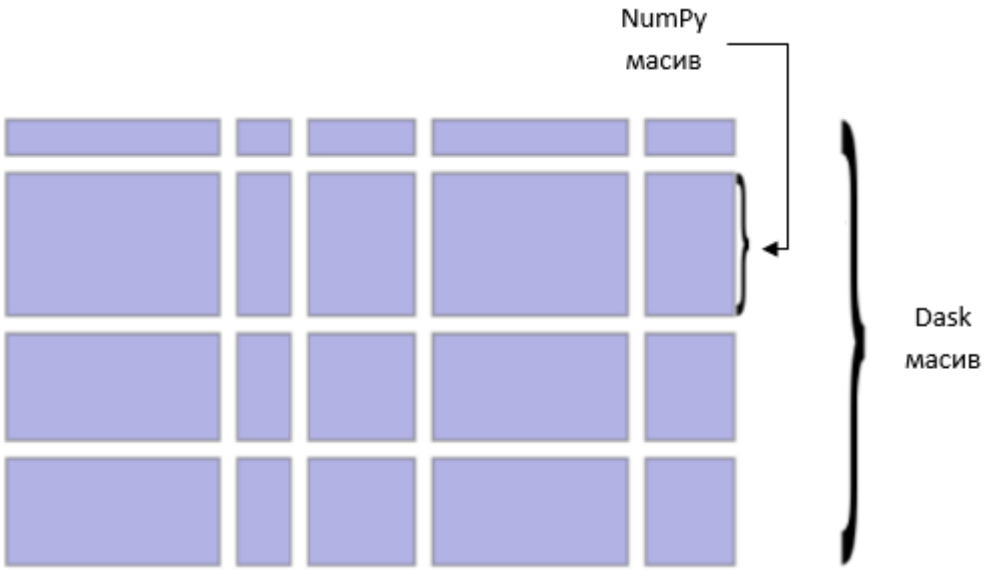

Рис. 2.11 – Масив Dask

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.2.12 зображено, як представляє структуру фрейму даних dask [28].

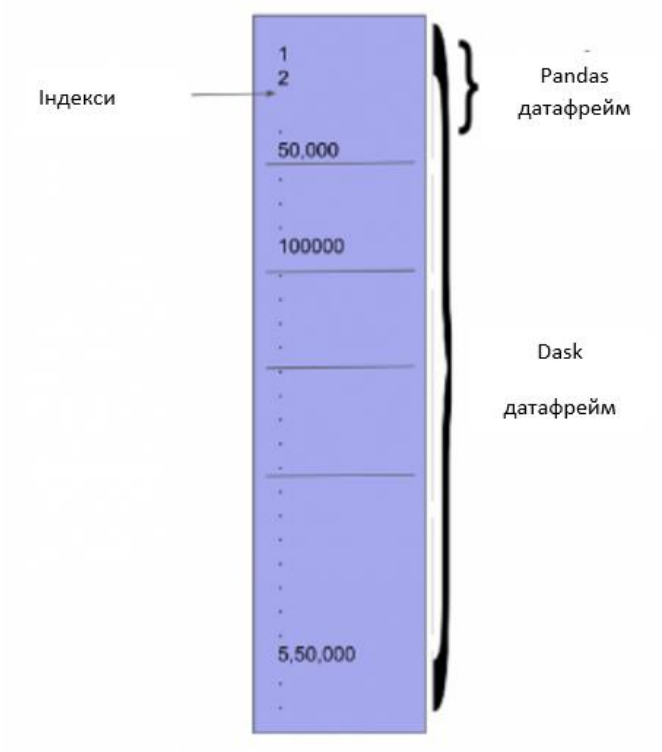

Рис. 2.12 – Фрейм даних dask

В таблиці 2.2 зображені порівняльний час відкриття даних використовуючи pandas та dask . Розмір даних (1815696 – строк, 18 – стовбців).

Таблиця 2.2

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.2.13 [29].

Рис. 2.13 – Принцип розпаралелювання

Незважаючи на те, що паралельні обчислення можна виконувати за допомогою scikit – learn, його не можна масштабувати на кілька машин. З іншого боку, dask добре працює на одній машині, а також може масштабуватися до кластера машин. Принцип масштабування зображений на рис.2.14 [29].

Рис. 2.14 – Масштабування алгоритмів

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. Не існує жорсткого правила, яке говорить, що слід використовувати dask або spark, таким чином побудована порівняльна характеристика між цими технологіями в таблиці 2.3. Яка ще раз аргументує вибір паралелізації dask [29].

Таблиця 2.3

Порівняльна характеристика Dask та Spark

Spark	Dask
spark написаний на scala (scala – мова програмування)	dask написаний мовою програмування python
забезпечує підтримку для R та python	підтримується лише python
spark має свою екосистему	dask є складовою екосистеми python
spark має свою API	dask повторно використовує pandas API
легше зрозуміти та застосувати для користувачів, знайомих з scala або sql.	практикуючий в машинному навчанні
Spark не підтримує багатовимірні масиви	повністю підтримує модель numpy для багатовимірних масивів, котрі піддаються обробці

Головною перевагою є те, що не потрібно вивчати досконало новий інструмент, щоб будувати свої моделі, коли зустрічаєшся з більшими наборами даних. Найкраще в dask полягає в тому, що він пропонує інтерфейс, дуже схожий на pandas, і є дуже невелика (іноді незначуща) різниця в коді.

