МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «**ДОСЛІДЖЕННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ В ТЕХНОЛОГІЯХ SMART CITY**»

Виконав: студент 2 курсу, групи 8.1222
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
В. В. Морозов
(ініціали та прізвище)

Керівник професор кафедри комп'ютерних наук, доцент, д.т.н. Шило Г. М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики ЗНУ, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри комп'ютерних наук, д.т.н., професор

_____ Шило Г.М.
(підпис)

" 05 " травня 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Шаповалову Ігореві Івановичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Дослідження методів машинного навчання в технологіях smart city

керівник роботи Шило Галина Миколаївна, д.т.н, доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 29.11.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація програмного забезпечення

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 05.05.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи	05.05.2023	
2.	Збір вихідних даних	25.05.2023	
3.	Обробка методичних та теоретичних	10.07.2023	
	джерел		
4.	Розробка першого та другого розділу	01.10.2023	
5.	Розробка третього розділу	01.10.2023	
6.	Оформлення та нормоконтроль	29.11.2023	
	кваліфікаційної роботи магістра		
7.	Захист кваліфікаційної роботи	13.12.2023	

Студент _____ В.В. Морозов
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.М. Шило
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Дослідження методів машинного навчання в технологіях smart city»: 64 с., 25 рис., 12 джерел, 3 додатки.

DATASET, KAGGLE, MACHINE LEARNING, SKLEARN, SMART CITY.

Об’єкт дослідження – технології та методи взаємодії машинного навчання зі smart city.

Мета роботи – дослідити методи взаємодії машинного навчання зі smart city.

Методи дослідження – програмний, аналітичний.

У світі, який стрімко розвивається, технології smart city стають необхідністю для оптимізації міського середовища та покращення якості життя громадян. Одним із ключових елементів впровадження smart city є використання методів машинного навчання, що дозволяють збирати та аналізувати величезний обсяг даних для прийняття розумних рішень.

Також постійний розвиток цих технологій відкриває нові можливості для створення більш інтелектуальних та ефективних міських середовищ.

Актуальність досліджень у цій області важлива, оскільки вона визначає можливості майбутнього розвитку міст та їхню готовність до викликів сучасності.

Таким чином, за результатами роботи були досліджені методи машиного навчання в технологіях smart city.

SUMMARY

Master's qualifying paper «Researching the Machine Learning Methods in Smart City Technologies»: 64 pages, 25 figures, 12 references, 3 supplements.

DATASET, KAGGLE, MACHINE LEARNING, SKLEARN, SMART CITY.

The object of the study is technologies and methods of interaction between machine learning and smart city.

The aim of the study is to study the methods of interaction between machine learning and smart city.

The methods of research are programming, analytical.

In a rapidly developing world, smart city technologies are becoming a necessity for optimizing the urban environment and improving the quality of life of citizens. One of the key elements of smart city implementation is the use of machine learning methods that allow collecting and analyzing a huge amount of data to make smart decisions.

The continuous development of these technologies also opens up new opportunities for creating more intelligent and efficient urban environments.

The relevance of research in this area is important because it determines the possibilities for future urban development and their readiness to meet the challenges of our time.

Thus, according to the results of the work, machine learning methods in smart city technologies were investigated.

ЗМІСТ

ВСТУП

В сучасному світі міста стають центрами інтенсивного розвитку, збільшуючи своє населення та вирішуючи складні завдання управління ресурсами та послугами. Однак, для досягнення оптимальної ефективності та сталого розвитку, сучасні міста все більше спираються на концепцію "смарт-сіті" та використовують передові технології, такі як машинне навчання.

Смарт-сіті, або інтелектуальні міста, стали ключовим елементом стратегій розвитку у зв'язку зі зростаючою комплексністю та масштабом сучасних міських систем. Завдяки використанню інформаційно-комунікаційних технологій, сенсорів та машинного навчання, смарт-сіті забезпечують ефективне управління ресурсами, оптимізують інфраструктуру та покращують якість життя мешканців.

Виходячи з цього, було вирішено дослідити, як саме методи машинного навчання можна пов’язати з технологіями smart city.

Актуальність дослідження: актуальність теми зумовлена стрімким розвитком технологій, які дозволяють оптимізувати та уніфікувати процеси керування складовими міста.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження:

Мета: розширення розуміння та визначенні оптимальних стратегій впровадження машинного навчання в смарт-сіті проекти для досягнення сталого розвитку та покращення якості міського середовища.

Задачі:

1) Оглянути та написати рецензію на літературу за темою;
2) Розглянути зв’язок смарт-сіті та машинного навчання;
3) Дослідити та порівняти різні підходи машинного навчання щодо обробки та аналізу даних процесів смарт-сіті;
4) Зробити висновки.

Об’єкт дослідження: процес дослідження методів машинного навчання, використання методів в технологіях смарт-сіті.

Предмет дослідження: методи машинного навчання та технології смарт-сіті.

Методи дослідження: програмний, аналітичний.

Перший розділ присвячено присвячений огляду та рецензії літератури на тему дослідження.

Другий розділ фокусується на конкретних аспектах смарт-сіті та їхньому зв'язку з машинним навчанням.

Третій розділ включає практичну частину з розробки та порівняння різних підходів машинного навчання.

1 ОГЛЯД ТА РЕЦЕНЗІЯ ЛІТЕРАТУРИ

1.1 Визначення основних термінів та понять

Смарт-сіті (іноді також використовується термін "інтелектуальне місто") – це міське середовище, в якому використовуються інноваційні технології для оптимізації та покращення якості життя мешканців. Основні характеристики смарт-сіті включають в себе використання інформаційно-комунікаційних технологій (ІКТ), сенсорів та інших засобів для збору та аналізу даних з метою ефективного управління ресурсами та послугами міста.

Машинне навчання – галузь штучного інтелекту, що дозволяє комп'ютерам вчитися з досвіду та вдосконалювати свою продуктивність у вирішенні завдань без явного програмування. Алгоритми машинного навчання використовуються для аналізу даних та розпізнавання взорів, що робить їх ефективними інструментами для обробки великого обсягу інформації, що надходить від сенсорів та інших джерел в смарт-сіті.

Зв'язок між цими двома поняттями полягає в тому, що машинне навчання може використовуватися для аналізу та обробки великої кількості даних, які збираються у смарт-сіті, забезпечуючи оптимальне управління ресурсами та послугами для поліпшення життя громадян. У цьому контексті, машинне навчання може бути використано для прогнозування попиту на ресурси, оптимізації систем енергозабезпечення, розпізнавання та вирішення проблем безпеки та інших аспектів, що важливі для сталого розвитку міста.

1.2 Зв'язок між смарт-сіті та машинним навчанням

Смарт-сіті концепція базується на використанні передових технологій для оптимізації управління різними аспектами міського життя. Машинне навчання, як важлива галузь штучного інтелекту, стає ключовим інструментом у впровадженні смарт-сіті рішень.

Роль машинного навчання в розвитку смарт-сіті:

1) *Оптимізація управління ресурсами:* Машинне навчання дозволяє аналізувати великі обсяги даних щодо споживання енергії, води та інших ресурсів. На основі цього аналізу системи управління можуть ефективно регулювати постачання ресурсів в реальному часі.
2) *Прогнозування попиту на послуги:* Машинне навчання може бути використане для аналізу історичних даних та прогнозування змін в попиті на міські послуги. Це дозволяє адаптувати ресурси та інфраструктуру для задоволення змінюючихся потреб мешканців.
3) *Розпізнавання та управління транспортним рухом:* Застосування машинного навчання у системах міської транспортної інфраструктури дозволяє покращити безпеку, оптимізувати потік транспорту та розробляти ефективні плани руху.

Наведемо приклади успішного впровадження машинного навчання в смарт-сіті проекти:

1) *Енергозабезпечення:* В Лондоні, система «Flex London» використовує машинне навчання для аналізу та прогнозування попиту на електроенергію у реальному часі. Це дозволяє оптимізувати роботу енергетичних систем, забезпечуючи стабільні постачання та зменшуючи витрати [3].

2) *Системи водопостачання:* В Барселоні, проект «Smart Water Platform» використовує машинне навчання для моніторингу та прогнозування споживання води у місті. Алгоритми аналізують дані з сенсорів та пристроїв для оптимізації роботи системи водопостачання [11].
3) *Управління відходами:* В Сіднеї, програма «Waste Less, Recycle More» використовує машинне навчання для вдосконалення управління відходами. Система аналізує дані від сенсорів на контейнерах для сміття, щоб прогнозувати рівень заповнення та оптимізувати маршрути вивезення сміття [12].

Ці приклади свідчать про ефективне впровадження машинного навчання в смарт-сіті проекти, що призводить до покращення управління ресурсами та послугами міста, а також до створення сталого та інноваційного міського середовища.

1.3 Огляд сучасних досліджень та публікацій

1.3.1 Аналіз статей і наукових досліджень

Стаття «IoT for Smart Cities: Machine Learning Approaches in Smart Healthcare»

У статті розглядається використання Інтернету речей (IoT), штучного інтелекту та машинного навчання для розвитку систем охорони здоров'я в контексті «розумних міст». Автори проводять ґрунтовний аналіз літератури з цієї тематики [5].

Основні ідеї статті:

- Бездротові сенсорні мережі (WSN) та IoT відкривають нові можливості для охорони здоров'я, збираючи та аналізуючи величезну кількість даних в режимі реального часу;

- IoT вже широко застосовується в охороні здоров'я – для моніторингу пацієнтів, «розумних пігулок», медичного обладнання тощо;
- Машинне навчання та AI дозволяють ефективно аналізувати дані та підтримувати прийняття рішень у сфері медицини;
- Розглянуто різні алгоритми машинного навчання – кероване, некероване та навчання з підкріпленням;
- Показано приклади застосування AI та машинного навчання для моніторингу стану здоров'я, діагностики та лікування захворювань тощо.

Стаття є ґрунтовним оглядом літератури щодо сучасних та майбутніх тенденцій в охороні здоров'я у «розумних містах». Робота охоплює всі ключові концепції, пов'язані з використанням AI та IoT у цій сфері.

Значною перевагою є детальний аналіз різних алгоритмів машинного навчання та демонстрація їх застосування на прикладах. Крім того, розглянуто як окремі елементи (WSN, IoT, AI), так і загальну картину їх інтеграції в системи охорони здоров'я.

До недоліків можна віднести певне дублювання інформації у деяких розділах. Також бракує більш детального обговорення проблем інформаційної безпеки таких систем.

Загалом, стаття може бути корисною для науковців та практиків, які цікавляться використанням сучасних інформаційних технологій в охороні здоров'я. Вона містить ґрунтовну базу знань про сучасний стан, проблеми та перспективи цієї галузі.

Стаття «Cyberattacks Detection in IoT-Based Smart City Applications Using Machine Learning Techniques»

У статті досліджується машинне навчання для виявлення кібератак в додатках розумних міст на основі Інтернету речей (IoT). Запропоновано підхід на основі ансамблевого навчання (bagging, boosting, stacking) та окремих класифікаторів (логістична регресія, машина опорних векторів, дерева рішень тощо) для виявлення аномалій та кібератак в мережевому трафіку [2].

Основні ідеї:

- Використано набори даних UNSW-NB15 та CICIDS2017, що містять сучасні кібератаки в IoT;
- Застосовано інформаційний критерій для вибору найвпливовіших ознак;
- Досліджено як окремі класифікатори, так і ансамблеві методи машинного навчання;
- Показано, що stacking-ансамбль дає кращі результати за метриками точності, відклику тощо;
- Запропонований підхід ефективно виявляє різні типи кібератак в IoT мережах.

Стаття присвячена актуальній темі виявлення кібератак в Інтернеті речей за допомогою машинного навчання. Робота ґрунтується на сучасних даних та враховує особливості IoT. Значною перевагою є порівняння як окремих алгоритмів, так і їх ансамблів.

Використання інформаційного критерію для відбору ознак та крос-валідації також підвищує якість дослідження. Основні результати є переконливими та демонструють ефективність stacking для виявлення кібератак.

Деякі недоліки статті включають відсутність аналізу складності обчислень та затримок для ансамблевих методів. Також бажано було б порівняти результати з більшою кількістю існуючих моделей.

В цілому, стаття містить корисний матеріал та результати для дослідників в галузі кібербезпеки Інтернету речей та розумних міст.

Стаття «Machine Learning in Wireless Sensor Networks for Smart Cities: A Survey»

У статті розглядається застосування машинного навчання в бездротових сенсорних мережах на основі Інтернету речей для розумних міст. Проведено детальний огляд алгоритмів машинного навчання, які можуть бути використані для вирішення різних проблем у таких мережах [8].

Основні ідеї:

- Бездротові сенсорні мережі є базовою інфраструктурою Інтернету речей;
- Машинне навчання дозволяє оптимізувати роботу сенсорних вузлів та мереж;
- Розглянуто 3 категорії машинного навчання: кероване, некероване та з підкріпленням;
- Представлено конкретні алгоритми та їх застосування для локалізації вузлів, маршрутизації, агрегації даних тощо;
- 61% опрацьованих робіт використовують кероване навчання, 27% - навчання з підкріпленням;
- Запропоновано використання машинного навчання для «розумних» систем охорони здоров'я.

Стаття містить ґрунтовний аналіз літератури з питань застосування машинного навчання в бездротових сенсорних мережах на основі Інтернету речей. Автори демонструють глибоке розуміння предметної області та різноманітності алгоритмів машинного навчання.

Позитивним є те, що розглянуто як теоретичні концепції, так і конкретні приклади використання в сенсорних мережах. Також наведено статистику щодо поширеності різних підходів в опрацьованій літературі.

Деяким недоліком є відсутність більш детального аналізу обмежень машинного навчання на ресурсо-обмежених сенсорних вузлах. Також актуальним було б розглянути питання інформаційної безпеки в контексті застосування машинного навчання.

Загалом, статтю можна рекомендувати як узагальнення сучасного досвіду застосування машинного навчання в бездротових сенсорних мережах на базі Інтернету речей у «розумних містах». Вона може бути корисною для науковців та інженерів у цій сфері.

Стаття «Machine learning based system for managing energy efficiency of public sector as an approach towards smart cities»

У статті розглядається питання енергоефективності в державному секторі та пропонується архітектура інтелектуальної системи управління енергоефективністю на основі машинного навчання [7].

Автори спочатку створюють прогнозні моделі споживання енергії громадських будівель за допомогою трьох методів машинного навчання: глибоких нейронних мереж, дерев рішень Rpart та випадкового лісу. Моделі тестуються на реальних даних з хорватської системи управління енергоспоживанням EMIS.

Найточніша модель отримана за допомогою алгоритму випадкового лісу, яка дає похибку прогнозування 13,6%. Всі моделі вибирають схожий набір найважливіших предикторів, пов'язаних з потужністю обігріву, внутрішньою температурою та даними про зайнятість.

На основі цих моделей автори пропонують архітектуру інтелектуальної системи управління енергоефективністю в державному секторі MERIDA. Вона складається з 6 рівнів: збір даних, попередня обробка, прогнозні моделі, візуалізація, прийняття рішень та оцінка вигоди. Система використовує IoT, Big Data та машинне навчання для автоматизованого збору даних, прогнозування споживання енергії та підтримки прийняття рішень щодо інвестицій в енергоефективність.

