Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему АНАЛІЗ І ПРОГНОЗУВАННЯ ЗЛОЧИННОСТІ НА ОСНОВІ МЕТОДІВ МАШИННОГО НАВЧАННЯ
ANALYZE AND PREDICT CRIME BASED ON MACHINE LEARNING METHODS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-220
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерні науки

СТИЧУК М.О.
(ПРІЗВИЩЕ та ініціали)
Керівник ЛЕОЩЕНКО С.Д.
(ПРІЗВИЩЕ та ініціали)
Рецензент СКРУПСЬКИЙ С.Ю.
(ПРІЗВИЩЕ та ініціали)

2024

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____ Сергій СУББОТІН
"____" _____2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

СТИЧУК Маргарити Олегівни
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Аналіз і прогнозування злочинності на основі методів машинного навчання.
Analyze and Predict Crime Based on Machine Learning Methods

керівник проєкту (роботи) д-р філос., ЛЕОЩЕНКО Сергій Дмитрович ,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від " 24 " квітня 2024 року № 168
2. Строк подання студентом проєкту (роботи) 03 червня 2024 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Опис програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЕОЩЕНКО С.Д., ст. викладач		
Нормоконтроль	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “ 15 ” квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	**Примітка**
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки.		
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження	5 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка) _____ Маргарита СТИЧУК
(підпис) (Імя ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Сергій ЛЕОЩЕНКО
(підпис) (Імя ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 114 с., 3 табл., 56 рис., 3 дод., 35 джерел.

ЗЛОЧИННІСТЬ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ, ДЕРЕВО РІШЕНЬ, ВИПАДКОВИЙ ЛІС, K-НАЙБЛИЖЧИХ СУСІДІВ, РЕГРЕСІЯ, МУЛЬТИНОМІАЛЬНА ЛОГІСТИЧНА РЕГРЕСІЯ, API.

Об’єкт дослідження – процес аналізу і прогнозування злочинності.

Предмет дослідження – програмні засоби для аналізу і прогнозування даних зі сфери злочинності.

Мета роботи – розробка програмного забезпечення для аналізу і прогнозування злочинності.

Матеріали, методи та технічні засоби: мова програмування – Python, середовище розробки – PyCharm. Основні використані бібліотеки: requests для API запитів, scikit-learn для алгоритмів машинного навчання, numpy для роботи з масивами, pandas для маніпулювання даними і їх аналізу, seaborn для візуалізації даних.

Результати. У ході виконання роботи було розроблено програмне забезпечення для аналізу і прогнозування злочинності за допомогою методів машинного навчання.

Практична цінність роботи полягає в розробці програмного забезпечення для розв’язання практичної задачі аналізу і прогнозування злочинності.

Висновки. Розроблено програмне забезпечення, яке отримує дані за допомогою API запитів, формує набір даних, обробляє атрибути, шукає між ними зв’язок, вибирає найважливіші, навчає моделі класифікації та регресії і тестує їх, проводить аналіз результатів і візуалізацію патернів.

Галузь використання – пошук зв’язків між даними та прогнозування злочинів у сфері криміналістики.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 114 pages, 3 tables, 56 figures, 3 appendixes, 35 sources.

CRIME, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, CLASSIFICATION, DECISION TREE, RANDOM FOREST, K-NEAREST NEIGHBORS, REGRESSION, MULTINOMIAL LOGISTIC REGRESSION, API.

The object of research is the process of analyzing and forecasting crime.

The subject of the research is software tools for analyzing and predicting data from crime data sets.

The purpose of the study is to create a software for crime analysis and forecasting.

Materials, methods and technical means: Python programming language, PyCharm development environment. Main libraries used: requests for API requests, scikit-learn for machine learning algorithms, numpy for working with arrays, pandas for data manipulation and analysis, seaborn for data visualization.

Results. In the course of the research, software for analyzing and predicting crime using machine learning methods was developed.

The practical value of the work lies in the development of software for solving the practical problem of crime analysis and forecasting.

Conclusions. We have developed software that receives data using API requests, generates a dataset, processes attributes, searches for connections between them, selects the most important ones, trains and tests classification and regression models, analyzes the results and visualizes patterns.

The field of application is the search for connections between data and crime prediction in the field of forensics.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface,

FN – False Negative;

FP – False Positive;

TN – True Negative;

TP – True Positive;

VS – Visual Studio;

ПЗ – програмне забезпечення;

ШІ – штучний інтелект.

ВСТУП

Прогнозування злочинності за допомогою машинного навчання привернуло значну увагу дослідників останніми роками, зосередившись на виявленні моделей і тенденцій у скоєнні злочинів. У цій галузі виникає багато актуальних задач, все більше з яких отримують практичне застосування. Машинне навчання – це галузь штучного інтелекту (ШІ) та комп'ютерних наук, що використовує дані та алгоритми для імітації процесу навчання людини.

Однією з основних задач в галузі машинного навчання є виявлення шаблонів та прийняття рішень на їх основі.

Тому актуальною є задача розробки ПЗ, що дозволить аналізувати та прогнозувати злочинність за допомогою машинного навчання.

Метою роботи є розробка ПЗ для аналізу і прогнозування злочинності.