3 ПРАКТИЧНА ЧАСТИНА

3.1 Статистичний аналіз великого набору даних

У минулих розділах були продемонстровані результати та графіки звичайного набору даних, набір даних був зібраний згідно технічного завдання. Результати методів машинного навчання не задовольнили потреб замовника. Отже 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 3.1. Дані зображені на рис. 3.2.

```
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

Рис. 3.1 – 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

Рис. 3.2 – 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.3.3.

```
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%
```

Рис. 3.3 – Перевірка пропущених значень вибірки

Далі переглянемо на наявність текстових значень. Результат виконання зображено на рис. 3.4.

```
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
```

Рис. 3.4 – 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 рис. 3.5.

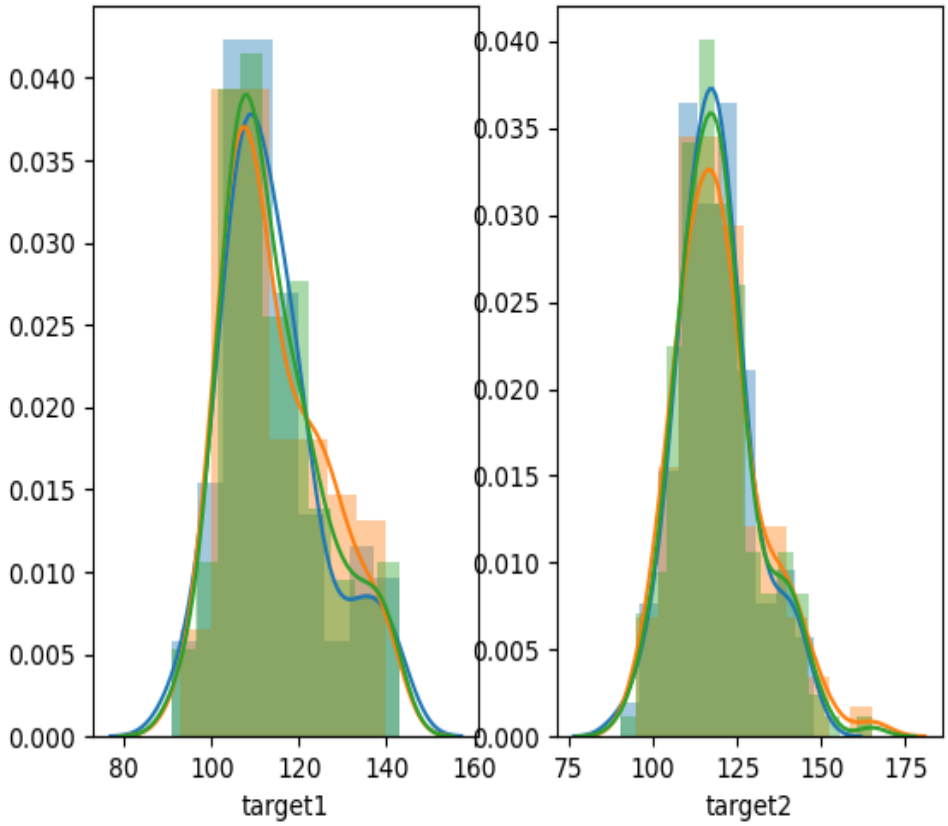

Рис. 3.5 – Графік щільності розподілу цільових змінних

Графік розсіювання котрий зображений на рис.3.6, 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.

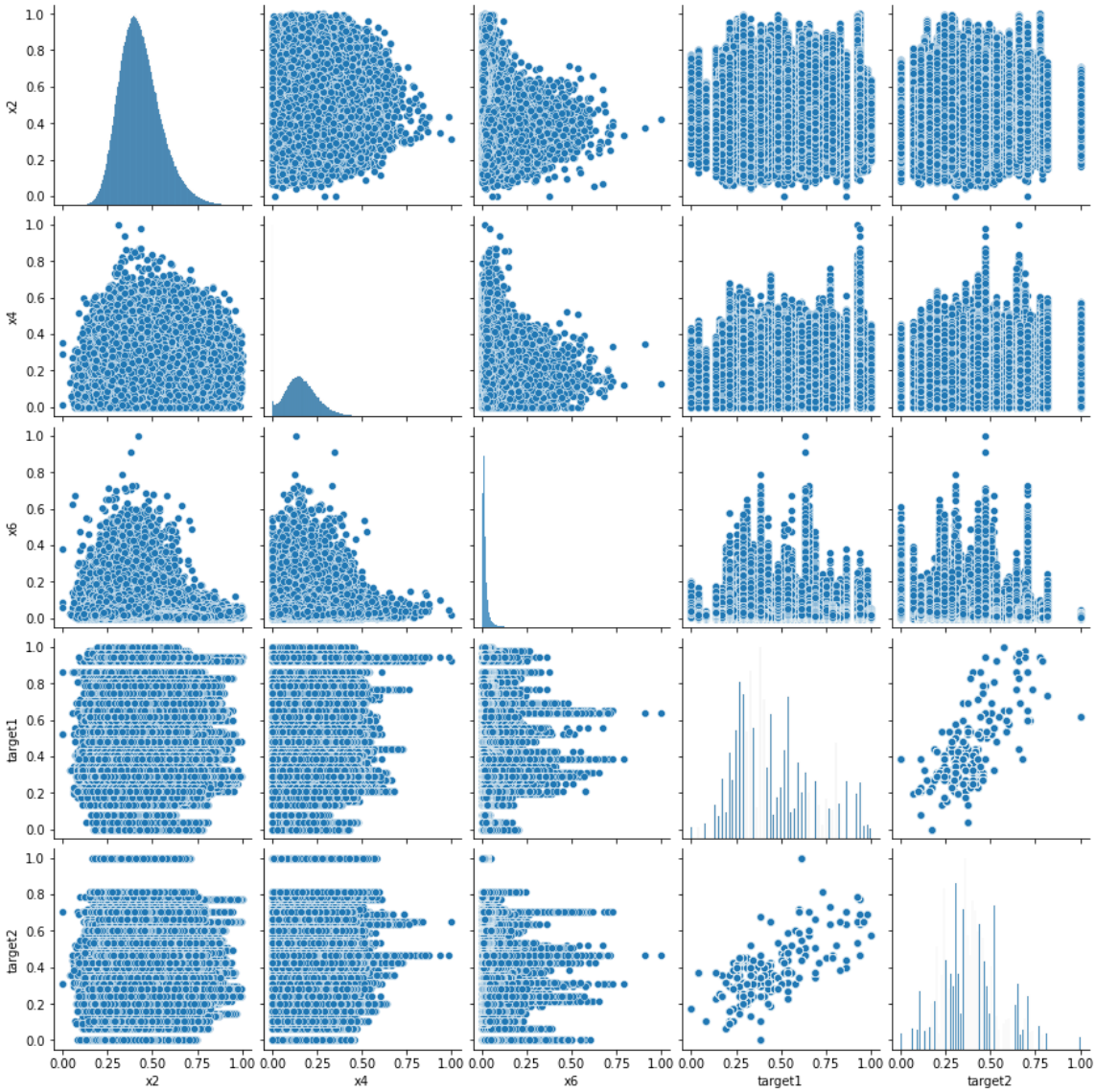

Рис. 3.6 – Діаграми розсіювання залежних та пояснювальних змінних

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 3.7.

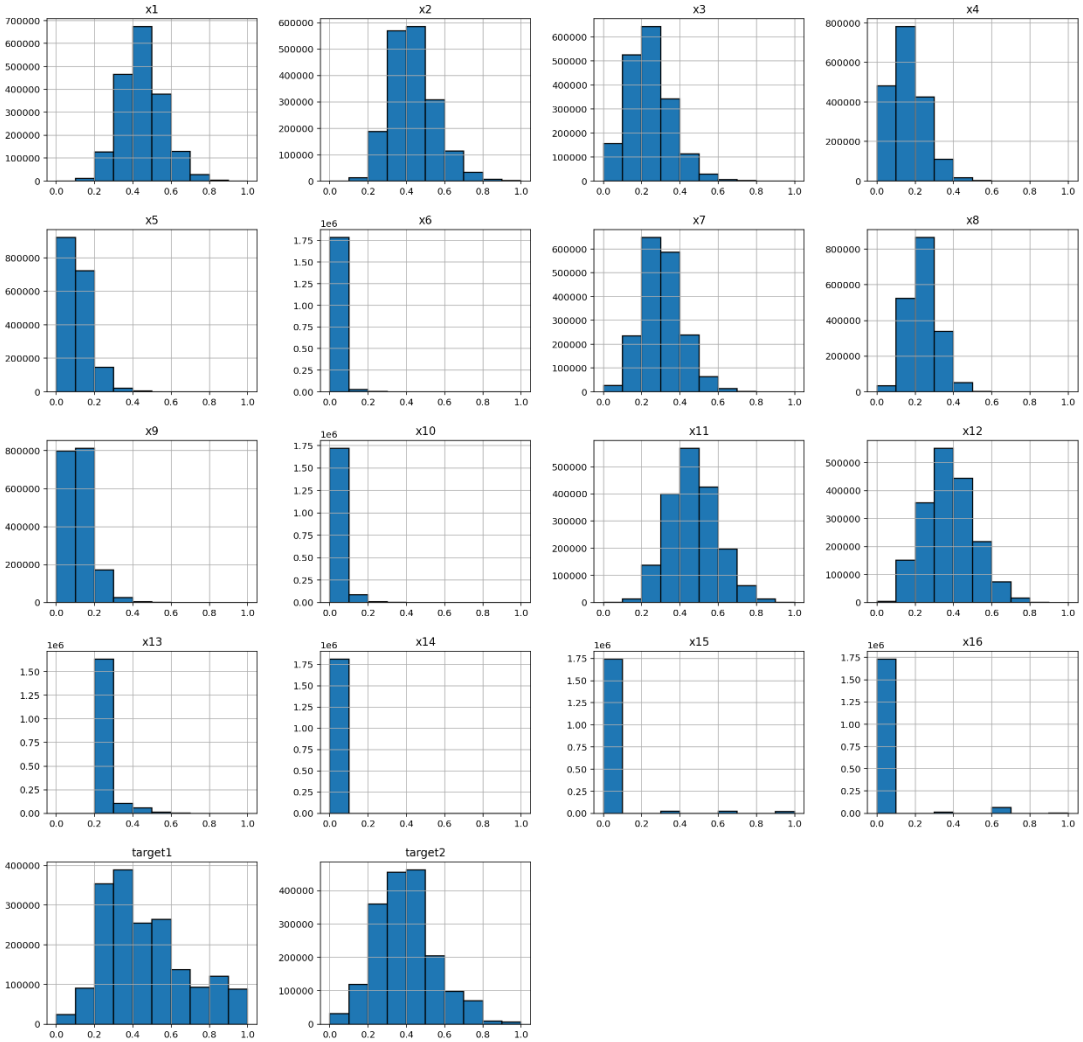

Рис. 3.7 – Гістограми розподілу даних

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 рис.3.8.

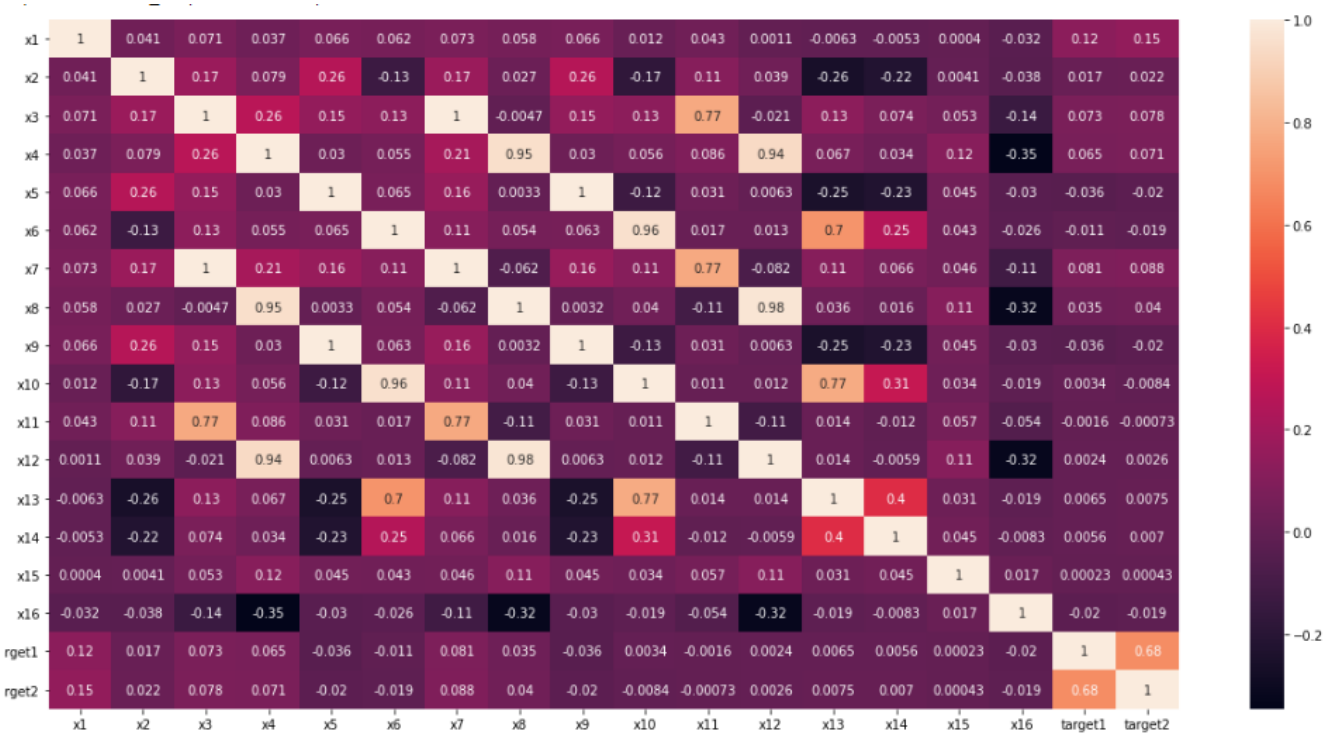

Рис. 3.8 – Кореляційна карта – матриця

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. 3.9.

	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