Стаття є актуальною та своєчасною, оскільки розглядає важливе питання енергоефективності в контексті концепції «розумного міста». В ній пропонуються конкретні моделі машинного навчання для прогнозування енергоспоживання, а також детальна архітектура інтелектуальної системи підтримки прийняття рішень, якої бракує в попередніх дослідженнях.

Основним внеском є застосування передових методів штучного інтелекту, зокрема глибокого навчання, для створення точних прогнозних моделей енергоспоживання. Крім того, дослідження виділяє найвагоміші фактори, які впливають на енергоефективність. Це дає практичні рекомендації для прийняття управлінських рішень.

Основним недоліком є невеликий розмір вибірки через проблеми з валідацією даних. У майбутньому автори планують розширити вибірку та покращити точність моделей. Також потрібно реально впровадити запропоновану систему та оцінити отриманий економічний ефект.

1.3.2 Визначення тенденцій та перспектив

Основні тенденції та перспективи розвитку машинного навчання в смарт сіті на основі цих розглянутих статей можна узагальнити так:

- Інтернет речей (IoT) та бездротові сенсорні мережі є базовою інфраструктурою для створення смарт сіті та «розумних міст». Вони дозволяють збирати величезні масиви даних в режимі реального часу.
- Машинне навчання активно застосовується для аналізу цих даних з метою прогнозування, класифікації, кластеризації, оптимізації та підтримки прийняття рішень в IoT та смарт сіті.
- Поширеними є методи керованого навчання (логістична регресія, нейронні мережі, машина опорних векторів, дерева рішень). Також застосовується некероване навчання (кластеризація) та навчання з підкріпленням.
- Активно розвиваються ансамблеві методи, що поєднують декілька алгоритмів машинного навчання. Вони часто дають кращі результати, ніж окремі моделі.
- Зростає увага до глибокого навчання та нейронних мереж, зокрема рекурентних та згорткових, для аналізу даних з IoT та сенсорів.
- Перспективним є використання машинного навчання в сферах «розумної» енергетики, охорони здоров'я, транспорту, безпеки та ін. Це дозволить підвищити ефективність та якість послуг в смарт сіті.
- Зростають вимоги до обробки даних в режимі реального часу, що потребує оптимізації алгоритмів машинного навчання.

– Важливим напрямком є інформаційна безпека смарт сіті, зокрема виявлення аномалій та кібератак за допомогою методів машинного навчання.

Отже, машинне навчання відіграватиме все більшу роль в аналітиці та оптимізації смарт сіті, сприяючи побудові інтелектуальних систем підтримки прийняття рішень в критичних сферах «розумних міст».

2 SMART CITY ТА МАШИННЕ НАВЧАННЯ

2.1 Опис smart city

2.1.1 Визначення та ключові характеристики smart city

Smart city (розумне місто) – це концепція інтеграції кількох інформаційних та комунікаційних технологій для управління міським середовищем та його складовими, такими як житлові будинки, транспорт, освіта, охорона здоров'я, утилізація відходів, водопостачання та енергопостачання (рис. 2.1).

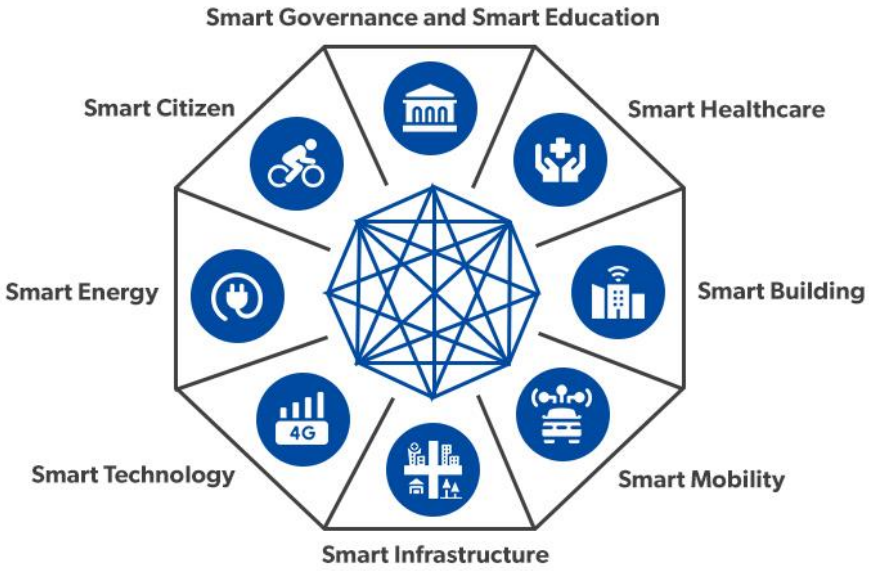

Рисунок 2.1 – Коцепція smart city

Основною метою впровадження smart city є підвищення якості життя мешканців за рахунок оптимізації використання ресурсів та прийняття своєчасних і зважених рішень в управлінні містом [9].

Ключові характеристики концепції:

– Застосування інформаційно-комунікаційних технологій та Інтернету речей для збору і аналізу даних в режимі реального часу;

- Використання хмарних та інших видів розподіленої обробки і зберігання даних;
- Впровадження цифрових платформ для інтеграції, аналізу даних і підтримки прийняття рішень;
- Розгортання мережі датчиків та сенсорів для моніторингу стану інфраструктури та навколишнього середовища;
- Застосування технологій штучного інтелекту та машинного навчання для оптимізації бізнес-процесів та прогнозної аналітики;
- Впровадження інтелектуальних систем освітлення, опалення, електропостачання та інших комунальних послуг;
- Розвиток концепцій «розумних будинків» та «розумних споруд»;
- Використання технологій доповненої та віртуальної реальності;
- Залучення громадян до взаємодії з цифровими сервісами та прийняття рішень.

2.1.2 Роль технологій у розвитку концепції smart city

Концепція smart city ґрунтується на широкому застосуванні сучасних інформаційно-комунікаційних технологій для інтегрованого управління активами та ресурсами міста [5].

Ключову роль у її реалізації відіграють:

- Технології Інтернету речей (IoT) – мережі датчиків, пристроїв та об’єктів з вбудованими технологіями для взаємодії один з одним або з зовнішнім середовищем. IoT є базовою концепцією для збору даних у режимі реального часу;
- Технології бездротового зв’язку – Wi-Fi, Bluetooth, стільникові мережі 3G/4G/5G, ZigBee, LoRaWAN тощо. Забезпечують передачу даних від датчиків та пристроїв до хмари або локальних хабів;

- Технології Big Data – платформи та інструменти для збору, зберігання та аналізу даних великих об’ємів з високою швидкістю надходження;
- Хмарні технології – моделі надання обчислювальних потужностей, баз даних, аналітичних сервісів як Інтернет-сервісів;
- Технології розподілених реєстрів (блокчейн) – децентралізовані бази даних операцій між незалежними суб’єктами;
- Методи штучного інтелекту та машинного навчання – нейронні мережі, глибоке навчання, кластеризація для аналізу даних та підтримки прийняття рішень;
- Інтелектуальні платформи управління – програмні комплекси для інтеграції даних, моделювання ситуацій та оптимізації бізнес-процесів.
- Інтеграція операційних та інформаційних технологій (IT/OT інтеграція) для автоматизації процесів виробництва, логістики, постачання товарів та послуг;
- Технології кібербезпеки – для захисту даних та інфраструктури smart city;
- AR/VR – технології доповненої (AR) та віртуальної (VR) реальності для візуалізації та взаємодії з даними.

2.2 Вплив машинного навчання на smart city

Технології машинного навчання (Machine Learning) та штучного інтелекту (AI) є ключовими складовими для реалізації концепції smart city, оскільки вони забезпечують інтелектуальний аналіз даних, прогнозну аналітику та підтримку прийняття рішень.

Наведемо, як машинне навчання використовується у smart city:

- Аналіз та прогнозування попиту на комунальні послуги – електроенергія, вода, тепло, газ. Дозволяє оптимізувати виробництво та розподіл ресурсів;

- Моніторинг та оптимізація руху на дорогах. Прогнозування заторів, визначення альтернативних маршрутів, управління світлофорами;
- Аналіз даних про злочинність та визначення зон ризику для розстановки патрулів поліції;
- Відеоаналітика – розпізнавання об’єктів та подій для відеоспостереження;
- Чат-боти та віртуальні асистенти для взаємодії з жителями та надання інформації;
- Аналіз відгуків та думок жителів з соцмереж для оцінки задоволеності роботою комунальних служб;
- Прогнозування попиту та оптимізація маршрутів громадського транспорту;
- Оцінка технічного стану та терміну служби інфраструктури міста на основі даних датчиків;
- Інтелектуальне керування вуличним освітленням з урахуванням погодних умов та руху людей/транспорту;
- Аналіз забруднення повітря за даними мережі сенсорів якості та рекомендації щодо його поліпшення.

2.2.1 Приклади впровадження машинного навчання у smart city

Компанія Huawei реалізувала проект Smart City в Китаї за підтримки технологій AI та машинного навчання. Система автоматично регулює вуличне освітлення, оптимізує маршрути громадського транспорту, відслідковує якість повітря та води.

Транспортний департамент Нью-Йорка застосовує AI стартап Metropolitan для аналізу заторів та інцидентів, що дозволяє ефективніше розподіляти ресурси та реагувати на них.

Корпорація Hitachi використовує відеоаналітику та комп'ютерний зір для автоматичного моніторингу дорожнього руху та виявлення дорожніх пригод у реальному часі.

Телеком-компанія Verizon аналізує дані про використання електромережі та погодні умови у різних районах міста за допомогою алгоритмів машинного навчання. Це дозволяє точніше прогнозувати навантаження та попит на електроенергію для її оптимального розподілу.

Microsoft розробляє платформу Azure Digital Twins, що поєднує Інтернет речей, машинне навчання та моделювання для створення цифрових двійників будівель та міської інфраструктури. Це дає змогу оптимізувати використання ресурсів, знизити витрати та підвищити безпеку мешканців.

Отже, технології штучного інтелекту та машинного навчання відіграють ключову роль у розвитку концепції smart city. Вони використовуються в різних сферах – транспорті, енергетиці, ЖКГ, безпеці, екології тощо. Це дозволяє автоматизувати процеси моніторингу, аналізу даних та прийняття рішень для комплексного управління міською інфраструктурою та сервісами.

На рисунку 2.2 наведено загальну схему взаємодії smart city та машинного навчання.

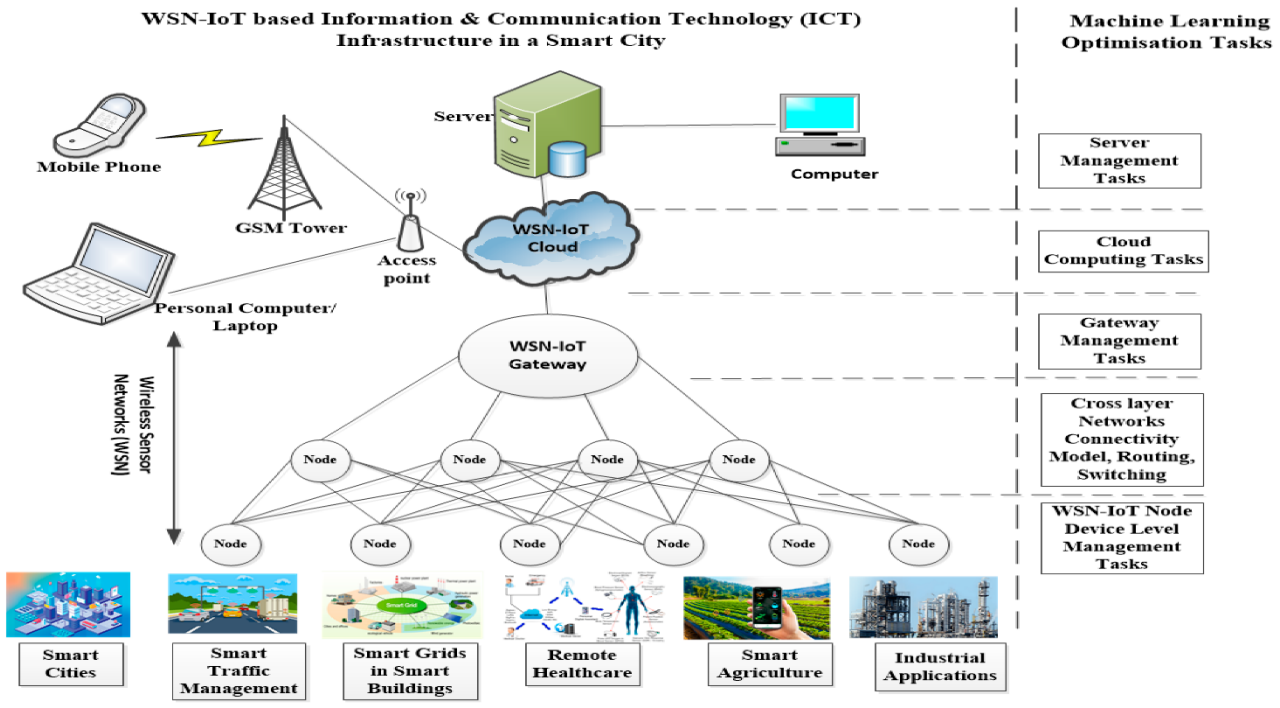

Рисунок 2.2 – Загальна схема

2.3 Візуальні засоби представлення взаємозв'язку машинного навчання зі smart city

Взаємозв'язок машинного навчання із смарт-містом (smart city) може бути представлений за допомогою різноманітних візуальних засобів. Давайте розглянемо, які саме інструменти можна використовувати:

Схема: На даній схемі (рис. 2.3) представлено компоненти типової системи смарт-мережі – датчики, лічильники, IoT платформа, хмарне сховище даних, а також модуль машинного навчання для аналізу даних і прийняття рішень.