Для досягнення поставленої мети в роботі потрібно вирішити такі задачі:

– виконати дослідження предметної області;

– проаналізувати використання ШІ у сфері злочинності;

– дослідити сферу машинного навчання, його роботу, категорії;

– дослідити потрібні методи вибору ознак, кореляційного аналізу, класифікації та регресії;

– вивчити особливості використання Application Programming Interface (API);

– розробити програму для аналізу та прогнозування злочинності.

Об'єкт дослідження – процес аналізу і прогнозування злочинності.

Предмет дослідження – програмні засоби для аналізу і прогнозування даних зі сфери злочинності.

Методи дослідження. Для розробки ПЗ застосовані методи машинного навчання та інтелектуального аналізу даних.

Наукова новизна одержаних результатів. Розроблено ПЗ для аналізу і

прогнозування злочинності, у якому, на відміну від існуючих, додатково присутні можливість більш глибокого аналізу даних, наявність прогнозування, відсутність необхідності користувачу самому шукати дані та можливість користуватися продуктом безкоштовно.

Практичне значення одержаних результатів полягає у тому, що розроблено ПЗ для аналізу і прогнозування злочинності, яке може бути використано для підвищення розуміння патернів та передбачення злочинності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Перший розділ буде присвячено детальному аналізу проблемної області пов’язаної з впливом злочинності на суспільство, використанням ШІ у криміналістиці, дослідженням машинного навчання, роботою з API, порівнянням аналогів та постановкою технічного завдання.

1.1 Аналіз впливу злочинності на суспільство

У цьому розділі буде проведено аналіз предметної області – злочинності у світі.

Злочинність є негативним явищем, котре трапляється у всьому світі не залежно від ступеню розвитку країни. Злочинна діяльність завдає серйозної шкоди економіці, а також впливає на якість життя та добробут населення, що в свою чергу приводить до проблем соціального та суспільного характеру [1].

Загалом, злочини відбуваються через різні обставини, серед яких специфічні мотиви, людська природа та поведінка, критичні ситуації та бідність. Крім того, сприяють зростанню насильницьких злочинів інші численні фактори, такі як безробіття, гендерна нерівність, висока щільність населення, дитяча праця та неграмотність. Густонаселені та зростаючі міста також мають сильну кореляцію з більш високим рівнем злочинності, який пов’язаний з різними типами середовища, такими як комерційні будівлі та муніципальні житлові райони. Однією з основ соціально стійкої громади є мінімізація злочинності, щоб люди могли жити мирно, в той час, як у корумпованих суспільствах, громади не можуть процвітати ні в соціальному, ні в економічному плані. Таким чином, аналіз звітів у сфері злочинності та статистичних даних є однією з причин, що призводить до покращення безпеки людства та підтримки сталого розвитку [1].

Нижче представлені деякі цифри, які допомагають зрозуміти, як злочинність впливає на економіку. В Європейському Союзі:

– вартість крадіжок оцінюється приблизно в 30 мільярдів євро на рік;

– збитки від шахрайства оцінюються приблизно в 20 мільярдів євро на рік;

– збитки від корупції оцінюються приблизно в 120 мільярдів євро на рік;

– збитки від організованої злочинності оцінюються приблизно в 200 мільярдів євро на рік.

У 2021 році протидія злочинності і насильству в Мексиці коштувала країні 192 мільярди доларів США, що еквівалентно 14,6% ВВП Мексики.

Або інший приклад: проведене в 2021 році дослідження Чиказького університету показало, що кожний рік США витрачають на протидію злочинності близько 5,76 трильйона доларів США, що становить приблизно 3,2% ВВП США [2].

1.2 Штучний інтелект у сфері криміналістики

ШІ – це підхід до вивчення розумної поведінки, який заснований на припущенні, що інтелект можна відтворити. Однією з цілей є вивчення та розуміння того, як працює людський інтелект, іншою – його імітація за допомогою комп’ютера [3].

Методи машинного навчання у сфері криміналістики зазвичай розглядаються, як інструменти досягнення конкретних цілей. Статистичні методи використовуються для отримання висновків про зв’язки між ознаками і залежною змінною, а методи машинного навчання – для концепцій передбачення і прогнозування [4]-[5].

Основою значної частини функцій ШІ в кримінальному судочинстві є використання розпізнавання образів для виявлення та прогнозування злочинної діяльності, але це не єдине його застосування. Нижче представлено деякі інші способи використання ШІ у цій сфері:

– цифровий аналіз;
– аналіз ДНК;
– розпізнавання пострілів;
– судові розгляди [4]-[5].

1.3 Машинне навчання

Машинне навчання – це галузь ШІ та комп'ютерних наук, яка фокусується на використанні даних та алгоритмів, що дозволяють ШІ імітувати спосіб навчання людини, поступово підвищуючи його точність.

Машинне навчання має певні недоліки:

– залежність від точності вхідних даних;
– витрати ресурсів (обчислювальних, часових);
– схильність до помилок;
– необхідність балансу між перенасиченням і недонасиченням даними [6].

Серед переваг машинного навчання важливо виділити наступні:

– обробка великих об'ємів даних;
– адаптивність;
– виявлення варіативних шаблонів/патернів;
– автоматизація;
– постійний розвиток;
– широкий спектр застосування [7].