Рис. 3.9 – 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 . Математичне пояснювання:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

де min і max задаються як мінімальне і максимальне допустиме значення, за замовчуванням $min = 0,\ max = 1$. Стандартизовані дані зображені на рис. 3.10.

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

Рис. 3.10 – Стандартизовані дані

3.2 Результати моделей машинного навчання

Далі будуємо 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. Використовувались такі метрики прогнозної якості, а також коефіцієнт детермінації R^2 :

– коефіцієнт R^2

$$R^2 = 1 - \frac{u}{v} = (y - y')^2 \qquad (2.10)$$

де u– залишкова сума квадратів

v – загальна сума квадратів

– середня квадратична помилка:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad (2.10)$$

– середня абсолютна помилка:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x| \qquad (2.11)$$

– середня абсолютна відсоткова помилка:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \qquad (2.12)$$

– середня відсоткова помилка:

$$MPE = \frac{100\%}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{y_i}\right) \quad (2.13)$$

Результати похибки прогнозної якості, а також коефіцієнта детермінації R^2 , наведені для таких моделей як, лінійна регресія, дерева рішень, випадкового лісу. Продемонстровано результати в табл. 3.1.

Таблиця 3.1

Результати похибки прогнозів використаних методів

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
Випадковій 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

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 рис.3.15. Дерево для target2 зображено на рис.3.16. 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.

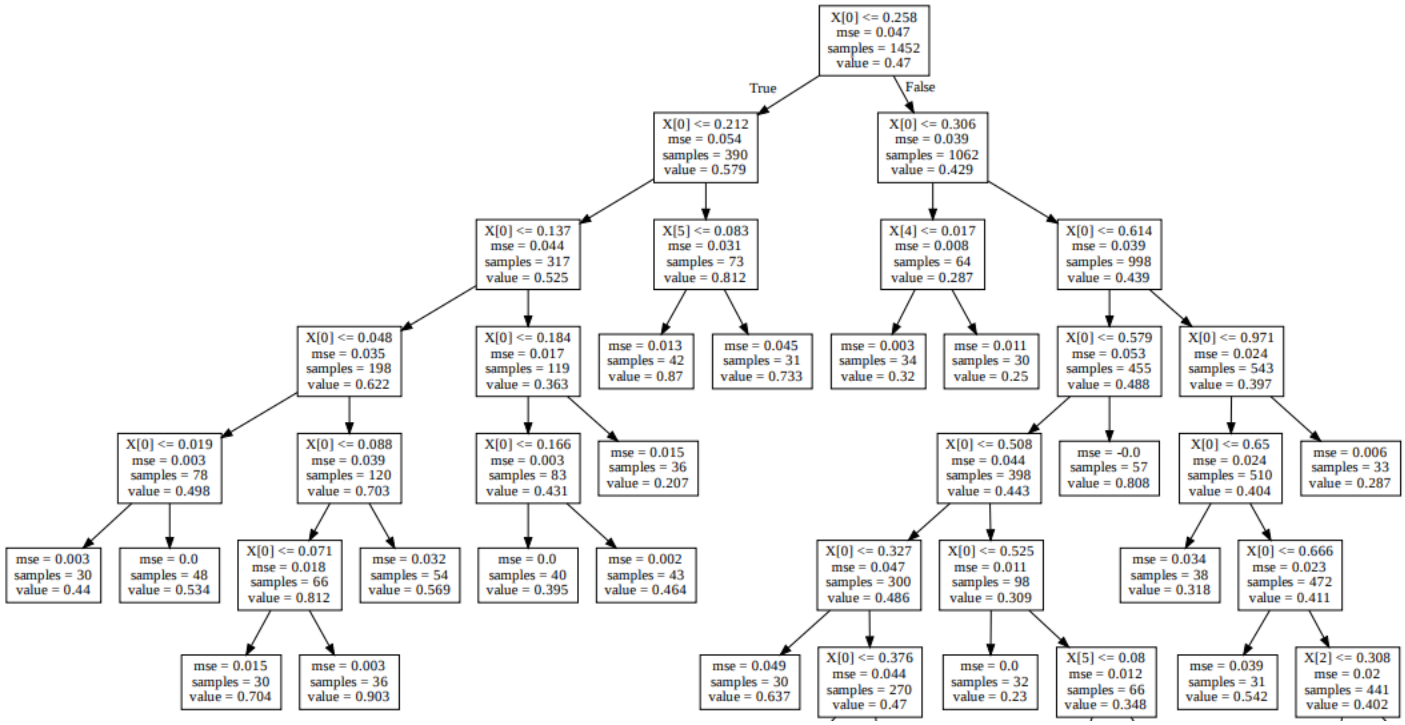

Рис. 3.14 – Дерево для цільової змінної (target1)

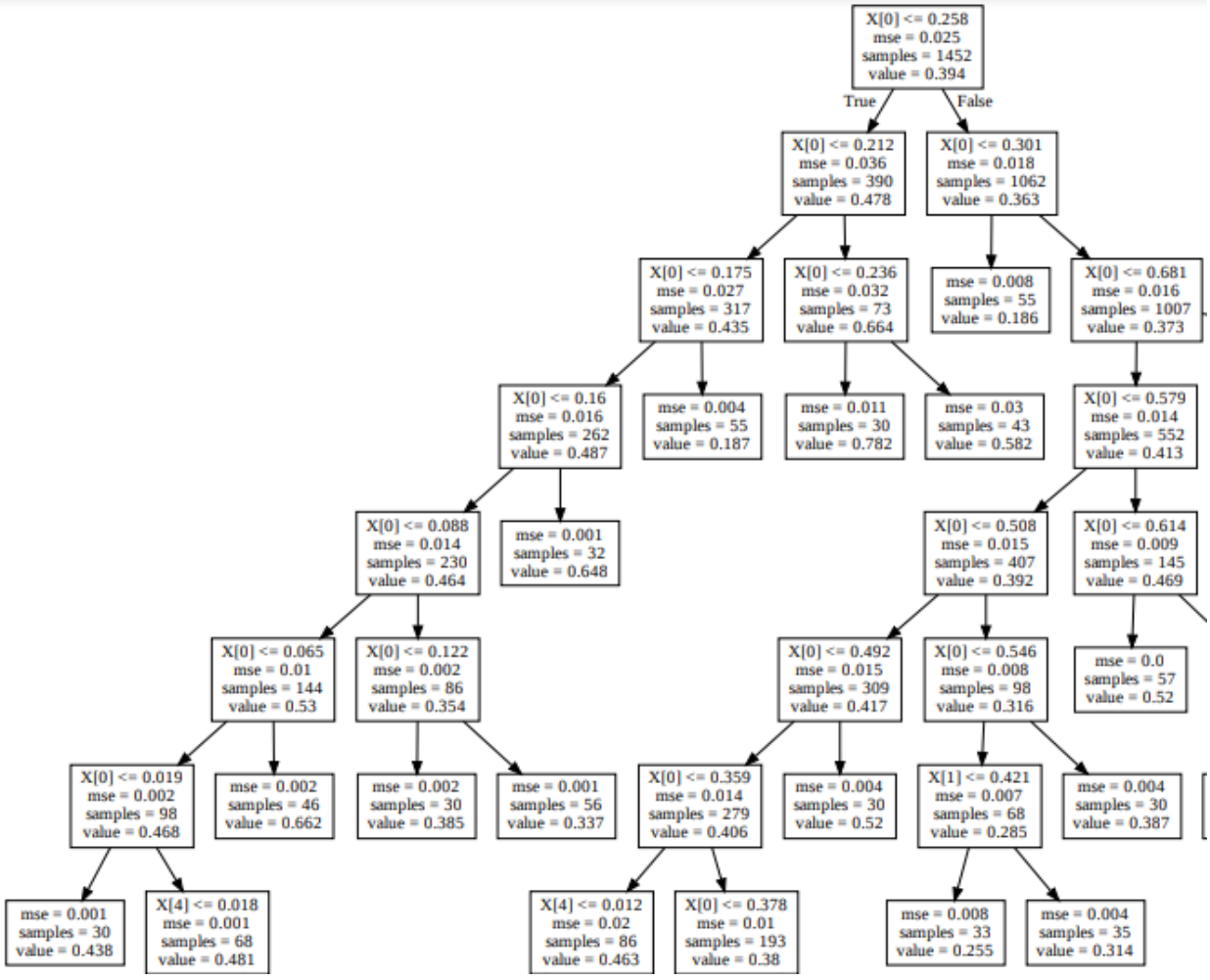

Рис. 3.15 – Дерево для цільової змінної (target2)

На рис. 3.16 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, а також це видно на прогнозних оцінках якості котрі були показані у табл. 3.1. Дані результати влаштовують потреби замовника і будуть передані йому.

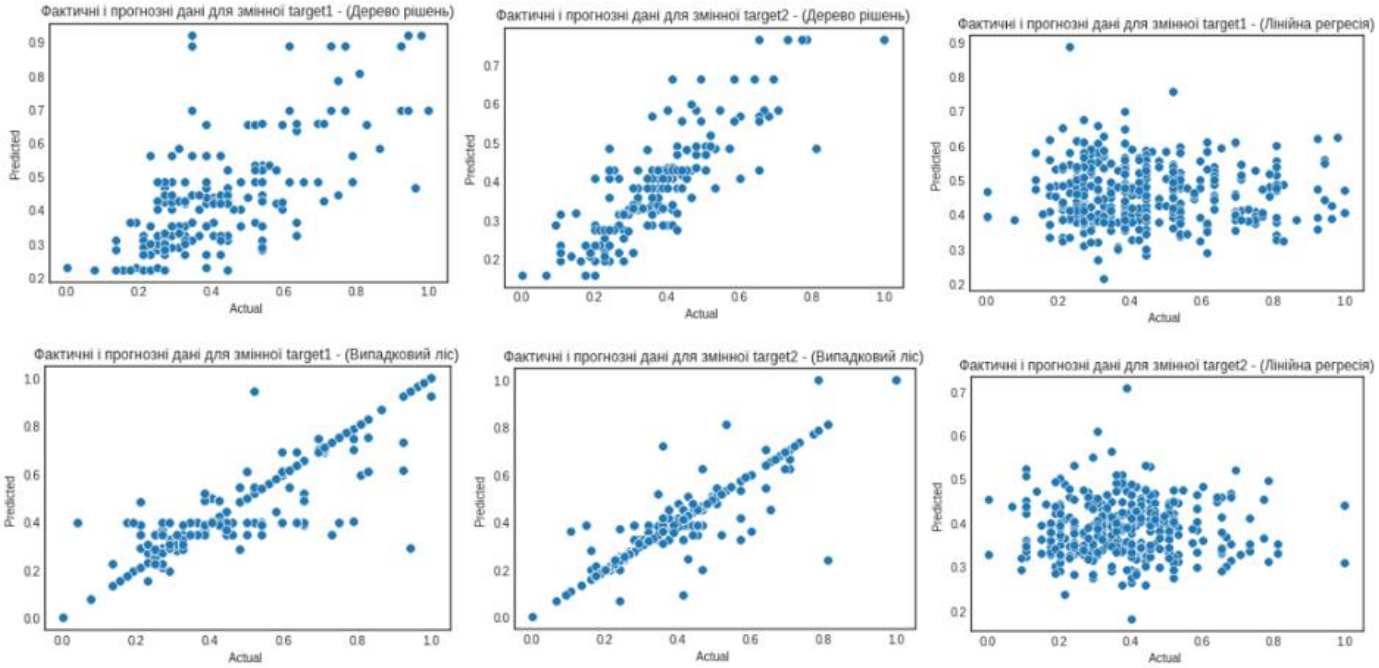

Рис. 3.16 – Результати прогнозних та фактичних даних

Для того, щоб узагальнити весь функціонал данної розробки, на рис. 3.17 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 легкістью 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 рис. 3.18.

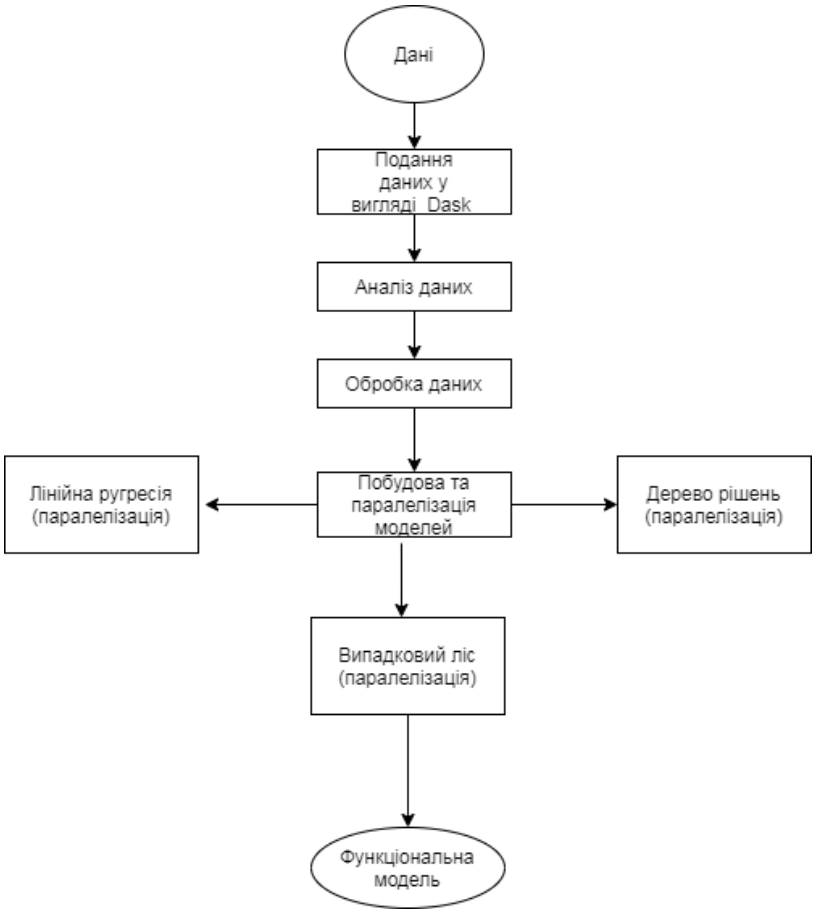

Рис. 3.17 – Блок схема роботи моделі

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

Рис. 3.18 – Вихідний результат

ВИСНОВКИ

При проходженні переддипломної практики був отриманий проект із ресурсу UpWork. Таким чином у ході проекту були проаналізовані та спрогнозовані дані. При розробці працюючої моделі для прогнозування цільових змінних були використані 2 алгоритма машиного навчання і статистичний метод як, лінійна регресія, випадковий ліс, дерево рішень.

Виходячи з результатів цих моделей був зроблений висновок, що даних недостатньо тому було прийняте рішення виконати конкатенацію дані іншим чином в результаті чого був отриманий набір даних з 1815696 стрічок. Оскільки стандартні бібліотеки python були не в змозі проаналізувати та спрогнозувати ці