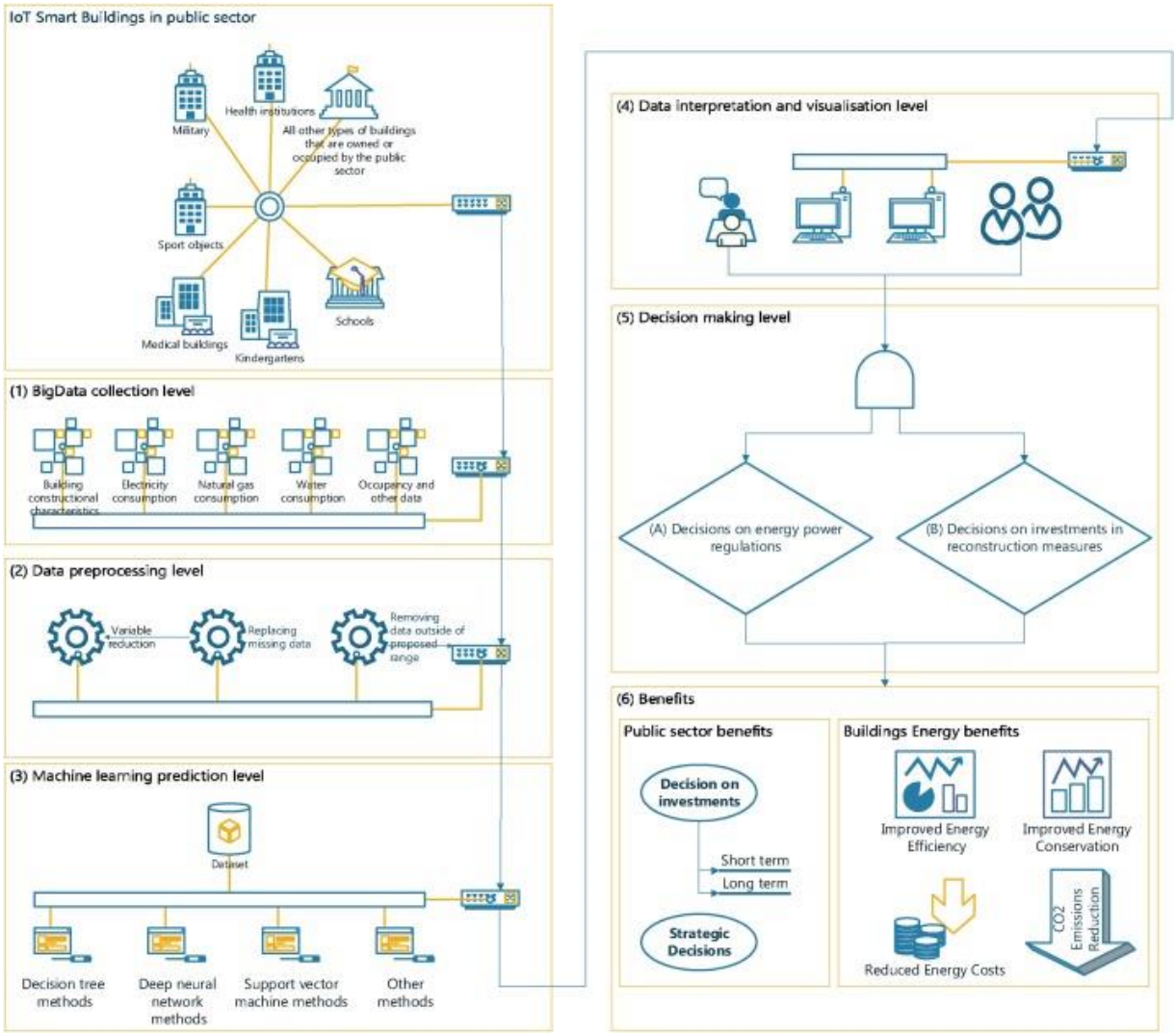

Рисунок 2.3 – Схема

Діаграма застосування: На діаграмі (рис. 2.4) представлено основні сфери, де алгоритми машинного навчання можуть застосовуватися в системах смарт-мережі:

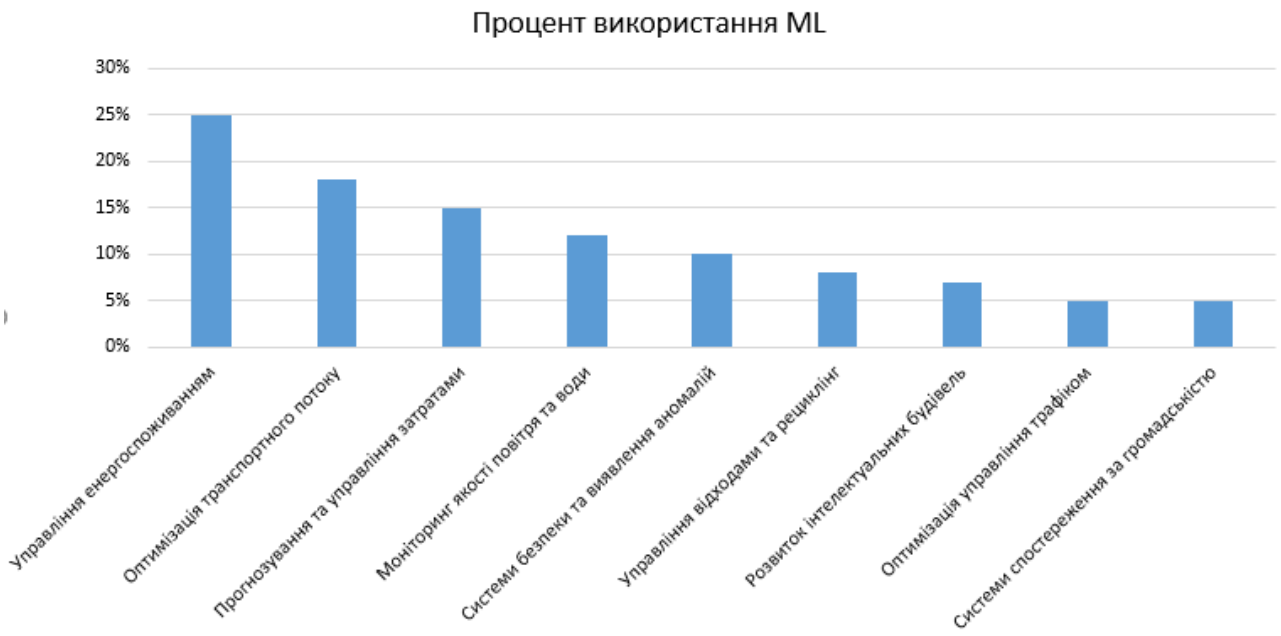

Рисунок 2.4 – Діаграма використання

Графік: Графік (рис. 2.5) демонструє використання нейронної мережі для прогнозування попиту на електроенергію – на основі історичних даних модель робить точніший прогноз на майбутнє.

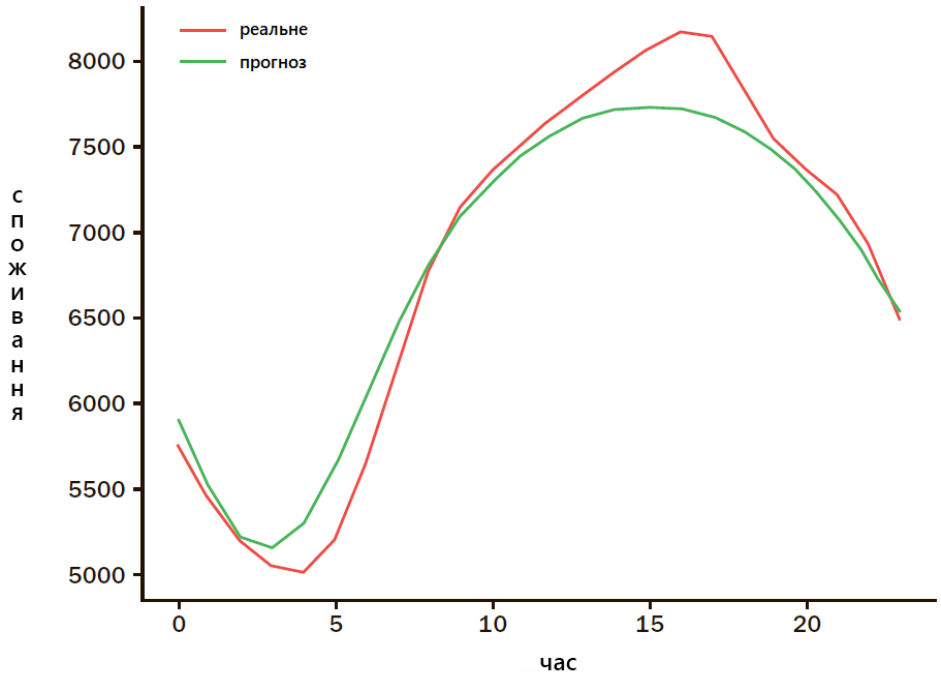

Рисунок 2.5 – Графік прогнозу попиту на е/е з ML

Отже, за допомогою наведених схем, графіків та діаграм можна наочно продемонструвати місце та роль машинного навчання в системах смарт-мережі, а також його переваги для вирішення практичних завдань.

3 ПРАКТИЧНА ЧАСТИНА

3.1 Вибір та опис наборів даних для аналізу

3.1.1 Dataset «Air Quality Dataset»

Цей набір даних містить відповіді газового мультисенсорного пристрою, встановленого на полі в одному з італійських міст. Середні погодинні значення відповідей записані разом із значеннями концентрації газу, отриманими від сертифікованого аналізатора. Цей набір даних взято з репозиторію машинного навчання UCI: https://archive.ics.uci.edu/ml/index.php.

Набір даних містить 9357 випадків усереднених погодинних відповідей з масиву з 5 хімічних датчиків на основі оксидів металів, вбудованих у хімічний мультисенсорний пристрій для вимірювання якості повітря. Пристрій був розташований на полі в значно забрудненій зоні, на рівні дороги, в межах італійського міста. Дані були записані з березня 2004 року по лютий 2005 року (один рік), що є найдовшими записами у вільному доступі про реакцію польових хімічних датчиків якості повітря. Ground Truth - це погодинні усереднені концентрації CO, неметанові вуглеводні, бензол, загальні оксиди азоту (NOx) та діоксид азоту (NO2), які були отримані від референтного сертифікованого аналізатора, розташованого поруч. Докази перехресної чутливості, а також концептуальні та сенсорні дрейфи присутні, як описано в De Vito та ін., Sens. And Act. B, Vol. 129,2,2008 (необхідне посилання), що в кінцевому підсумку впливає на здатність датчиків оцінювати концентрацію. Відсутні значення позначені зі значенням -200.

Інформація про атрибути:

1) Дата (ДД/ММ/РР);
2) Час (ЧЧ.ММ.СС);

3) Дійсне погодинне усереднене значення концентрації CO в мг/м^3 (еталонний аналізатор);
4) PT08.S1 (оксид олова) погодинний усереднений відгук датчика (номінально націлений на CO);
5) Істинна погодинно усереднена загальна концентрація неметанові вуглеводні в мкг/м^3 (еталонний аналізатор);
6) Істинна погодинна усереднена концентрація бензолу в мкг/м^3 (еталонний аналізатор);
7) PT08.S2 (титан), усереднений погодинний відгук датчика (номінально націлений на НМГЦ);
8) Істинна погодинна усереднена концентрація NOx в ppb (еталонний аналізатор);
9) PT08.S3 (оксид вольфраму) усереднений погодинний відгук датчика (номінально націлений на NOx);
10) Істинна погодинна усереднена концентрація NO2 в мкг/м^3 (еталонний аналізатор);
11) PT08.S4 (оксид вольфраму) усереднений погодинний відгук датчика (номінально націлений на NO2);
12) PT08.S5 (оксид індію) усереднений погодинний відгук датчика (номінально націлений на O3);
13) Температура в °C;
14) Відносна вологість (%);
15) Абсолютна вологість AH.

3.1.2 Dataset «New York City Bus Data»

Цей набір даних отримано з сервісу потокового передавання даних автобусів NYC MTA. Приблизно з 10-хвилинним інтервалом у кожному рядку вказано місцезнаходження автобуса, маршрут, зупинку тощо. Також включено

запланований час прибуття за розкладом, щоб показати, де має бути автобус (наскільки він відстає від графіка, встигає чи навіть випереджає його).

Інформація про атрибути:

1) Дата та час запису;
2) Напрямок маршруту;
3) Назва маршруту;
4) Інформація про автобус (назва, початкові координати);
5) Інформація про місце призначення;
6) Очікуваний час прибуття;
7) Запланноваий час прибуття.

3.1.3 Dataset «San Francisco Crime Classification»

Цей набір даних містить інформацію про інциденти, отриману з системи звітності про кримінальні інциденти поліції С.-Південної Флориди. Дані охоплюють період з 1.01.2003 по 13.05.2015. Навчальна та тестова вибірки змінюються щотижня, тобто тижні 1,3,5,7... належать до тестової вибірки, а тижні 2,4,6,8 - до навчальної. Оригінальний набір даних взято з SF OpenData, центрального інформаційного центру даних, опублікованих містом і округом Сан-Франциско.

Всього 9 змінних:

1) Дата – часова мітка події злочину;
2) Категорія – категорія інциденту (тільки в train.csv);
3) Опис – детальний опис інциденту (тільки в train.csv);
4) DayOfWeek – день тижня;
5) PdDistrict – назва району відділу поліції;
6) Вирішення – як було вирішено інцидент (тільки в train.csv);
7) Адреса – приблизна адреса вулиці, на якій сталася подія злочину;
8) X – довгота, Y – широта.

3.2 Air quality

В цьому розділі спрогнозуємо концентрацію шкідливих речовин в повітрі методами регресії.

Перед початком роботи необхідно імпортувати необхідні бібліотеки: numpy, pandas, matplotlib, seaborn, scikit-learn, а також вимикнути деякі попередження.

Завантажуємо набір даних із файлу CSV. Формат даних трохи нестандартний, тому вказуємо розділювач «;» та десятковий розділювач «,».

Виводимо інформацію про розмір набору даних, назви стовпців тощо (рис. 3.1).

	Date	Time	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH	Unnamed: 15	Unnamed: 16
0	10/03/2004	18.00.00	2.6	1360.0	150.0	11.9	1046.0	166.0	1056.0	113.0	1692.0	1268.0	13.6	48.9	0.7578	NaN	NaN
1	10/03/2004	19.00.00	2.0	1292.0	112.0	9.4	955.0	103.0	1174.0	92.0	1559.0	972.0	13.3	47.7	0.7255	NaN	NaN
2	10/03/2004	20.00.00	2.2	1402.0	88.0	9.0	939.0	131.0	1140.0	114.0	1555.0	1074.0	11.9	54.0	0.7502	NaN	NaN
3	10/03/2004	21.00.00	2.2	1376.0	80.0	9.2	948.0	172.0	1092.0	122.0	1584.0	1203.0	11.0	60.0	0.7867	NaN	NaN
4	10/03/2004	22.00.00	1.6	1272.0	51.0	6.5	836.0	131.0	1205.0	116.0	1490.0	1110.0	11.2	59.6	0.7888	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9466	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9467	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9468	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Рисунок 3.1 – Інформація про дані

Далі перевіряємо наявність пропущених значень, візуалізуємо за допомогою теплової карти (рис. 3.2).