Процес машинного навчання починається зі збору необхідних даних, наприклад, цифр, зображень або тексту. Їх називають навчальні (вхідні) дані і чим більше таких даних буде зібрано, тим кращою буде програма.

Наступним кроком є вибір моделі машинного навчання та надання їй даних. Після чого комп’ютерна модель навчиться на навчальних даних виявляти закономірності та робити прогнози [7].

Роботу системи машинного навчання розділяють на три основні типи:

– процес прийняття рішень: алгоритми намагаються знайти закономірність, на основі якої вони проведуть класифікацію або передбачать потенційний результат;

– функція помилки: алгоритми перевіряють точність прогнозу, порівнюючи його з аналогічними прикладами;

– процес оптимізації або оновлення: на цьому етапі алгоритм аналізує та оцінює рівень помилки, а потім оновлює процес прийняття рішень, щоб у наступних випадках помилка була меншою [7].

Загалом, алгоритми машинного навчання поділяють на три категорії:

– контрольоване навчання – тип машинного навчання, при якому модель навчається на наборах даних з вже заданими правильними «відповідями» – маркованих. Алгоритм аналізує тренувальні дані і виробляє функцію, яка може бути використана для прогнозування нових екземплярів. Контрольоване навчання включає в себе наступні алгоритми: регресійні, класифікації, наївні класифікатори Байєса, нейронні мережі та алгоритми випадкових лісів [8];

– неконтрольоване навчання – це тип машинного навчання, при якому робляться висновки з немаркованих наборів даних. Такі алгоритми допомагають в розпізнаванні зв’язків між об’єктами, патернів. Найпоширенішим методом неконтрольованого навчання є кластерний аналіз, який використовується для аналізу даних, щоб знайти приховані закономірності або групувати дані. Неконтрольоване навчання включає в себе наступні типи алгоритмів: кластеризація К-середніх, ієрархічна кластеризація, імовірнісна кластеризація [8];

– навчання з підкріпленням – тип машинного навчання, де практикується намагання змусити агента діяти у світі так, щоб максимізувати свою винагороду. Агенту не повідомляється які дії потрібно виконувати, він сам повинен визначити, які дії приносять більше винагороди, спробувавши їх. Навчання з підкріпленням відрізняється від того ж контрольованого, тому що воно не дає прикладів як і що треба робити агенту. Зазвичай такі методи

використовуються в розробці відеоігор або для навчання роботів [8].

1.3.1 Вибір ознак

Вибір ознак – важливий етап у розробці ефективних моделей машинного навчання, який допомагає мінімізувати обчислювальну складність і надмірну перенасиченість. Це процес визначення та відбору релевантних ознак, які мають важливі аспекти, що підвищують прогностичну здатність і точність моделі [9].

Методи вибору ознак поділяють на три основні групи:

– методи фільтрації – підбирають внутрішні властивості ознак, виміряні за допомогою одновимірної статистики. Ці методи швидші та менш обчислювально затратні, ніж методи обгортки. При роботі з даними високої розмірності використання таких методів є дешевшим з точки зору обчислень [10];

– методи обгортки – вимагають певного методу для пошуку в просторі всіх можливих підмножин ознак, оцінюючи їхню якість шляхом навчання та оцінювання класифікатора з цією підмножиною ознак. Процес вибору ознак базується на конкретному алгоритмі машинного навчання, який ми намагаємося пристосувати до заданого набору даних. Він слідує підходу жадібного пошуку (вибір найкращого доступного варіанту на кожному кроці), оцінюючи всі можливі комбінації ознак відповідно до критерію оцінки. Методи обгортки зазвичай дають кращу точність прогнозування, ніж методи фільтрації [10];

– вбудовані методи – охоплюють переваги методів обгортки та фільтрації, враховуючи взаємодію ознак, але при цьому зберігаючи розумні обчислювальні витрати. Вбудовані методи є ітеративними в тому сенсі, що вони дбають про кожну ітерацію процесу навчання моделі і ретельно виділяють ті ознаки, які роблять найбільший внесок у навчання для конкретної ітерації [10].

1.3.1.1 Дерева рішень та випадковий ліс при виборі ознак

У контексті вибору ознак методи дерев рішень та випадкового лісу є вбудованими методами.

Дерева рішень – це алгоритм керованого навчання, який використовується для класифікації та регресійного моделювання. Дерева класифікують або передбачають категоріальні або безперервні змінні на основі правила розділяй і володарюй. Оригінальний набір даних розділяється на менші піднабори. Кожний піднабір піддається серії питань про попередньо зафіксовані атрибути всередині класифікатору. Щоб отримати відповідь, класифікатор задає питання. Це відбувається поки не виноситься фінальне рішення – якому з класів належить піднабір. З розбиттям набору даних на менші і менші піднабори дерева рішень все більше розростаються. Цей процес розбиття називають рекурсивним розбиттям [11].

Щоб побудувати дерево рішень потрібно:

а) протестувати всі атрибути і обрати функцію, що визначить найкращий кореневий вузол;

б) розбити набір даних на піднабори на основі гілок(відповідей) від кореневого вузла;

в) протестувати решту атрибутів, щоб побачити котрі з них більше підходять гілкам від кореневого вузла;

г) продовжувати цей процес для всіх гілок поки не виконається одне з наступного:

1) всі екземпляри піднабору не будуть одного типу;
2) не залишиться екземплярів;
3) не залишиться атрибутів [11].