дані, було прийняте рішення використовувати інструменти для Big Data. Після аналізу інструментів для великих даних, було виявлено доволі складні та не зручні налаштування середовища, це є не зручним для замовника. Тому було звернено увагу на паралелізацію. Для data science існує бібліотека Dask. Яка є проста, зручна та ефективна у своїй роботі. Бібліотека Dask надала гарну можливість працювати з великим набором даних та зробити певні моделі машиного навчання. Були не лише проаналізовані моделі, а також побудовані візуалізації роботи певних з методів. Надана візуалізація результатів кожного методу машиного навчання. Виходячи з результатів то можна казати, що якіснішою моделлю виявилась модель випадкового лісу. Модель продемонструвала високі прогнозні оцінки, а також якісні. Вона надала дійсно точне прогнозування, яка задовольнила вимогам замовника.

ПЕРЕЛІК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Беркана А. Что такое Big data: собрали всё самое важное о больших данных. Rusbase. URL: https://rb.ru/howto/chto–takoe–big–data/ (дата звернення: 09.12.2020)

2. Рокатанский М. Hadoop: что, где и зачем. Хабр. URL: https://m.habr.com/ru/post/240405/ (дата звернення: 02.12.2020)

3. 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 (date of access: 03.12.2020).

4. 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/ (date of access: 01.12.2020).

5. 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 (date of access: 03.12.2020).

6. Trinidad A. Top 10 Big Data Issues and How to Solve Them. Datafloq. URL: https://datafloq.com/read/top–10–big–data–issues–how–to–solve–them/2335 (date of access: 06.12.2020).