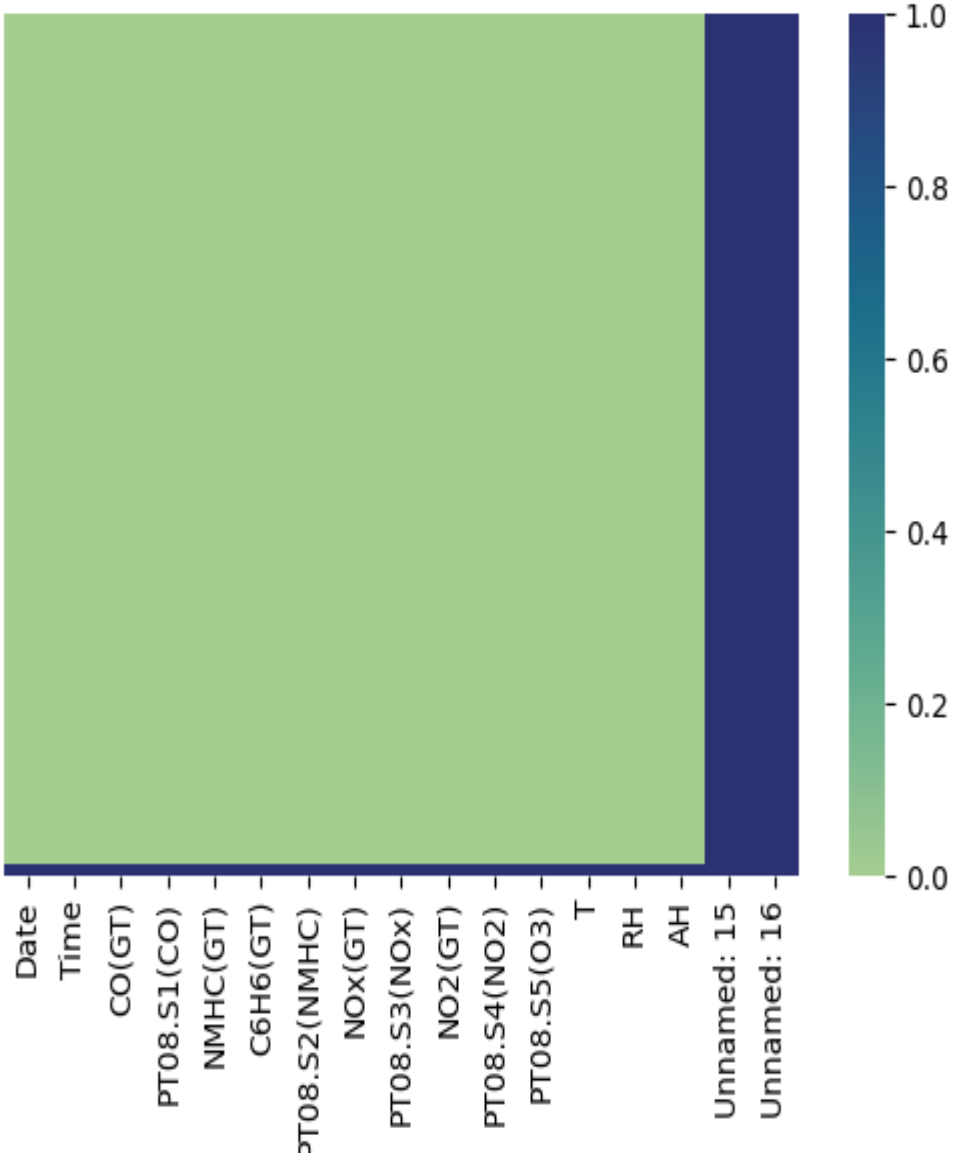

Рисунок 3.2 – Теплова карта

Необхідно видалити деякі зайві стовпці та рядки, а також значення «-200» замінити на пропущені значення NaN. Які в свою чергу заповнюються середнім по стовпцю.

Далі за допомогою міжквартильного розмаху визначаємо викиди і замінюємо на медіану. Будуємо теплову карту кореляцій між ознаками (рис. 3.3).

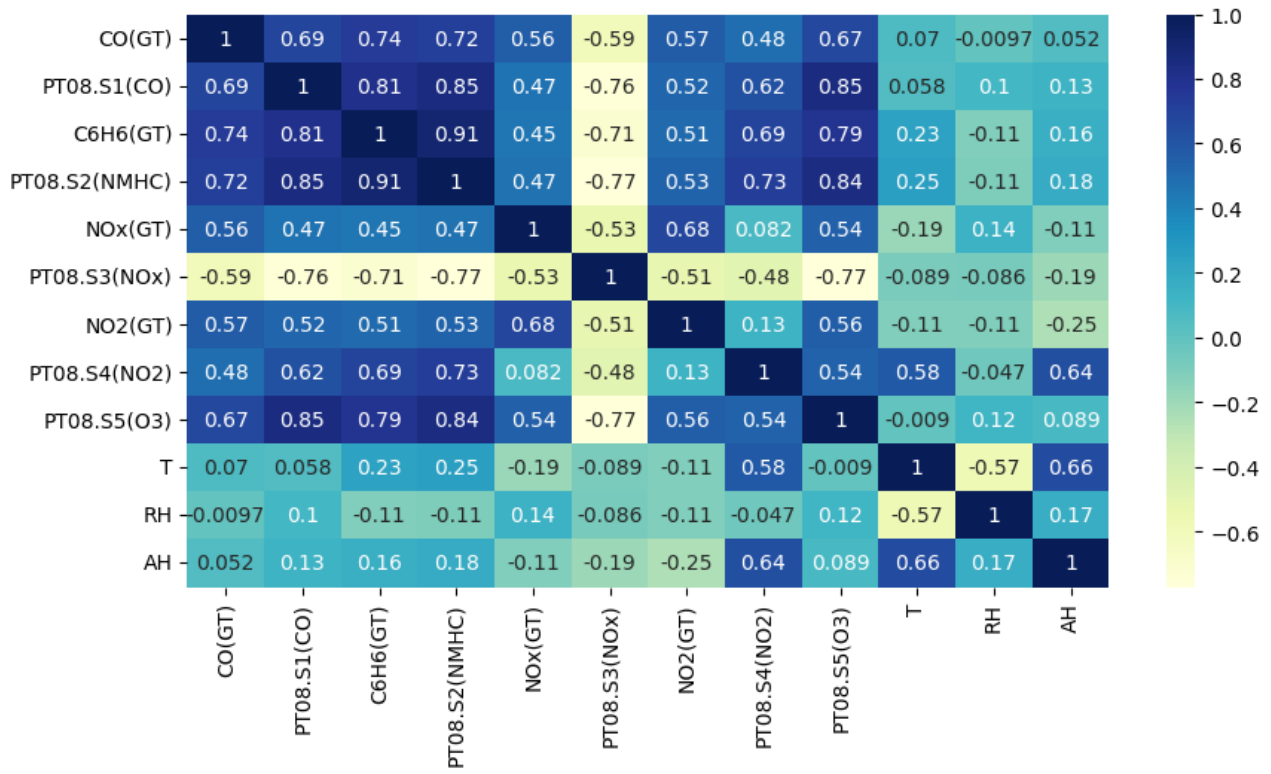

Рисунок 3.3 – Теплова карта кореляції

Відповідно до кореляції Пірсона, 0-ва кореляція означає відсутність зв'язку. Позитивна та негативна кореляція показує, що існує певний зв'язок.

Такі стовпчики, як T, RH, AH не мають сильної кореляції з іншими ознаками. NO2(GT) та NOx(GT) мають кореляцію з іншими ознаками, але не таку сильну, як CO(GT), C6H6(GT) та стовпчики з PT. CO(GT) і C6H6(GT) повинні бути колонками, які корелюють з усіма іншими характеристиками і повинні бути цільовими.

Далі будуємо та порівнюємо три моделі регресії: лінійна, Lasso і Ridge. Для кожної моделі розраховуються метрики якості, візуалізується важливість ознак і графік залежності предикт/ціль.

Кореляція використовується з двох основних причин:

1) Для візуалізації зв'язків між предикторами (незалежними змінними). Теплова карта кореляцій (df.corr()) дозволяє побачити, які ознаки сильно корелюють між собою, а які - ні. Це корисно для подальшого аналізу та побудови моделей;

2) Для відбору предикторів в моделі. Сильно корельовані між собою предиктори можуть викликати проблему мультиколінеарності. Тому корисно залишити з них тільки один предиктор, щоб уникнути перенавчання та перепідгонки моделі.

Наприклад, в нашому випадку було відібрано 8 предикторів з усього набору даних. Саме ті, які слабо корелюють між собою і добре корелюють з цільовою змінною.

Отже, аналіз кореляцій допомагає краще зрозуміти взаємозв'язки в даних і побудувати якісніше моделі машинного навчання. Це стандартний підхід в Data Science.

Моделі:

– LinearRegression - звичайна лінійна регресія, яка моделює залежність між цільовою змінною та предикторами за допомогою лінійного рівняння;
– Lasso - лінійна регресія з L1 регуляризацією, яка дозволяє відбирати значущі предиктори та уникати перенавчання моделі. Регуляризаційний параметр alpha встановлений рівним 0.1;
– Ridge - лінійна регресія з L2 регуляризацією, яка також допомагає уникнути перенавчання. Параметр регуляризації alpha також дорівнює 0.1.

Тепер наведемо приклад роботи моделей для двох основних показників, почнемо з C6H6(GT). На рисунках 3.4 – 3.6 наведено метрики якості, візуалізується важливість ознак і графік залежності предикт/ціль.

```
R2_score of Linear Regression: 0.8515116616599925
R2_score of Lasso Regression: 0.850955652815017
R2_score of Ridge Regression: 0.8515116900760945
```

Рисунок 3.4 – Метрики якості C6H6(GT)

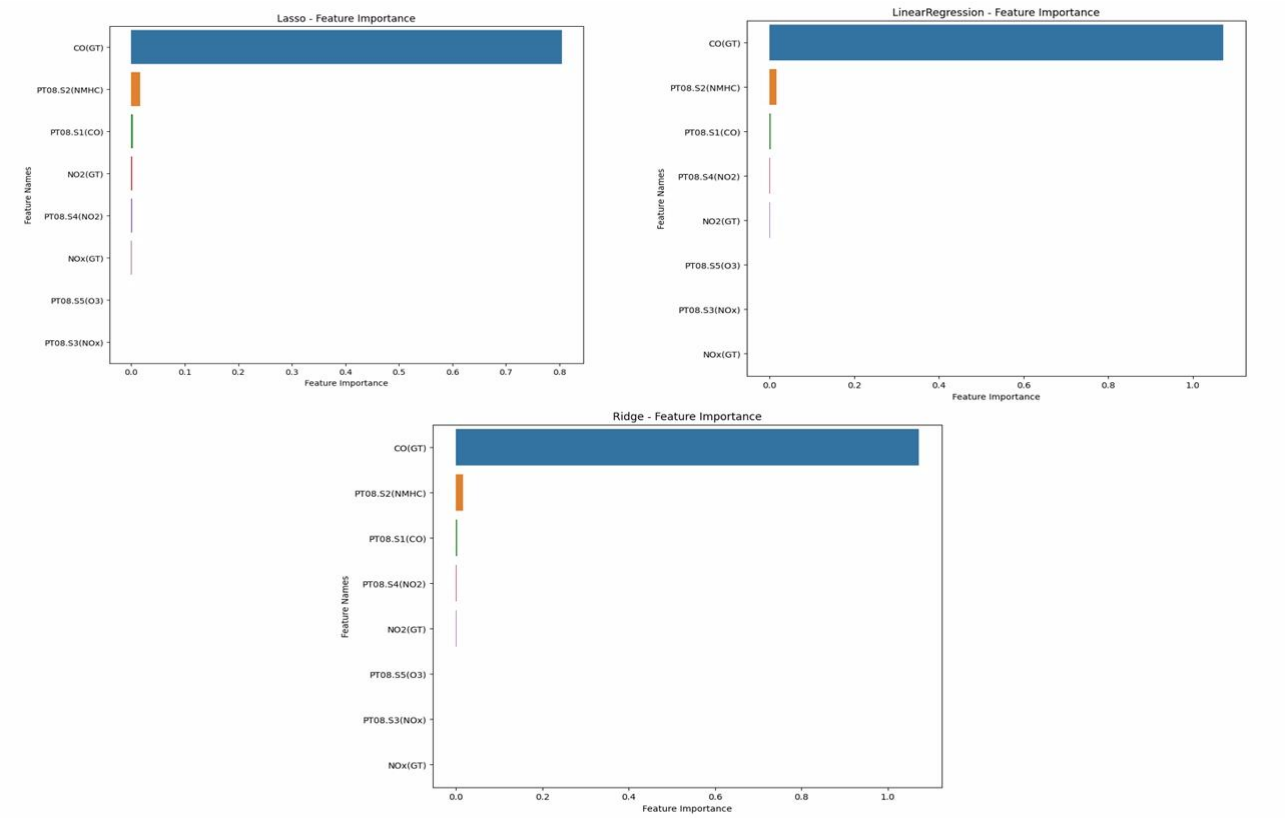

Рисунок 3.5 – Важливість ознак C6H6(GT)

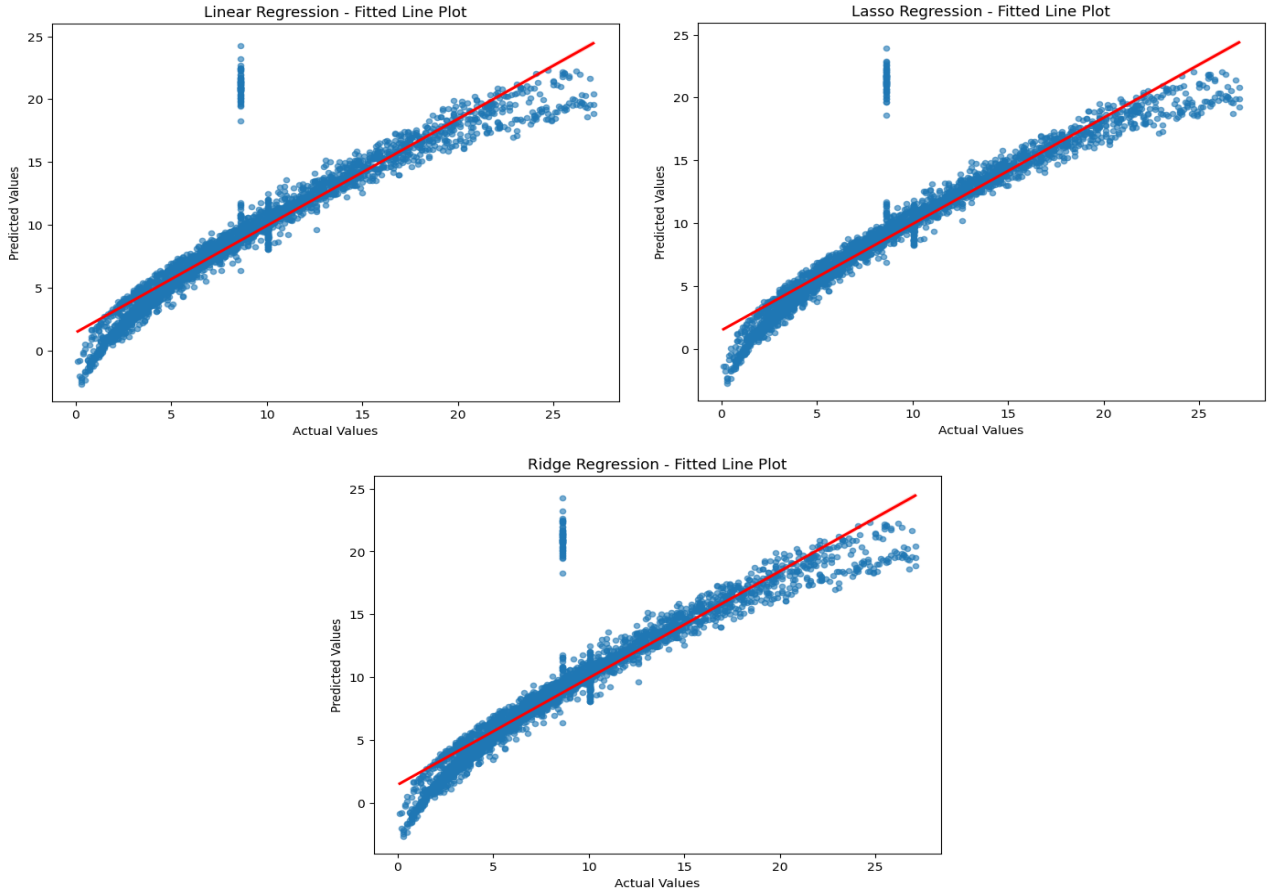

Рисунок 3.6 – Графік залежності предикт/ціль C6H6(GT)

Далі наведемо приклад роботи моделей для CO(GT). На рисунках 3.7 – 3.9 наведено метрики якості, візуалізується важливість ознак і графік залежності предикт/ціль.

```
R2_score of Linear Regression: 0.6368494897082215
R2_score of Lasso Regression: 0.6356957346228977
R2_score of Ridge Regression: 0.636849496505258
```