Графічною репрезентацією рекурсивного розбиття є дерево. Оригінальний набір даних міститься у кореневому вузлі – перший вузол, від якого розходяться гілки. Спираючись на перше рішення, дані розбиваються на два чи більше вузлів. Подальші розбиття відбуваються до тих пір, поки не

досягнуто кінцевого вузла. Процедура будування дерева методом поділяй і володарюй проходить так:

– вибір функції визначення кореневого вузла, створення гілок для кожної відповіді;

– розбиття даних на піднабори – один піднабір для кожної гілки, що виходить з вузла;

– повторення цього процесу для кожної гілки з використанням лише тих екземплярів, які досягли цієї гілки;

– зупинка рекурсії, коли всі екземпляри мають один клас.

Підсумовуючи, дерева рішень мають кореневий вузол, гілки, звичайні вузли та листові вузли. Вузли – це рішення, з яких потім виходять гілки. Гілки рекурсивно створюють листові вузли. Кожен листовий вузол представляє собою результат класифікації – мітку класу [11]. Схему дерева рішень представлено на рис. 1.1.

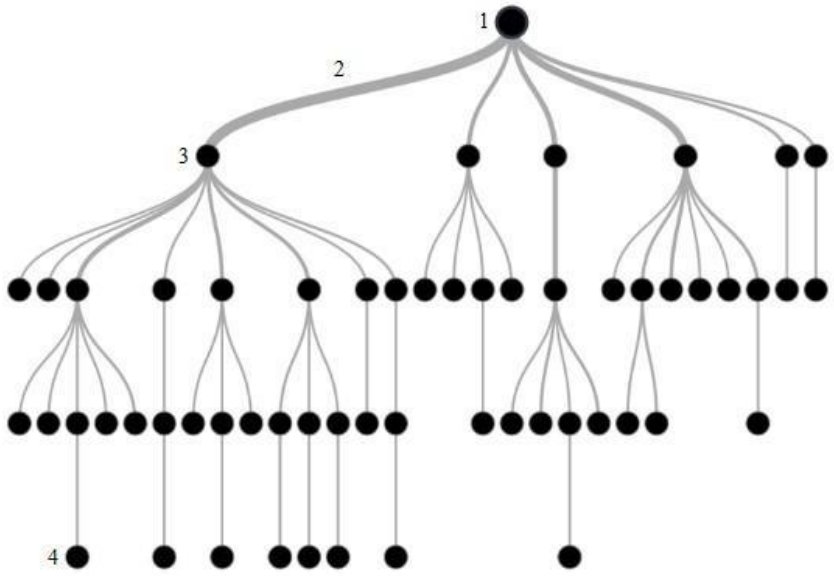


Рисунок 1.1 – Схема дерева рішень [12]

Рисунок 1.1 має наступні позначення:

– 1 – кореневий вузол;

– 2 – гілка;

– 3 – звичайний або внутрішній вузол;

– 4 – листовий вузол.

Серед переваг дерев рішень можна виділити:

– простоту розуміння та інтерпретації;

– можливість працювати з категоріальними і числовими атрибутами;

– можливість застосування у задачах класифікації і регресії;

– можливість використовувати атрибут feature importance для визначення важливості ознак;

– можливість праці з відсутніми значеннями та викидами [13].

Серед недоліків можна виділити:

– можливість перенавчання;

– чутливість результатів до невеликих змін у даних;

– потенційне упередження при наявності незбалансованих даних [13].

Випадковий ліс – це різновид алгоритму бегінгу (bagging algorithm), який використовує певну кількість дерев рішень. Більш детально про випадковий ліс буде описано нижче у підрозділі «Випадковий ліс».

1.3.1.2 Кореляційний аналіз

Кореляційний аналіз – це статистичний метод для визначення сили зв'язку між двома змінними. Він використовується для виявлення закономірностей і тенденцій у даних та прогнозування майбутніх подій [14].

Кореляція показує наскільки сильний зв’язок існує між двома змінними і визначається від -1 до 1. Чим більше це значення, тим сильніша кореляція. Знак кореляції показує напрямок відношення між змінними – вона може бути позитивна, негативна або нульова [14].

Позитивна кореляція означає, що дві змінні мають прямий зв’язок. Якщо одна зі змінних збільшується, інша також збільшується. Негативна кореляція означає, що дві змінні мають зворотній зв’язок. Якщо одна зі

змінних збільшується, то інша зменшується. Нульова кореляція означає, що між двома змінними немає зв’язку. Зміна однієї змінної ніяк не вплине на іншу [14].

Хоча кореляційний аналіз корисний для виявлення взаємозв'язків між змінними, важливо зазначити, що кореляція не обов'язково означає причинно-наслідковий зв'язок. Те, що два фактори змінюються разом на основі наявних даних, не означає, що один фактор викликає зміни в іншому. Може існувати якась третя, базова змінна, що впливає на обидва фактори [15].