7. Machhi C. PySpark – SparkContext. tutorialspoint. URL: https://www.tutorialspoint.com/pyspark/pyspark_sparkcontext.htm (date of access: 04.12.2020).

8. Андреас МюллерВведение в машинное обучение с помощью Python – Пер. сангл. – СПб.: Символ–Плюс, 2016. – 15 с.

9. Николенко С. Базовые принципы машинного обучения на примере линейной регрессии. Habr. URL: https://habr.com/ru/company/ods/blog/322076/ (дата звернення: 05.12.2020).

10. Pathak P. KDdnuggets. A Beginner’s Guide to Linear Regression in Python with Scikit–Learn. URL: https://www.kdnuggets.com/2019/03/beginners–guide–linear–regression–python–scikit–learn.html (date of access: 10.12.2020).

11. Sklearn.linear_model.LinearRegression. sklearn. URL: https://scikit–learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (date of access: 06.12.2020).

12. 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 (date of access: 07.12.2020).

13. Kashnitsky Y. Открытый курс машинного обучения. Open Data Science. URL: https://habr.com/ru/company/ods/blog/323890/ (дата звернення: 11.12.2020).

14. Kashnitsky Y. Topic 3. Classification, Decision Trees and k Nearest Neighbors. Open Machine Learning Course. URL: https://medium.com/open–machine–learning–course/open–machine–learning–course–topic–3–classification–decision–trees–and–k–nearest–neighbors–8613c6b6d2cd (date of access: 17.12.2020).

15. Brownlee J. Why Use Ensemble Learning?. Machine Learning Mastery. URL: https://machinelearningmastery.com/why–use–ensemble–learning/ (date of access: 18.12.2020).