Рисунок 3.7 – Метрики якості CO(GT)

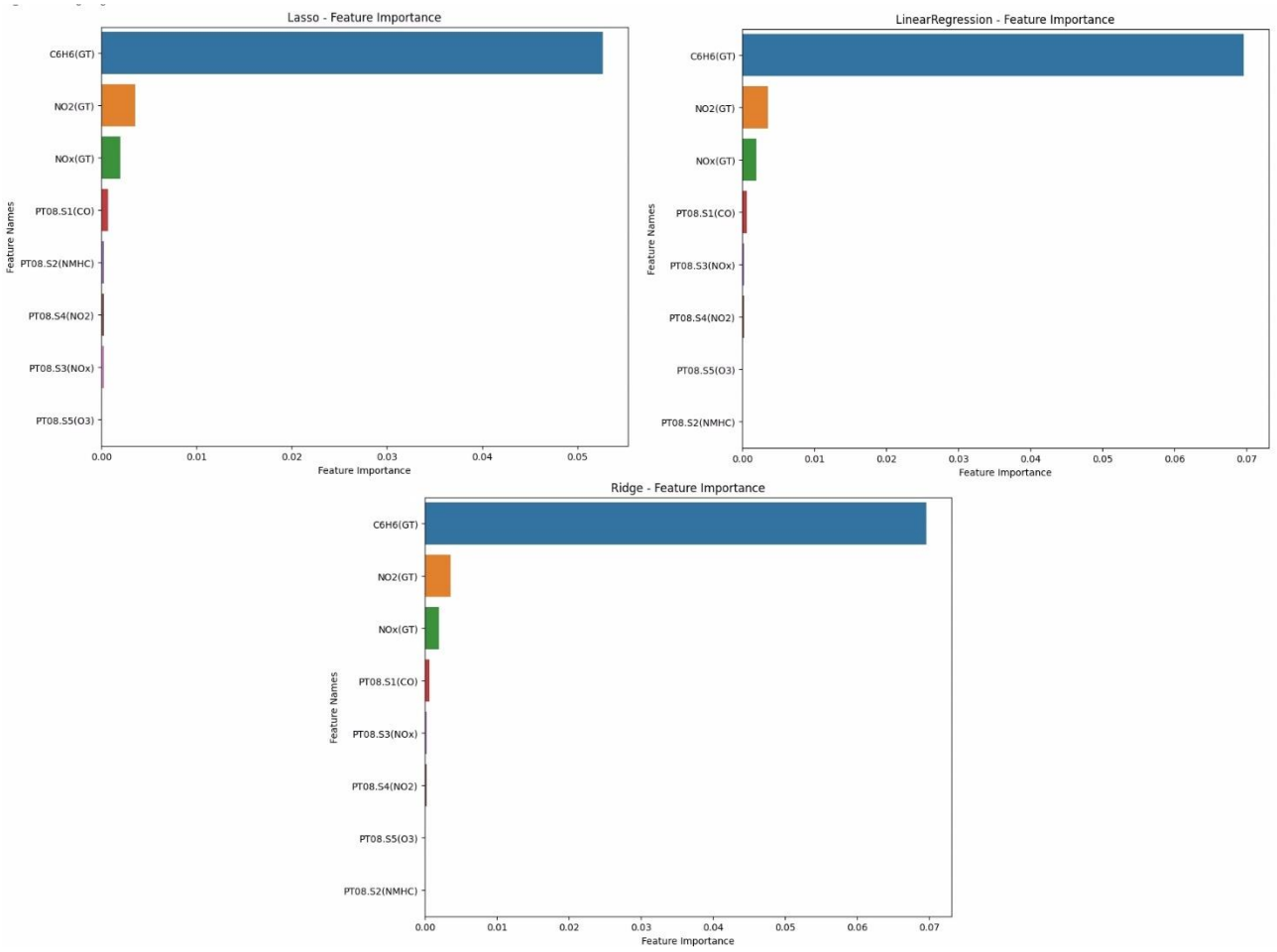

Рисунок 3.8 – Важливість ознак CO(GT)

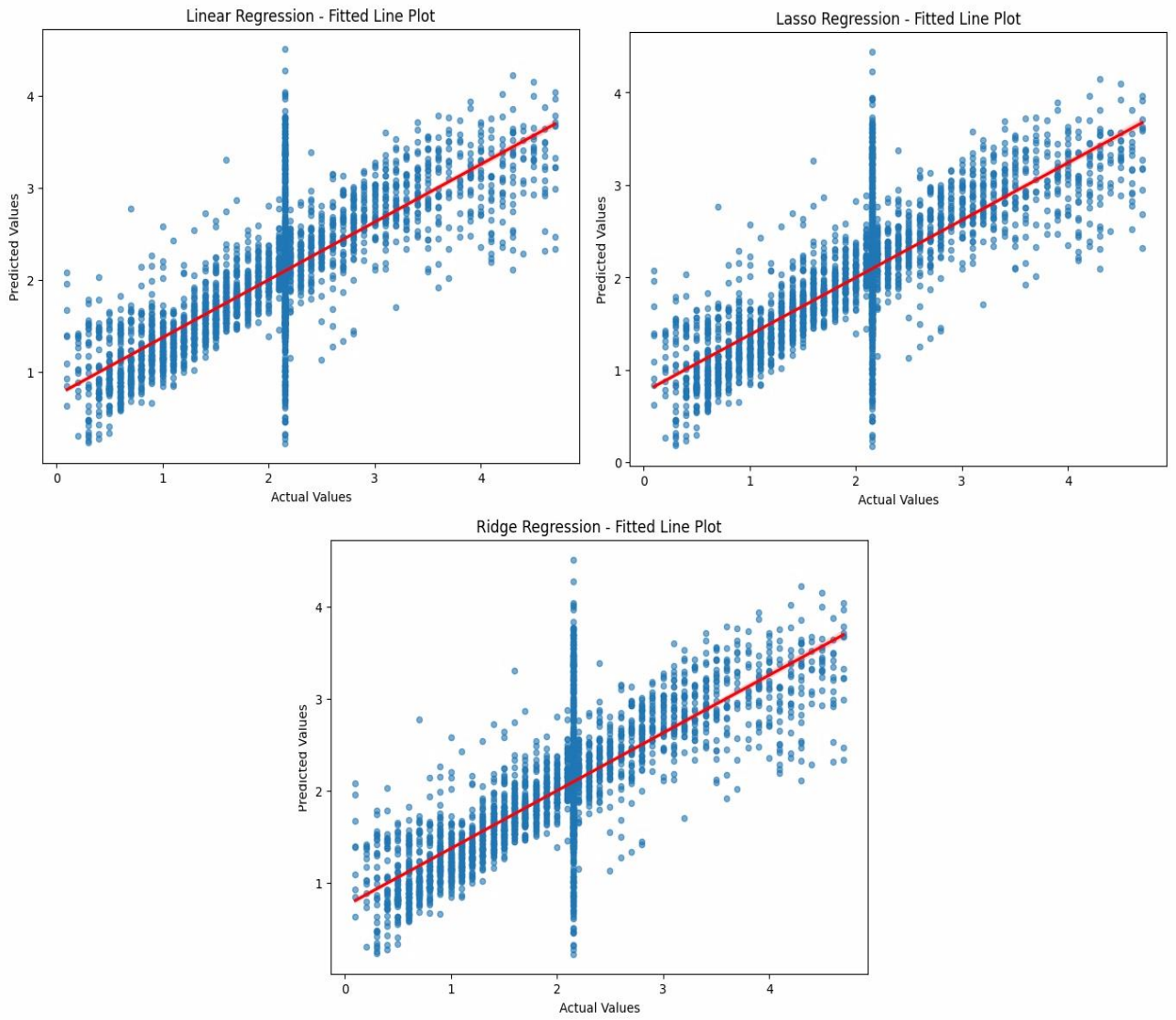

Рисунок 3.9 – Графік залежності предикт/ціль CO(GT)

Опираючись на результат, можна зробити висновки, що C6H6 може бути остаточно визначений як цільова змінна, оскільки r2score для CO(GT) як цільової змінної є меншим порівняно з r2_score для C6H6(GT).

Таким чином, лінійні моделі досить непогано описують залежності між показниками якості повітря. Їх можна застосувати для прогнозування рівня забруднення.

3.3 Late NYC Bus

У цьому розділі проаналізуємо запізнення громадського транспорту та виявимо проблемні ділянки маршрутів.

Завантажуємо необхідні бібліотеки та набір даних про транспорт Нью-Йорка. Далі досліджуємо дані, виправляємо деякі помилки та видаляємо дублікати.

Фільтруємо тільки ті рядки, де транспорт знаходиться на зупинці (DistanceFromStop = 0) (рис. 3.10).

tinationName	DestinationLat	DestinationLong	VehicleRef	VehicleLocation.Latitude	VehicleLocation.Longitude	NextStopPointName	ArrivalProximityText	DistanceFromStop	ExpectedArrivalTime	ScheduledArrivalTime
K AIRPORT via FARMERS BL	40.647137	-73.779427	NYCT_8107	40.647079	-73.779486	JFK AIRPORT/TERMINAL 5 AirTrain STATION	at stop	0	NaN	NaN
RIDGE SHORE RD	40.611717	-74.035072	NYCT_422	40.611669	-74.035166	SHORE RD/4 AV	at stop	0	NaN	NaN
EST SIDE BWAY - 106 ST	40.801422	-73.968239	NYCT_5848	40.801484	-73.9682	BROADWAY/W 106 ST	at stop	0	NaN	NaN
WASHINGTON TS GW BRIDGE	40.849033	-73.937309	NYCT_7751	40.848988	-73.937327	W 179 ST/BROADWAY	at stop	0	NaN	NaN
EST SIDE WEST END AV CROSSTOWN	40.795105	-73.973091	NYCT_6694	40.795177	-73.973036	W 96 ST/WEST END AV	at stop	0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
SELECT BUS SERVICE BAY RIDGE 86 ST STATN	40.62252	-74.028343	NYCT_4278	40.622546	-74.028453	4 AV/86 ST	at stop	0	NaN	NaN
PELHAM BAY STATION	40.852577	-73.827362	NYCT_8323	40.852585	-73.827303	BRUCKNER BL/BURR AV	at stop	0	NaN	NaN
EST SIDE BWAY - 106 ST	40.801422	-73.968239	NYCT_5843	40.801484	-73.9682	BROADWAY/W 106 ST	at stop	0	NaN	NaN
ATLANTIC AV BARCLAYS ENTER via 3 AV	40.683529	-73.979073	NYCT_761	40.68351	-73.979021	4 AV/DEAN ST	at stop	0	NaN	NaN
AMAICA 165 ST TERM	40.707615	-73.79554	NYCT_6486	40.707646	-73.795448	165 ST TERM/165 ST TERM BAY 6	at stop	0	NaN	NaN

Рисунок 3.10 – Фільтрація за DistanceFromStop

Необхідно перетворити деякі стовпці у потрібний формат дати/часу та після цього обраховати запізнення по відношенню до розкладу (scheduled - RecordedAtTime) (рис. 3.11).

	scheduled	RecordedAtTime	late	adherence
7	2017-06-01 00:08:00	2017-06-01 00:03:24	False	4.0
48	2017-06-01 00:19:00	2017-06-01 00:03:23	False	15.0
52	2017-06-01 00:00:10	2017-06-01 00:03:30	True	-4.0
53	2017-05-31 23:59:49	2017-06-01 00:03:49	True	-4.0
87	2017-05-31 23:51:05	2017-06-01 00:03:24	True	-13.0
...	...	...	...	...
6730721	2017-06-30 23:51:28	2017-06-30 23:53:32	True	-3.0
6730775	2017-06-30 23:46:12	2017-06-30 23:53:19	True	-8.0
6730783	2017-06-30 23:12:17	2017-06-30 23:53:31	True	-42.0
6730846	2017-06-30 23:47:40	2017-06-30 23:53:31	True	-6.0
6730853	2017-07-01 00:02:00	2017-06-30 23:53:13	False	8.0

Рисунок 3.11 – Scheduled to RecordedAtTime

За допомогою DBSCAN кластеризуються координати зупинок та будуємо графік розподілу запізнень по кластерам зупинок (рис. 3.12).

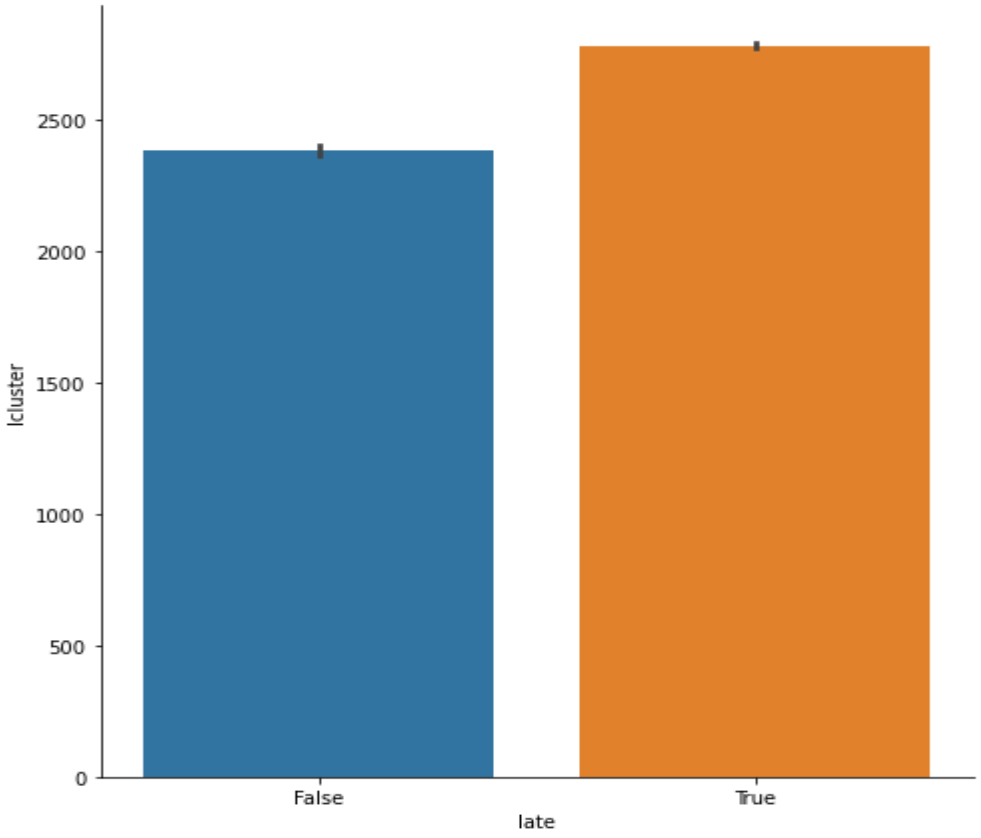

Рисунок 3.12 – Графік розподілу запізнень

Аналізуємо викиди, виявлені при кластеризації DBSCAN (рис. 3.13):

1) Спочатку фільтруємо рядки, які потрапили в шум/викиди (lcluster == -1);
2) Виділяємо стовпці, які цікавлять для подальшого аналізу викидів
3) За допомогою функції pairplot бібліотеки Seaborn будуємо матрицю scatter плотів для цих стовпців.

Кожен графік - це залежність між двома змінними. Діагональ - гістограми розподілів. Параметр hue='late' фарбує точки на графіках залежно від значення стовпця 'late', тобто чи рейс запізнився.