1.3.2 Формування вибірок

Дані, які використовують для навчання моделі, не повинні використовувати для її тестування. Це може привести до випадків, коли модель помилково вважають дуже точною, коли насправді це не так. Щоб цьому запобігти, дані розділяють на дві частини – одна з яких буде використовуватись тільки для того, щоб навчити модель, а інша тільки для тестування моделі [16].

1.3.3 Класифікація

Класифікація – це алгоритм машинного навчання, при якому відбувається віднесення об’єкта до однієї з вже відомих категорій на підставі його ознак.

Задача класифікації – розбити множину об’єктів на класи, всередині кожного з яких вони передбачаються схожими один на одного, мають приблизно однакові ознаки. Ці рішення ґрунтуються на основі аналізу значень атрибутів [17].

Зазвичай процес вирішення задачі класифікації складається з таких кроків: завантаження даних, попередня обробка даних, формування вибірок

для тренування та тестування, тренування та перевірка якості моделі класифікації [17].

1.3.3.1 Метрики якості класифікації

Матриця помилок – таблична репрезентація всіх фактичних та прогнозованих класів кожного об’єкта у вибірці. Рядки показують мітки фактичних класів, а стовбці – передбачених. Числа репрезентують кількість випадків у кожній комбінації оригінального та передбачуваного класів [18].

Матриця помилок допомагає зрозуміти які класи модель класифікує добре, а які ні. Таким чином, добре навчений класифікатор покаже матрицю помилок, в якій найбільші значення стоять на діагоналі матриці, а невеликі значення (в ідеалі нулі) – на інших позиціях. На рисунку 1.2 представлено приклад матриці помилок.

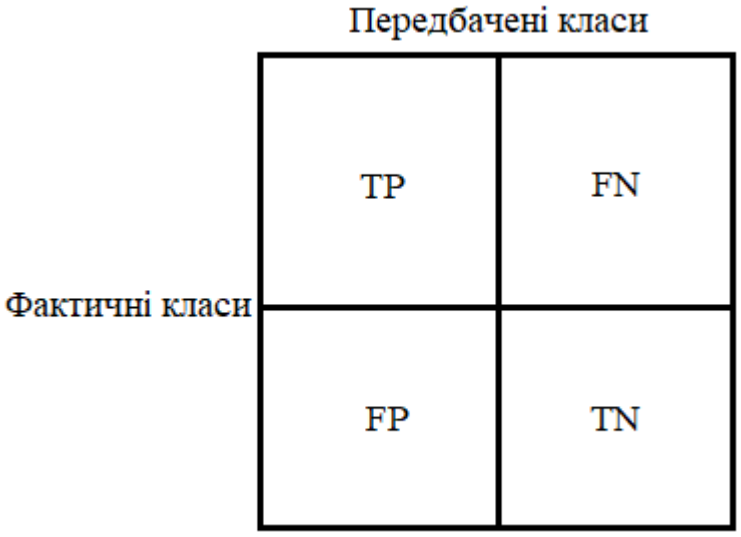


Рисунок 1.2 – Приклад матриці помилок

Матриця має чотири значення, які оцінюють продуктивність моделі:

– справжні позитивні результати (TP) – правильні прогнози позитивних результатів;

– справжні негативні результати (TN) – правильні прогнози негативних результатів;

– помилкові позитивні результати (FP) – неправильні прогнози позитивних результатів;

– помилкові негативні результати (FN) – неправильні прогнози негативних результатів [19].

Існують також інші стандартні метрики, котрі широко використовуються для оцінки моделі, такі як:

– accuracy (частка успішності);

– error (помилка);

– precision (точність);

– recall, sensivity або TP rate (чутливість);

– fp rate;

– tn rate (специфічність);

– f-measure (середнє гармонійне) [19].

Частка успішності показує частку правильно класифікованих екземплярів і вираховується за формулою (1.1):

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (1.1)$$

Помилка показує частку неправильно класифікованих екземплярів і вираховується за формулою (1.2):

$$error = 1 - accuracy = \frac{FP+FN}{TP+TN+FP+FN}, \quad (1.2)$$

Точність показує частку екземплярів, які було класифіковано до «позитивного» класу, і які дійсно належать цьому класу. Точність вираховується за формулою (1.3):

$$precision = \frac{TP}{TP+FP} = \frac{TP}{P'}, \quad (1.3)$$

Чутливість показує наскільки добре було розпізнано «позитивний» клас і вираховується за формулою (1.4):

$$recall = \frac{TP}{TP+FN} = \frac{TP}{P}, \quad (1.4)$$

FP rate показує відсоток об’єктів інших класів, які було помилково занесено в клас, що розглядається, і вираховується за формулою (1.5):

$$FP\ rate = \frac{FP}{FP+TN} = \frac{FP}{N}, \quad (1.5)$$

Специфічність показує наскільки добре було розпізнано «негативний» клас і вираховується за формулою (1.6):

$$TN\ rate = \frac{TN}{TN+FP} = \frac{TN}{N}, \quad (1.6)$$

F-measure – середнє гармонійне між точністю і чутливістю. Цей параметр досягає максимуму при значеннях, рівних одиниці, і близький до нуля, якщо один з аргументів близький до нуля. Зазвичай ці два параметри розглядаються сумісно, а в реальній практиці стоїть завдання знайти оптимальний баланс між цими двома метриками. Середнє гармонійне вираховується за формулою (1.7):

$$F - measure = 2 * \frac{precision*recall}{precision+recall}, \quad (1.7)$$

Проте, необхідно вибирати таку метрику, яка найбільш точно і без зайвих припущень робить оцінку. Наприклад, якщо є перекіс у розподілі класів

(один з класів представлено набагато більшою кількістю екземплярів, ніж інший), то багато показників треба використовувати з обережністю і розумінням. Та ж частка успішності для такого незбалансованого набору даних може бути інтерпретована неправильно. Навіть при повному не розпізнаванні меншого класу це число буде великим, оскільки скоріш за все модель правильно розпізнає екземпляри більшого класу [19].