16. Tibshirani R. Decision Rules. Interpretable machine learning. URL: https://christophm.github.io/interpretable–ml–book/rules.html (date of access: 19.12.2020).

17. Lorraine J. Classification and Regression Analysis with Decision Trees. Dev. URL: https://dev.to/nexttech/classification–and–regression–analysis–with–decision–trees–jgp (date of access: 20.12.2020).

18. sklearn.tree.DecisionTreeRegressor. sklearn. URL: https://scikit–learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html (date of access: 18.12.2020).

19. Menezez D. Ensemble Learning: 5 Main Approaches. KDnuggets. URL: https://www.kdnuggets.com/2019/01/ensemble–learning–5–main–approaches.html (date of access: 19.12.2020).

20. Singh A. A Comprehensive Guide to Ensemble Learning (with Python codes). Analytics Vidhya. URL: https://www.analyticsvidhya.com/blog/2018/06/comprehensive–guide–for ensemble–models/ (date of access: 21.12.2020).

21. 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/ (date of access: 23.12.2020).

22. Lutins E. Ensemble Methods in Machine Learning: What are They and Why Use Them?. towards data science. URL: https://towardsdatascience.com/ensemble–methods–in–machine–learning–what–are–they–and–why–use–them–68ec3f9fef5f (date of access: 13.12.2020).

23. 3.2.4.3.2. sklearn.ensemble.RandomForestRegressor. sklearn. URL: https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html (date of access: 14.12.2020).

24. pandas documentation. pandas. URL: https://pandas.pydata.org/docs/ (date of access: 25.12.2020).

25. NumPy. NumPy. URL: https://numpy.org/ (date of access: 28.12.2020).

26. Dask natively scales Python. Dask. URL: https://dask.org/ (date of access: 28.12.2020).