Така візуалізація дозволяє побачити взаємозв'язки між різними характеристиками для викидів, що може дати пояснення чому саме ці спостереження є викидами.

Отже, аналізується природа викидів серед зупинок, виявлених DBSCAN.

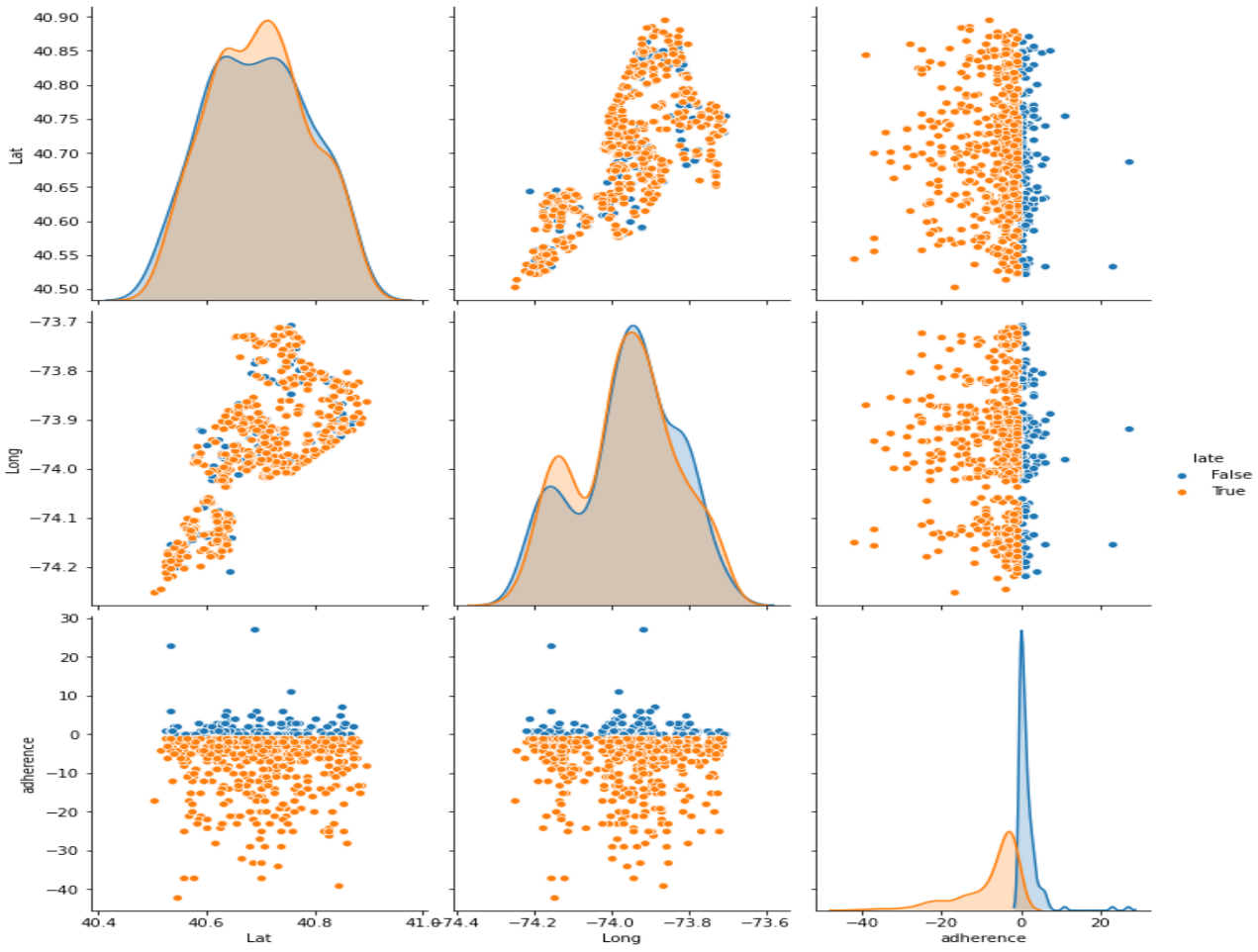

Рисунок 3.13 – Аналіз викидів DBSCAN

За допомогою KMeans кластеризуємо запізнення (adherence) за допомогою пошуку оптимального k та будуємо графіки розподілу запізнення по кластерам (рис. 3.14).

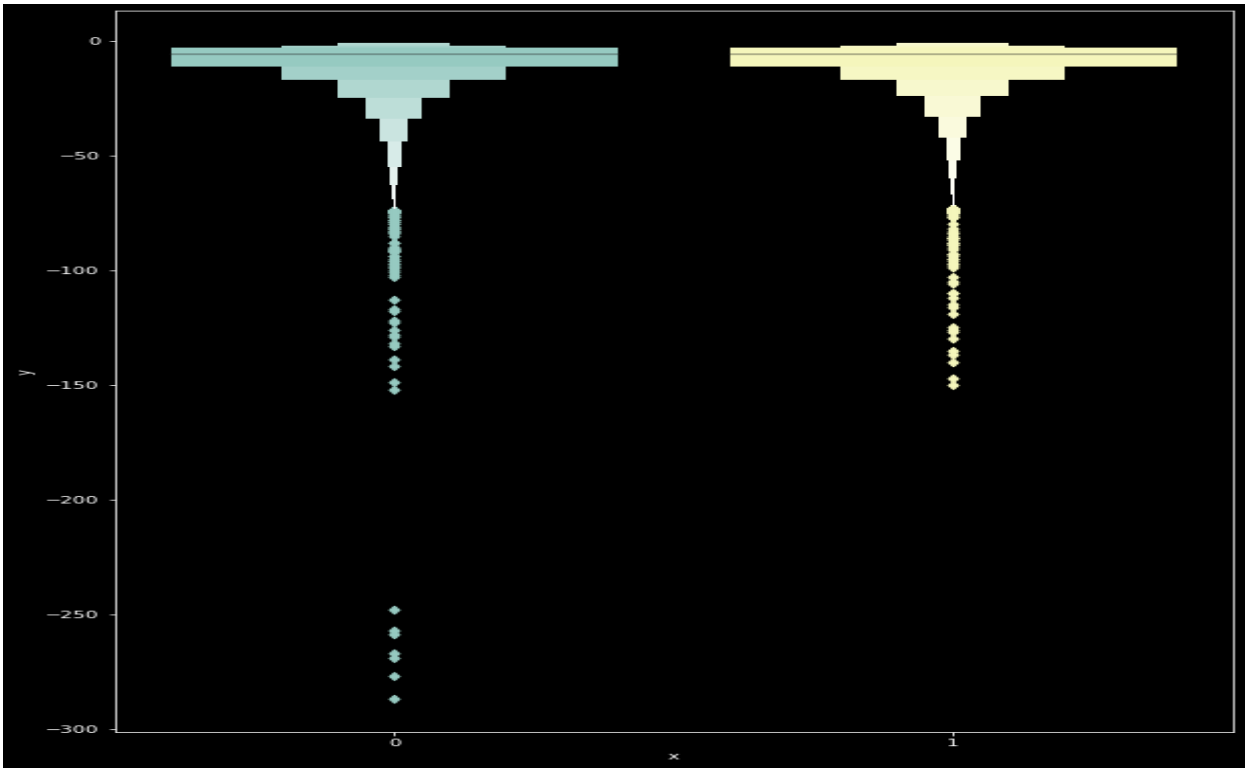

Рисунок 3.14 – Графіки розподілу запізнення по кластерам

Погляд на максимальні та мінімальні значення для кластерів показує, що дані розділені за датою запису – нулі пов'язані з записами, які були зроблені між 17 та 30 числами місяця, тоді як одиниці пов'язані з записами, які були зроблені між 1 та 16 числами місяця (рис. 3.14 – 3.15).

	RecordedAtTime	PublishedLineName	DestinationName	ScheduledArrivalTime	Lat	Long	RecordDate	sch
acluster								
0	2017-06-30 23:53:37	X9	YORKVILLE 91 ST via YORK AV	28:32:43	40.912385	-73.701417	2017-06-30	20 01 00
1	2017-06-16 05:18:46	X9	YORKVILLE 91 ST via YORK AV	28:34:00	40.912385	-73.701492	2017-06-16	20 16 07

Рисунок 3.15 – Максимальні значення кластеру

	RecordedAtTime	PublishedLineName	DestinationName	ScheduledArrivalTime	Lat	Long	RecordDate	sc
acluster								
0	2017-06-16 05:08:26	B1	108 ST H HRDNG EXY	00:00:00	40.503001	-74.252130	2017-06-16	2(1(0:
1	2017-06-01 00:03:21	B1	108 ST H HRDNG EXY	00:00:00	40.502962	-74.252130	2017-06-01	2(3 2:

Рисунок 3.16 – Мінімальні значення кластеру

Отже, кластерний аналіз дозволив виявити групи зупинок зі схожим рівнем запізнень, що може використовуватися для подальшого аналізу причин та оптимізації маршрутів.

3.4 San Francisco Crime Classification

В цьому розділі проаналізуємо дані про злочини у Сан-Франциско та побудуємо моделі машинного навчання для класифікації рішень по цих злочинах.

Завантажуємо бібліотеки та датасет злочинів (рис. 3.16).

	Dates	Category	Descript	DayOfWeek	PdDistrict	Resolution	Address	X	Y
0	2015-05-13 23:53:00	WARRANTS	WARRANT ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.425892	37.774599
1	2015-05-13 23:53:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	OAK ST / LAGUNA ST	-122.425892	37.774599
2	2015-05-13 23:33:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN	ARREST, BOOKED	VANNESS AV / GREENWICH ST	-122.424363	37.800414
3	2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN	NONE	1500 Block of LOMBARD ST	-122.426995	37.800873
4	2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	PARK	NONE	100 Block of BRODERICK ST	-122.438738	37.771541

Рисунок 3.17 – Датасет злочинів

Функція hist_vis будує гістограми по заданому стовпцю датасету. Використовується для візуалізації розподілів категорій злочинів, районів міста, видів вирішення справ тощо (рис. 3.17).

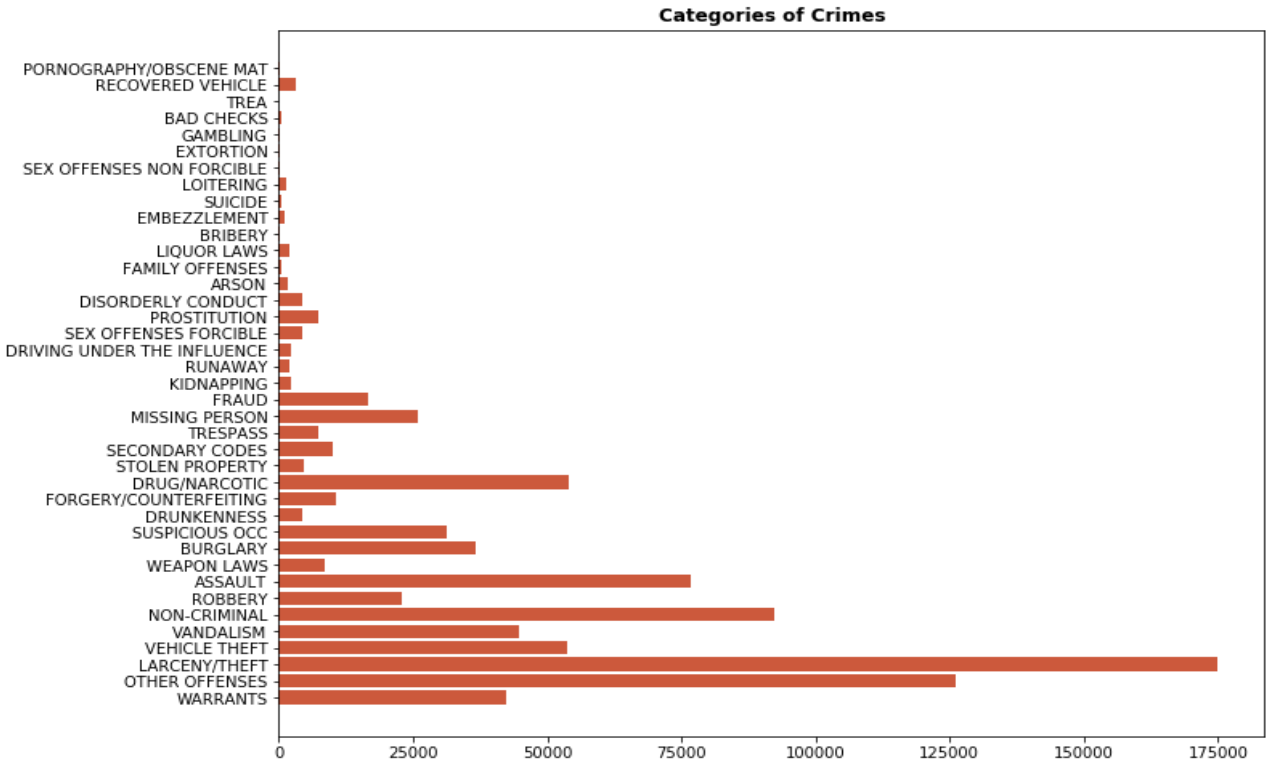

Рисунок 3.18 – Візуалізація розподілу категорії злочину

Функція visualize_desc аналізує окремі типи злочинів, виводить їх кількість, назву та гістограми характеристик (день тижня, тип вирішення) (рис. 3.18).

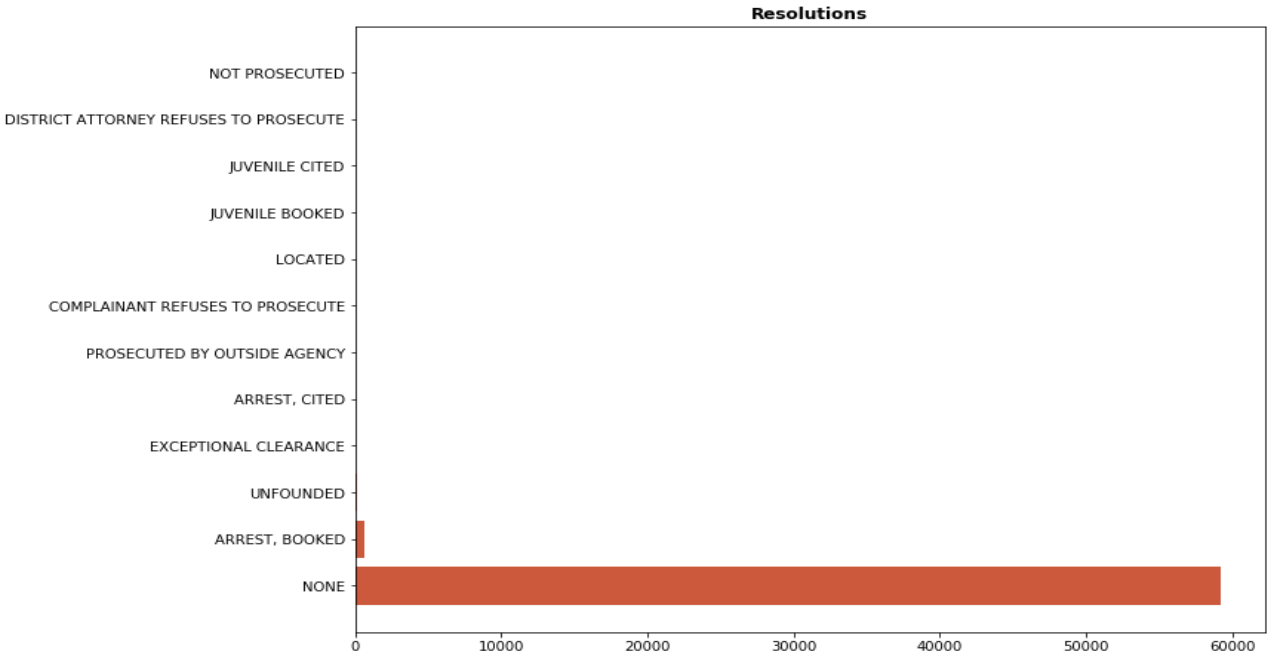

Рисунок 3.19 – Візуалізація типу вирішення для крадіжки авто

За допомогою цих функцій візуалізуються та аналізуються злочини в Сан-Франциско.