1.3.3.2 Ансамблеве навчання

З огляду на те, що ми не можемо заздалегідь знати, яка модель буде працювати краще з конкретним набором даних, знаходження найкращої моделі для проєкту може зайняти багато часу. Ансамблеве навчання – це процес пошуку ефективної моделі, який включає у себе підготовку декількох різних моделей, а потім об’єднання їх прогнозів для покращення результатів [20].

Найпопулярніший метод ансамблевого навчання включає в себе тренування кількох версій однієї і тієї ж моделі таким чином, щоб кожний ансамбль був різним (наприклад, використання дерев рішень на різних піднаборах тренувального набору даних), а потім поєднання прогнозів шляхом усереднення. Менш популярний, проте не менш дієвий, метод включає в себе тренування різних алгоритмів на одних і тих самих даних і потім поєднання їх прогнозів. Ефективність ансамблевого навчання полягає в тому, що кожна модель виконує прогнози незалежно одна від одної. Проте треба брати до уваги, що при виконанні кількох моделей замість однієї, продуктивність буде меншою [20].

Загалом, ансамблеве навчання зменшує розкиди і покращує середню ефективність передбачення.

Три основних види ансамблевого навчання:

– беггінг – навчання багатьох дерев рішень на різних вибірках одного набору даних і усереднювання результатів;

– бустинг – послідовне навчання різних моделей на одних і тих же даних з метою виправлення помилок у прогнозах і зважене усереднювання результатів;

– стекінг – навчання кількох моделей на одних і тих же даних і використання ще однієї моделі для кращого об’єднання результатів прогнозування [20].

1.3.3.3 Випадковий ліс

Випадкові ліси – це ансамблевий класифікатор, який складається з багатьох дерев рішень, результати котрих він усереднює і на основі яких робить прогноз. Метод поєднує у собі беггінг та випадковий вибір ознак. Це означає, що він випадково вибирає підмножини даних для навчання кожного дерева, а також підмножини ознак для кожної розбіжності у дереві [21].

Загалом, для багатьох наборів даних випадкові ліси:

– доволі точно прогнозують клас;

– стійкі до шуму;

– можуть оцінювати важливість ознак;

– можуть обробляти відсутні дані та викиди.

Серед недоліків можна виділити:

– схильність до перенавчання;

– важливість ознак для категоріальних атрибутів не є надійним параметром;

– використання більшої кількості пам’яті [21].

1.3.3.4 Алгоритм k-найближчих сусідів

Алгоритм k-найближчих сусідів (KNN) — це непараметричний класифікатор із контрольованим навчанням, який використовує близькість для класифікацій або прогнозів щодо групування окремої точки даних.

Метою алгоритму k-найближчих сусідів є ідентифікація найближчих сусідів даної точки, щоб можна було призначити цій точці мітку класу [22].

Нижче представлено послідовність роботи алгоритму k-найближчих сусідів:

– вибір оптимального значення K. K представляє кількість найближчих сусідів, яку потрібно враховувати під час прогнозування;

– розрахунок відстані. Для вимірювання подібності між цільовими та навчальними точками даних використовується евклідова (пряма відстань між двома точками), манхеттенська (абсолютне значення між двома точками) або відстань мінковського (узагальнена форма евклідової та манхеттенської відстані). Відстань обчислюється між кожною точкою даних у наборі даних і цільовою точкою;

– пошук найближчих сусідів. K точок даних з найменшою відстанню до цільової точки є найближчими сусідами;

– голосування за класифікацію або визначення середнього для регресії. У задачі класифікації мітки класу визначаються голосуванням більшості. Клас із найбільшою кількістю входжень серед сусідів стає прогнозованим класом для цільової точки даних. У задачі регресії мітка класу обчислюється шляхом усереднення цільових значень K найближчих сусідів. Розраховане середнє значення стає прогнозованим результатом для цільової точки даних [22].

1.3.4 Регресія

Регресійний аналіз – це статистична техніка для дослідження та моделювання зв’язків між змінними. Задача регресії – визначити найбільш підходящу функцію, яка характеризує зв’язок між змінними.

Регресійні алгоритми у машинному навчанні діляться на наступні основні типи:

– лінійні;

– логістичні;
– поліноміальні;
– Ridge;
– LASSO.

1.3.4.1 Мультиноміальна логістична регресія

Логістична регресія призначена для наборів даних, які мають числові вхідні змінні та категоріальну цільову змінну, яка представлена двома класами. Логістична регресія не може працювати з наборами даних, де цілова змінна представлена більш, наж двома класами. Задачі цього типу називаються задачами бінарної класифікації. Логістична регресія розроблена для задач двох класів, моделюючи ціль за допомогою біноміальної функції розподілу ймовірностей [23]-[24].