27. Dask is a flexible library for parallel computing in Python. Dask. URL: https://docs.dask.org/en/latest/ (date of access: 29.12.2020).

28. Bakharia A. Почему каждый Data Scientist должен знать Dask. Habr. URL: https://habr.com/ru/company/piter/blog/454262/ (дата звернення: 30.12.2020).

29. Augspurger T. Dask for Machine Learning. Dask. URL: https://examples.dask.org/machine–learning.html (date of access: 31.12.2020).

Додаток А

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
```

Додаток Б

Технічне завдання до диплому проекту

Scope

1. Scope of this project is to predict as precisely as possible the values of the targets (Target1,Target2) relative to the validation set.

Data Sets

2. We provide two different data sets (DataSet–A & DataSet–B) for each data set one training and one validation set. **For both data sets holds the same goal 1.**
   DataSet–A contains 1024 samples split as follows: Training–A with 819 Samples (80%), Validation–A with 205 Samples (20%)

   DataSet–B contains 884 samples split as follows: Training–B with 707 Samples (80%) , Validation–B with 177 Samples (20%)

3. **IMPORTANT**: For both data sets : the information of each sample are split into two files.
   Consider for example **0003–101–005166**, it has 0003–101–005166_1–2–1–1.csv and 0003–101–005166_1–2–1–2.csv.

   This means that Training–A has 819*2 =1638 files and Validation–A 205*2=410 files.

   Training–B has 707*2 =1414 files and Validation–A 177*2=354 files

Target Values

The target values for both Data Sets can be found in: Table_DataSet–A.xlsx, Table_DataSet–B.xlsx