Далі необхідно очистити дані, а саме: видалити адреси поліції та деякі стовпці.

Будуємо моделі класифікації типу вирішення злочинів: DecisionTreeClassifier та кілька RandomForestClassifier з різними гіперпараметрами. Для них розраховуємо метрики: accuracy, f1 score, час навчання.

Будуємо графік порівняння результатів (рис. 3.19).

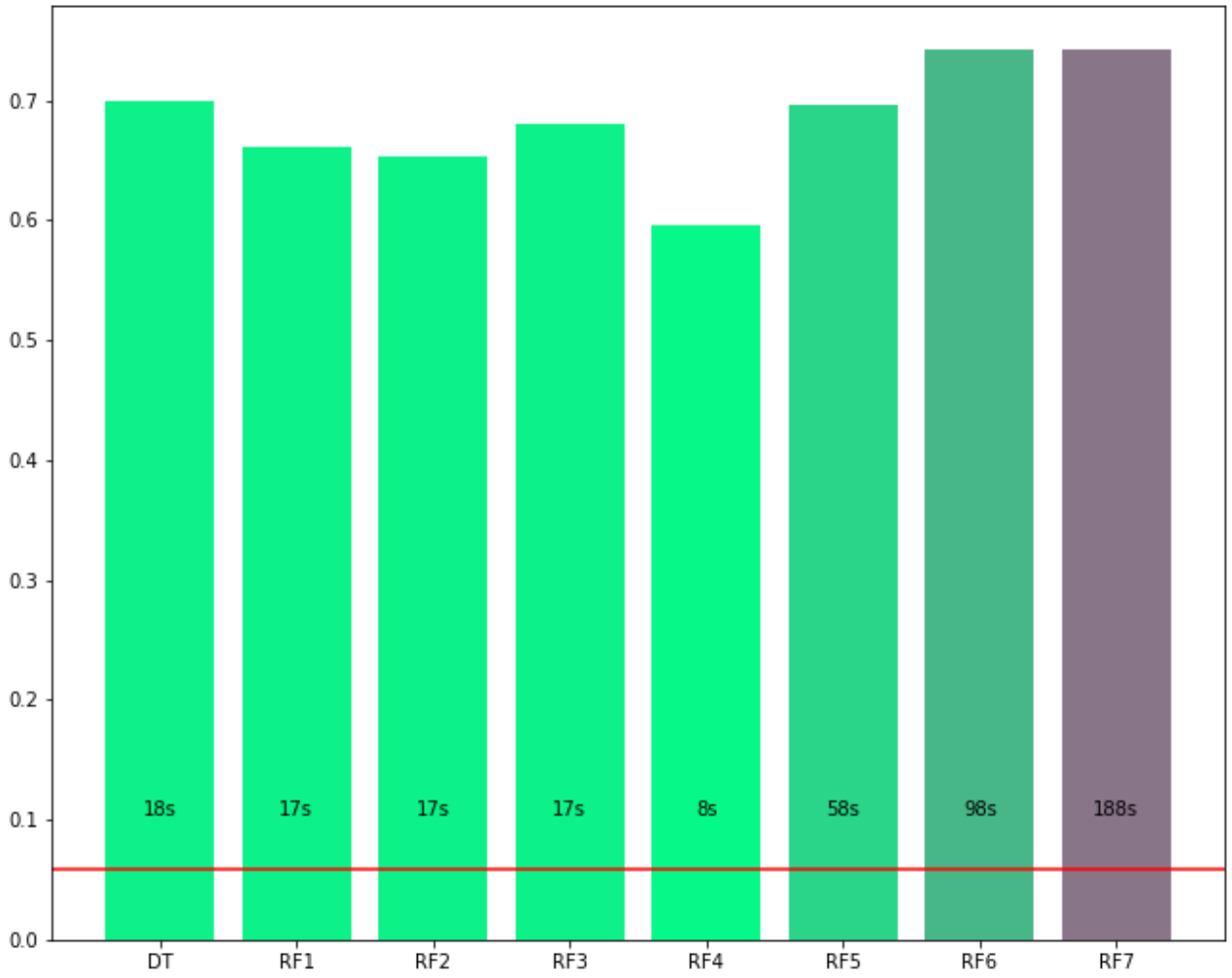

Рисунок 3.20 – Графік порівняння результатів

Моделі:

1) DecisionTreeClassifier – дерево прийняття рішень. Дозволяє будувати ієрархічну структуру правил для класифікації.
2) RandomForestClassifier – ансамбль дерев прийняття рішень. Комбінує прогнози від випадкового набору дерев для кращої якості та уникнення перенавчання.

Метрики:

1) Accuracy - частка правильних відповідей моделі;
2) F1-score - усереднена метрика, що враховує точність і повноту;
3) Час навчання моделі.

Результати:

– Найкращий F1-score (~0.7) показали RandomForestClassifier;
– Більш складні моделі з більшою кількістю дерев та глибиною працюють краще;
– Але вони вимагають більше часу на навчання (до 190 с);

Отже, код аналізує дані про злочинність та тестує моделі для передбачення результату розслідування злочину. Це може бути корисно для оптимізації роботи поліції.

ВИСНОВКИ

В результаті роботи було були досліджені методи машиного навчання в технологіях smart city. Для дослідження було обрано платформу Kaggle та 3 датасети: Air Quality Dataset, New York City Bus Data, San Francisco Crime Classification.

У відповідності з метою кваліфікаційної роботи були досліджені методи машинного навчання в технологіях smart city із застосуванням наступних технологій:

– Kaggle для реалізації та тестування різних підходів та моделей.

У відповідності з поставленими задачами були виконані наступні етапи створення системи:

– оглянуто літературу, дотичну до теми, а також написано рецензії та визначено основні тентенції та перспективи розвитку машиного навчання в цій галузі;
– розглянути зв’язок машинного навачання зі smart city, надано схеми та приклади взаємодії;
– дослідили та порівняли різні підходи машинного навчання щодо обробки та аналізу даних процесів смарт-сіті.

ПЕРЕЛІК ПОСИЛАНЬ

1. Auffarth B. Machine Learning for Time Series: Use Python to forecast, predict, and detect anomalies with state-of-the-art machine learning methods. Birmingham : Packt Publishing, 2024. 392 p.
2. Cyberattacks Detection in IoT-Based Smart City Applications Using Machine Learning Techniques. URL: https://www.mdpi.com/1660-4601/17/24/9347 (дата звернення: 01.10.2023).
3. FlexLondon. URL: https://www.london.gov.uk/programmes-and-strategies/environment-and-climate-change/energy/flexlondon (дата звернення: 22.09.2023).
4. Gupta S.K, Karras D.A., Khang A., Rani S. mart Cities: IoT Technologies, Big Data Solutions, Cloud Platforms, and Cybersecurity Techniques. : CRC Press, 2023. 428 p.
5. IoT for Smart Cities: Machine Learning Approaches in Smart Healthcare—A Review. URL: https://www.mdpi.com/1999-5903/13/8/218 (дата звернення: 01.10.2023).
6. Liu Y. Python Machine Learning By Example: Unlock machine learning best practices with real-world use cases. Birmingham : Packt Publishing, 2024. 456 p.
7. Machine learning based system for managing energy efficiency of public sector as an approach towards smart cities. URL: https://www.sciencedirect.com/science/article/pii/S0268401219302968 (дата звернення: 01.10.2023).
8. Machine Learning in Wireless Sensor Networks for Smart Cities: A Survey. URL: https://www.mdpi.com/2079-9292/10/9/1012 (дата звернення: 01.10.2023).
9. Marchesani A. The Global Smart City: Challenges and Opportunities in the Digital Age. : Emerald Publishing Limited, 2023. 252 p.

10. Petrou T. Master Data Analysis with Python. : No Starch Press, 2024. 504 p.

11. Optimising Water Resource Management: Smart Water Solutions and Success in Barcelona. URL: https://medium.com/mark-and-focus/optimising-water-resource-management-smart-water-solutions-and-success-in-barcelona-637611941b0d (дата звернення: 22.09.2023).

12. Waste Less, Recycle More. URL: https://www.epa.nsw.gov.au/your-environment/recycling-and-reuse/waste-less-recycle-more (дата звернення: 22.09.2023).

ДОДАТОК А

Air Quality

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score,accuracy_score
import warnings
warnings.filterwarnings('ignore')

# reading the dataset
# since the dataset has been delimited by ';', we need to change the delimiter to ';' instead ',' and decimals are represented by commas
df = pd.read_csv('/kaggle/input/air-quality-data-set/AirQuality.csv',sep=";", decimal=",",header=0)
df

# shape of our dataset
# df.shape
print("No of rows in dataset:",df.shape[0])
print("No of columns in dataset:",df.shape[1])

# getting the names of the columns
df.columns

# getting the dtypes of the all columns
df.dtypes

# getting the numerical estimates of all the numerical column
df.describe()

df.info()

# checking null values in our dataset
```

```
df.isna().sum()

# visualizing the na values using heatmap
sns.heatmap(df.isna(),yticklabels=False,cmap='crest')
plt.show()

df.tail()

# dropping the last two columns
# assigning inplace argument as True which we result in completly dropping off of the last two columns
df.drop(columns=['Unnamed: 15', 'Unnamed: 16'],inplace=True)

# remaining columns
df.columns

# dropping the last 114 rows which are fully empty
df.dropna(inplace=True)

# no null values are now present in the dataset which was provided
sns.heatmap(df.isna(),yticklabels=False,cmap='crest')
plt.show()

#first labelling -200 value as null value
df.replace(to_replace=-200,value=np.nan,inplace=True)

sns.heatmap(df.isna(),yticklabels=False,cmap='crest')
plt.show()

df.drop(columns=['NMHC(GT)'],inplace=True)

df.isna().sum()

df.columns

# getting the datatypes of all the columns having null values
# first storing all the column names in a list having null values
col = ['CO(GT)', 'PT08.S1(CO)', 'C6H6(GT)','PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)',
'PT08.S4(NO2)','PT08.S5(O3)', 'T', 'RH', 'AH']
df = df[col]
df[col].dtypes

df[col].head()
```

```
for i in col:
    df[i] = df[i].fillna(df[i].mean())

df.isna().sum()

# plotting a boxplot
plt.figure(figsize=(5,5))
sns.boxplot(data=df)
plt.xticks(rotation='vertical')
plt.show()

# getting the quartile one and quartile 3 values of each column
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
# finally calculating the interquartile range IQR
IQR = Q3 - Q1

# if the values fall behind Q1 - (1.5 * IQR) or above Q3 + 1.5*IQR,
#then it is been defined as outlier
((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).sum()

mask = (df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))
mask

# now replacing all the outliers using the median of that particular column
for i in mask.columns:
    df[i].astype('float')
    temp = df[i].median()
    df.loc[mask[i], i] = temp

# outliers are now being handled and are replaced with that column's median value
((df[col] < (Q1 - 1.5 * IQR)) | (df[col] > (Q3 + 1.5 * IQR))).sum()

plt.figure(figsize=(5,5))
sns.boxplot(data=df)
plt.xticks(rotation='vertical')
plt.show()

df.head(20)

df.dtypes
```

```
# using pearson's correlation to find the correlation between all the features
df.corr()

plt.figure(figsize=(10,5))
sns.heatmap(df.corr(),cmap='YlGnBu',annot=True)
plt.show()

def plot_feature_importance(model, feature_names):
    importance = model.coef_ if hasattr(model, 'coef_') else model.feature_importances_
    feature_names = np.array(feature_names)
    data = {'feature_names': feature_names, 'feature_importance': importance}
    fi_df = pd.DataFrame(data)
    fi_df.sort_values(by=['feature_importance'], ascending=False, inplace=True)

    plt.figure(figsize=(10, 8))
    sns.barplot(x=fi_df['feature_importance'], y=fi_df['feature_names'])
    plt.title(model.__class__.__name__ + ' - Feature Importance')
    plt.xlabel('Feature Importance')
    plt.ylabel('Feature Names')

def fitted_line_plot(y_true, y_pred, model_name):
    plt.figure(figsize=(8, 6))
    sns.regplot(x=y_true, y=y_pred, scatter_kws={'s': 20, 'alpha': 0.6}, line_kws={'color': 'red'})
    plt.title(f'{model_name} - Fitted Line Plot')
    plt.xlabel('Actual Values')
    plt.ylabel('Predicted Values')
    plt.show()

def solve(X,y):

    X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state=1)
    # using linear regression
    model = LinearRegression()
    model.fit(X_train,y_train)
    lr_pred = model.predict(X_test)
    print("R2_score of Linear Regression:",r2_score(y_test,lr_pred))
    plot_feature_importance(model, X.columns)
    fitted_line_plot(y_test, lr_pred, 'Linear Regression')

    model = Lasso(alpha=0.1)
    model.fit(X_train,y_train)
```

```
    ls_pred = model.predict(X_test)
    print("R2_score of Lasso Regression:",r2_score(y_test,ls_pred))
    plot_feature_importance(model, X.columns)
    fitted_line_plot(y_test, ls_pred, 'Lasso Regression')

    model = Ridge(alpha=0.1)
    model.fit(X_train,y_train)
    r_pred = model.predict(X_test)
    print("R2_score of Ridge Regression:",r2_score(y_test,r_pred))
    plot_feature_importance(model, X.columns)
    fitted_line_plot(y_test, r_pred, 'Ridge Regression')

# choosing C6H6 as the target variable and other float dtype features as independed features
X = df[['CO(GT)', 'PT08.S1(CO)', 'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4(NO2)','PT08.S5(O3)']]
y = df['C6H6(GT)']

solve(X,y)

# now selecting CO(GT) as the target variables and other float dtype features as independent features
X = df[['C6H6(GT)','PT08.S1(CO)', 'PT08.S2(NMHC)', 'NOx(GT)', 'PT08.S3(NOx)', 'NO2(GT)', 'PT08.S4(NO2)','PT08.S5(O3)']]
y = df['CO(GT)']

solve(X,y)
```

ДОДАТОК Б

Late NYC Bus

```
import datetime #date time
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

#Graphs, visualizations, etc.
import seaborn as sns
import matplotlib.pyplot as plt

#Unsupervised learning metrics and clustering.
from sklearn.cluster import DBSCAN, KMeans
from sklearn import metrics

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

names = range(0, 19)
transit = pd.read_csv("/kaggle/input/new-york-city-transport-statistics/mta_1706.csv",header=None,names=names,
low_memory=False)
transit.shape

#Investigate the last two columns.
transit[[17, 18]].describe(include='all')



#drop the null column.
transit.drop(columns=18, inplace=True)