Мультиноміальна логістична регресія корисна в ситуаціях, коли потрібно мати можливість класифікувати об’єкти, представлені мінімум трьома класами.

Незалежні змінні можуть бути факторами або коваріантами. Загалом, фактори мають бути категоріальними змінними, а коваріанти мають бути неперервними змінними [23]-[24].

1.4 API

API – інтерфейс прикладного програмування. Це інтерфейс, який дозволяє додаткам взаємодіяти один з одним через мережу без втручання користувача. API робить це шляхом «розкривання» частини функціоналу з обмеженим доступом для інших додатків. Це дозволяє вільно обмінюватись даними з іншими системами, не надаючи ,безпосереднього доступу до коду [25].

API спрощують під’єднання, інтеграцію та розширення програмних систем. Програмні системи зазвичай ізольовані та до функціональності однієї з них неможливо отримати доступ з іншої системи. API пропонують можливість з’єднання цих ізольованих програмних сутностей [25].

Коли клієнт робить виклик API, адресатом цього запиту є кінцева точка API (endpoint). Кінцеві точки API – це спосіб доступу до даних або функціональності програми, сервісу чи ПЗ. Ця точка є конкретною URL-адресою, яку можна використовувати для доступу до даних із зовнішнього джерела, наприклад, бази даних або вебсервісу [26].

Коли кінцева точка API надсилає відповідь, це називається відгуком (response). Важливо знати, що відповіді можуть бути різними. Часто відповіддю є дані, які клієнт запитував у виклику API. Якщо API не може верифікувати клієнтів API, відповіді можуть також повертати коди помилок [26].

1.5 Аналіз існуючих аналогів

1.5.1 RapidMiner Studio

RapidMiner Studio – ПЗ з відкритим кодом для підготовки, аналізу та моделювання даних. Важливим моментом є те, що для роботи з ним потрібно самостійно шукати і завантажувати набори даних. Його інтерфейс представлено на рис. 1.3 [27].

Серед переваг даного ПЗ можна визначити просту установку, наявність безкоштовної версії для навчання і невеликих проєктів, інтуїтивний інтерфейс, підтримка широкого спектру алгоритмів для глибокого аналізу.

Недоліками є наявність дорогої ліцензії для використання повного спектру можливостей, менша гнучкість у порівнянні з Python, наявність не всіх алгоритмів, необхідність завантажувати свої власні набори даних.

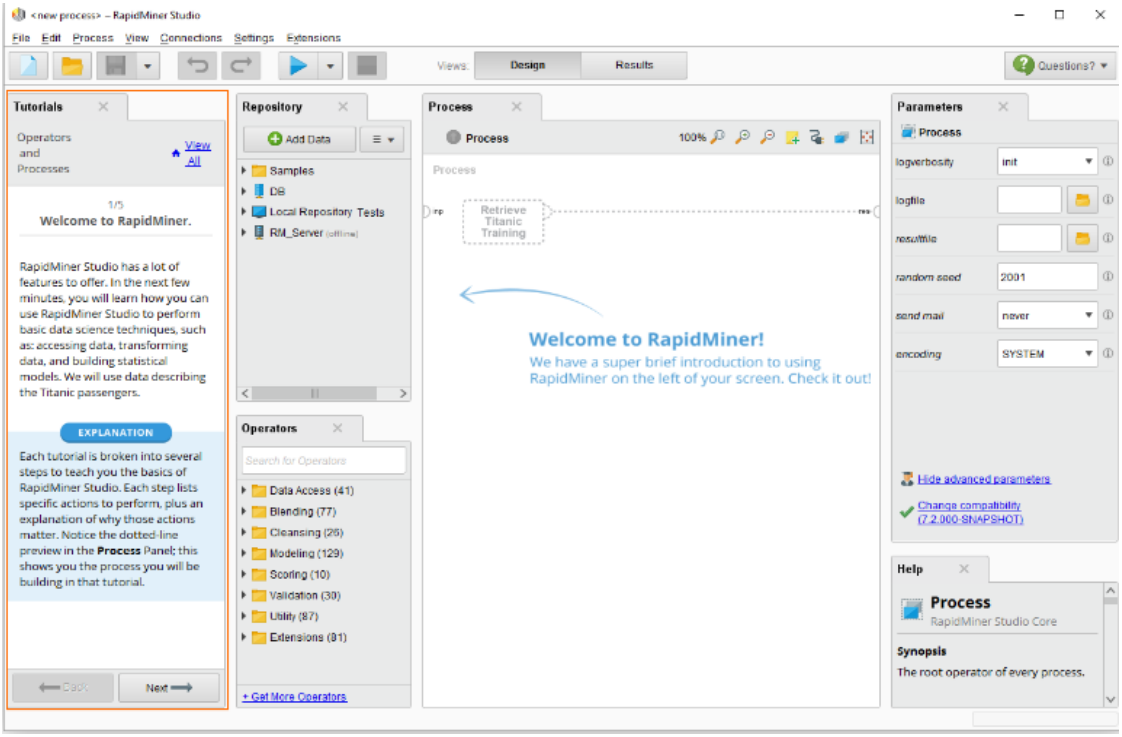


Рисунок 1.3 – Інтерфейс RapidMiner Studio [27]

1.5.2 Crime Data Explorer від FBI

Crime Data Explorer – це програма ФБР, у рамках якої збираються та публікуються дані про злочинність у США. Аналітику можна знайти на їх офіційному сайті. Його інтерфейс можна побачити на рис. 1.4 [28].