#I need headers.
headers = transit.iloc[0]
#Get rid of those nulls.
headers = headers.fillna(value='unknown')
#Rename the headers.
transit = transit.rename(columns=headers)
```

```
transit.drop(transit.index[0], inplace=True)
transit

problems = transit.loc[~transit['unknown'].isna()]

problems.iloc[:,-8:]

#Arrival Proximity.
transit["ArrivalProximityText"].loc[~transit['unknown'].isna()] = problems['DistanceFromStop']

#DistanceFromStop
transit["DistanceFromStop"].loc[~transit['unknown'].isna()] = problems['ExpectedArrivalTime']

#ExpectedArrivalTime
transit["ExpectedArrivalTime"].loc[~transit['unknown'].isna()] = problems['ScheduledArrivalTime']

#ScheduledArrivalTime
transit["ScheduledArrivalTime"].loc[~transit['unknown'].isna()] = problems['unknown']

#drop that unknown column
transit.drop(columns='unknown', inplace=True)

#A summary of missing variables represented as a percentage of the total missing content.
def missingness_summary(df, print_log=False, sort='ascending'):
    s = df.isnull().sum()*100/df.isnull().count()

    if sort.lower() == 'ascending':
        s = s.sort_values(ascending=True)
    elif sort.lower() == 'descending':
        s = s.sort_values(ascending=False)
    if print_log:
        print(s)

  return pd.Series(s)

  missingness_summary(transit)

  #Drop the duplicates, inclusive of all but the scheduled arrival time (sometimes there's more than one.)
subset = transit.columns[0:16]
transit.drop_duplicates(subset=subset, inplace=True)
```

```
#get the rows and columns of interest.
coi = ['RecordedAtTime','PublishedLineName', 'DestinationName','VehicleLocation.Latitude',
'VehicleLocation.Longitude','DistanceFromStop','ScheduledArrivalTime']
atthestop = transit[coi].loc[transit.DistanceFromStop == '0'].copy()
atthestop.info()

#quick look at what's going on with the scheduled arrival time, null values.
transit[(transit.DistanceFromStop == '0') & (transit.ScheduledArrivalTime.isna())]

#I can't disern anything. There's plenty of information. Drop these.
atthestop.dropna(inplace=True)

#We don't need DistanceFromStop anymore
atthestop.drop(columns='DistanceFromStop', inplace=True)

atthestop

#Convert the lat and long to numeric.
atthestop['Lat'] = atthestop['VehicleLocation.Latitude'].apply(pd.to_numeric, errors='coerce')
atthestop['Long'] = atthestop['VehicleLocation.Longitude'].apply(pd.to_numeric, errors='coerce')
atthestop.drop(columns=['VehicleLocation.Latitude', 'VehicleLocation.Longitude'], inplace=True)
atthestop

#Recorded at time converted to date time.
atthestop['RecordedAtTime'] = pd.to_datetime(atthestop['RecordedAtTime'], errors = 'coerce')
tindex = pd.DatetimeIndex(atthestop.RecordedAtTime)
#Recorded Year, Day, and Month as a separate column for convenience
atthestop['RecordDate'] = tindex.strftime('%Y-%m-%d ')

#Troubleshooting scheduled arrival time.
sTimes = atthestop[['ScheduledArrivalTime', 'RecordDate']].loc[atthestop.ScheduledArrivalTime >= "24"]

#year-month-day as string
ymd = sTimes.RecordDate
hours = sTimes.ScheduledArrivalTime.str[0:2]
rest = sTimes.ScheduledArrivalTime.str[2:]
# subtract 24
hours = hours.astype(int) - 24
# reassign the string
converted = hours.astype(str) + rest
converted = ymd + converted
# make it datetime, adding one to the day because it's after midnight.
```

```
converted = pd.to_datetime(converted, format = '%Y-%m-%d %H:%M:%S')
converted = converted

#scheduled column as Scheduled arrival time converted to date time.
atthestop['scheduled'] = pd.to_datetime(atthestop['RecordDate'] + atthestop['ScheduledArrivalTime'], format = '%Y-%m-%d %H:%M:%S', errors='coerce')
atthestop['scheduled'].fillna(value=converted, inplace=True)

#correct for situations after midnight.
condition = (atthestop.scheduled - atthestop.RecordedAtTime).astype('timedelta64[m]') >= 1000
atthestop.loc[condition, ['scheduled']] = atthestop.loc[condition, ['scheduled']] - pd.Timedelta(days=1)

condition = (atthestop.scheduled - atthestop.RecordedAtTime).astype('timedelta64[m]') <= -1000
atthestop.loc[condition, ['scheduled']] = atthestop.loc[condition, ['scheduled']] + pd.Timedelta(days=1)

#late indicator
condition = (atthestop.scheduled - atthestop.RecordedAtTime).astype('timedelta64[m]') < 0
atthestop['late'] = condition

#adherence
atthestop['adherence'] = (atthestop.scheduled - atthestop.RecordedAtTime).astype('timedelta64[m]')

#Look at it
atthestop[['scheduled', 'RecordedAtTime', 'late', 'adherence']]

#stop coordinates.
coord = atthestop[['Lat', 'Long']]



#DB Scan location Clusters
lcluster = DBSCAN(eps=0.00003, min_samples=2).fit_predict(coord)

#Assign the clusters to the stop table.
atthestop['lcluster'] = lcluster

print( metrics.silhouette_score(coord, lcluster))

#how do I want to plot over one thousand clusters.
g = sns.catplot(y="lcluster", x='late',
                data=atthestop, kind="bar",
                height=7, aspect=1)
```

```
#Have a look at the outliers.
outliers = atthestop[atthestop.lcluster == -1]
clmns = ['RecordedAtTime','ScheduledArrivalTime', 'Lat', 'Long', 'RecordDate', 'scheduled',
      'late', 'adherence']
ax = sns.pairplot(data=outliers[clmns], hue='late', aspect=.8, height=4)
plt.show()

#Scatter plot of recorded time against the scheduled time.
s = atthestop.copy()
s['x'] = pd.to_datetime(s['RecordedAtTime']).dt.strftime('%H:%M')
s['y'] = pd.to_datetime(s['scheduled']).dt.strftime('%H:%M')

sns.scatterplot(x='x', y='y', data=s)
plt.show()

#k-means cluster using pca.
scores = []
K = range(2,11)
goal = 0

#stop adherence.
adherence = atthestop[['scheduled', 'RecordedAtTime']]



print("KMeans Cluster Search \n Start time: {}".format(datetime.datetime.now()))
print("**********************************************************************************************")
for k in K:

  #Building and fitting the model, date time and labels for knowing when a first cluster and silhouette score has completed.
  print("Building cluster {} at {}".format(k, datetime.datetime.now()))
  sample_pred = KMeans(n_clusters=k).fit_predict(adherence)

  print("scoring the model at: {}".format(datetime.datetime.now()))
  score = metrics.silhouette_score(adherence, sample_pred)
  #track for the max score.
  if score > goal:
    goal = score
    cluster = k
    print("best score so far is {} with a k of {}".format(goal, k))
  scores.append(score)
  print("-----")
print("**********************************************************************************************")
```

```
plt.style.use(['dark_background'])
plt.subplots(figsize=(15, 15))
plt.plot(K, scores, 'bx-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette Score')
plt.title('The Silhouette Plot\n Best score is {:0.2f} at k = {}'.format(goal, cluster))
plt.show()


#Adherence Clusters
kadherence =  KMeans(n_clusters=cluster)
acluster = kadherence.fit_predict(adherence)
#Assign the clusters to the stop table.
atthestop['acluster'] = acluster

g = sns.catplot(x="acluster", y='adherence', col="late",
                data=atthestop, kind="boxen",
                height=10, aspect=.8)

#Plot against the location and adherence clusters.
s = atthestop[atthestop.late == True].copy()
s['x'] = atthestop.acluster
s['y'] = atthestop.adherence


fig, ax = plt.subplots(figsize=(10,14))
ax = sns.boxenplot(x='x', y='y', data=s)
plt.show()

m = atthestop.groupby(by='acluster').max()
m.style.background_gradient()

m = atthestop.groupby(by='acluster').min()
m.style.background_gradient()
```

ДОДАТОК В

San Francisco Crime Classification

```
import pandas as pd  # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib as plt
import numpy as np  # linear algebra
import time
from collections import Counter
import matplotlib.pyplot as plot
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score
%matplotlib inline

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input directory

import os

# Any results you write to the current directory are saved as output.

df = pd.read_csv('../input/train.csv')
df.head()

def hist_vis(df, title):
    ctrdict = dict(Counter(df))
    labels, values = ctrdict.keys(), ctrdict.values()
    fig, ax = plot.subplots(figsize=(11, 9))
    plot.title(title, weight='bold')
    plot.barh(range(len(labels)),list(values), color='#cc593c')
    plot.yticks(range(len(labels)), labels)
    plot.plot()

def visualize_desc(desc):
    retdf = df[df["Descript"] == desc]
```

```
    if len(retdf) < 1:
        print ("No incidents found for " + desc)
        return 0
    else:
        print(len(retdf))
        print(desc)

        hist_vis(retdf["Resolution"], "Resolutions")
        hist_vis(retdf["DayOfWeek"], "Day of the Week")

def hplot_vis(category, value, title):
    fig, ax = plot.subplots(figsize=(11, 9))
    plot.title(title, weight='bold')
    ax.barh(category,value, color='#34b180')

    plot.show()
    print(len(category))

df.Category.unique()

hist_vis(df.Category, "Categories of Crimes")

len(df[df.Category=="OTHER OFFENSES"])

hist_vis(dict(Counter(df[df.Category=="OTHER OFFENSES"].Descript).most_common(30)), "Top 30 'OTHER' crimes")

hist_vis(dict(Counter(df.Descript).most_common(30)), "Top 30 of All Crimes")

df["PdDistrict"].unique()
hist_vis(dict(Counter(df["PdDistrict"]).most_common()), "Crimes by District")

hist_vis(dict(Counter(df[df["PdDistrict"]=="SOUTHERN"].Descript).most_common(25)), "Top 25 Crimes in the Southern District")

hist_vis(dict(Counter(df[df["PdDistrict"]=="RICHMOND"].Descript).most_common(25)), "Top 25 Crimes in the Richmond District")

hist_vis(dict(Counter(df[df["PdDistrict"]=="TENDERLOIN"].Descript).most_common(25)), "Top 25 Crimes in the Tenderloin")
```

```
hist_vis(dict(Counter(df[df["PdDistrict"]=="NORTHERN"].Descript).most_common(25)), "Top 25 Crimes in the Northern District")

visualize_desc("GRAND THEFT FROM LOCKED AUTO")

visualize_desc("STOLEN AUTOMOBILE")

visualize_desc("PLACING WIFE IN HOUSE OF PROSTITUTION")

visualize_desc("PLACING HUSBAND IN HOUSE OF PROSTITUTION")

visualize_desc("DANGER OF LEADING IMMORAL LIFE")

visualize_desc("MAYHEM WITH A DEADLY WEAPON")

visualize_desc("FORTUNE TELLING")

visualize_desc("ASSAULT, AGGRAVATED, W/ MACHINE GUN")

visualize_desc("DESTRUCTION OF PROPERTY WITH EXPLOSIVES")

df['Resolution'].unique()

hist_vis(dict(Counter(df['Address']).most_common(25)), "Top 25 Most Common Addresses")
len(Counter(df["Address"]))

hist_vis(dict(Counter(df[df["Address"] == '800 Block of BRYANT ST'].X).most_common(5)), "braynta")
hist_vis(dict(Counter(df[df["Address"] == '800 Block of BRYANT ST'].Y).most_common(5)), "braynta")

hist_vis(dict(Counter(df[df["Address"] == '800 Block of MARKET ST'].X).most_common(25)), "braynta")
hist_vis(dict(Counter(df[df["Address"] == '800 Block of MARKET ST'].Y).most_common(25)), "braynta")

#Clean the data

#800 Bryant street is the police station... What's going on here?
#Let's remove everythign with this address
df_list = df.index[df["Address"] == "800 Block of BRYANT ST"].tolist()
cleaned_df = df.drop(df.index[df_list])
cleaned_df[cleaned_df["Address"] == "800 Block of BRYANT ST"]

cleaned_df.drop(["Dates", "Address", "Descript"], axis = 1, inplace=True)
```

```
df.head()

print(len(cleaned_df.Resolution))
hist_vis(dict(Counter(cleaned_df.Resolution)), "y")

%%time
classfs = []
accuracy = []
fscore = []
time_ls = []


cats = pd.Series(cleaned_df["Resolution"]).astype('category')

label_ints = cats.cat.codes
labels = cats
cleaned_df = cleaned_df.drop(["Resolution"], axis=1)
cleaned_df = pd.get_dummies(cleaned_df, prefix='sf_')
#Scale Data
scaler = StandardScaler()
scaler.fit(cleaned_df)
scaled_X_vals = scaler.transform(cleaned_df)
print(len(scaled_X_vals))
print(len(label_ints))
X_train, X_test, y_train, y_test = train_test_split(scaled_X_vals, label_ints, test_size=0.5)

start = time.time()
clf = DecisionTreeClassifier()
fitted = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)


end = time.time()
time_ls.append(end - start)
classfs.append("DT")
decision_accuracy = accuracy_score(y_test, y_pred)
accuracy.append(decision_accuracy)

f1 = f1_score(y_test, y_pred,  average='weighted')
print("f1 score: {}".format(f1))
print("accuracy score: {}".format(decision_accuracy))
print("Runtime: " + str(time_ls))
```

```
# hplot_vis(labels.unique(), f1, "f1 scores")


def random_forest( trees, max_d, min_sample, name):
    start = time.time()

    clf = RandomForestClassifier(n_estimators=trees, max_depth=max_d, min_samples_split=min_sample)
    clf.fit(X_train, y_train)
    preds = clf.predict(X_test)
    accuracy.append(accuracy_score(y_test, preds))
    fscore.append(f1_score(y_test, preds, average='weighted'))
    classfs.append(name)
    end = time.time()
    time_ls.append(end - start)
    print("accuracy score: {}".format(accuracy[-1]))
    print("Weighted f1: {}".format(fscore[-1]))
    print("Runtime: {} seconds".format(time_ls[-1]))

random_forest(30, 4, 50, "RF1")

random_forest(30, 4, 5, "RF2")

random_forest(30, 4, 100, "RF3")

random_forest(30, 1, 50, "RF4")

random_forest(60, 8, 50, "RF5")

random_forest(60, 16, 50, "RF6")

random_forest(120, 16, 50, "RF7")

timecolor = []
#Generate Colors on red/green axis based on execution time
for time in time_ls:
    percent_red = time/350
    percent_green = 1 - percent_red
    red_10 = int(percent_red * 255)
    green_10 = int(percent_green * 255)
    red_16 = str(hex(red_10))[-2:].replace("x", "0")
    green_16 = str(hex(green_10))[-2:].replace("x", "0")
    timecolor.append("#"+str(red_16)+str(green_16)+"88")
```

```
fig, ax = plot.subplots(figsize=(11, 9))
rects = ax.bar(classfs, accuracy, color=timecolor)
random_chance = 1/len(label_ints.unique())
plot.axhline(y=random_chance, color='r', linestyle='-')

# Indicate Times.
labels = ["%ds" % t for t in time_ls]



plot.plot()
for rect, label in zip(rects, labels):
    ax.text(rect.get_x() + rect.get_width() / 2, .1, label,ha='center', va='bottom')
```