Серед переваг треба виділити безкоштовність, різноманітні варіанти візуалізації, гнучкість фільтрування, доступ до офіційних даних ФБР, можливість завантажити дані, простота використання.

До недоліків можна віднести обмеженість даних, обмежену географічну область, відсутність інструментів прогнозування, не підходить для більш складного аналізу.

Рисунок 1.4 – Інтерфейс Crime Data Explorer [28]

1.5.3 Accurint Crime Analysis

Accurint Crime Analysis – це хмарна платформа для аналізу даних про злочинність. Вона призначена для допомоги правоохоронним органам, дослідникам та іншим, хто зацікавлений у вивченні та розумінні злочинності. Його інтерфейс можна побачити на рис. 1.5 [29].

До переваг можна віднести потужну візуалізацію даних, різні джерела даних(поліцейські звіти, записи суддів, публічні бази даних), можливість завантажити дані, наявність інструментів для аналізу, можливість прогнозування даних.

Недоліками є висока вартість ліцензії, складність використання.

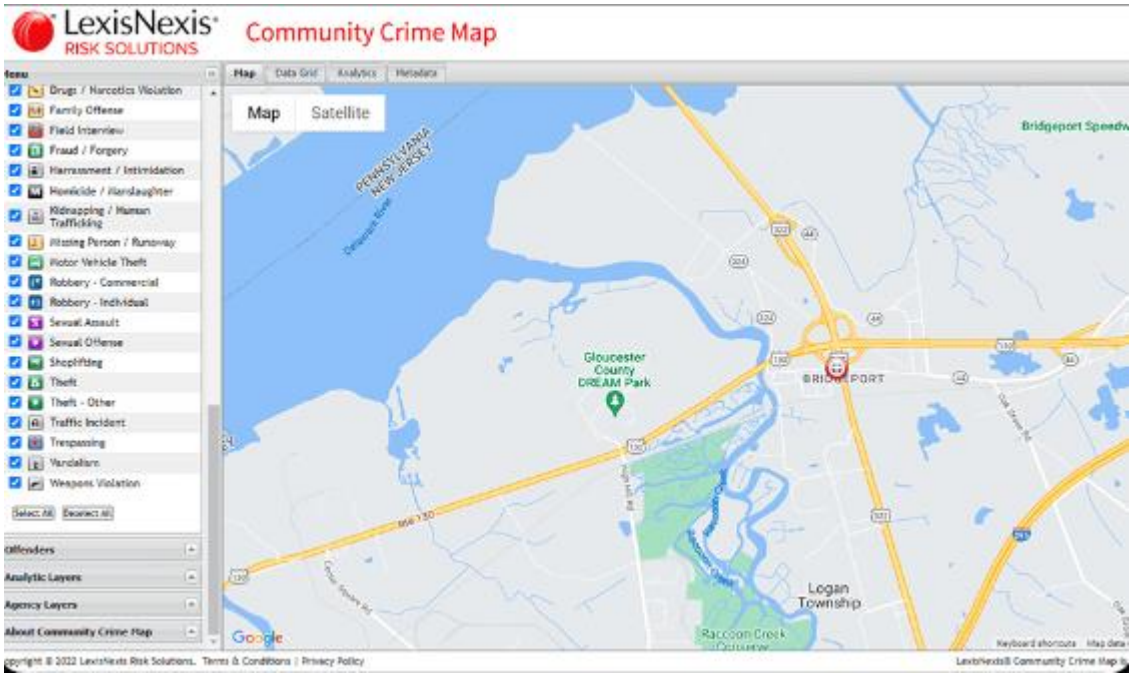


Рисунок 1.5 – Інтерфейс Accurint Crime Analysis [29]

1.5.4 Порівняння аналогів

Проведемо порівняння за певними критеріями, хід якого представимо у вигляді таблиці (табл. 1.1). У таблиці будуть порівнюватися попередні аналоги із за стосунком, що розробляється – «Програмне забезпечення для аналізу і прогнозування злочинності». Серед критеріїв виділимо наступні:

– безкоштовність;

– простота встановлення – критерій, що характеризує швидкість встановлення ПЗ або можливість легко почати користуватися вебсервісом;

– необхідність користувачеві самостійно завантажувати набори даних – критерій, що характеризує чи потрібно користувачеві самостійно шукати і завантажувати набори даних для їх подальшого аналізу і прогнозування;

– використання декількох алгоритмів для більш детального аналізу;

– наявність прогнозування – критерій, що характеризує наявність механізму для прогнозування злочинності;

– наявність візуалізації – критерій, що характеризує наявність візуального відображення аналізу і результатів.

Таблиця 1.1 – Порівняння аналогів за критеріями

Критерій порівняння	Порівняння аналогів і застосунку			
	RapidMiner Studio	Crime Data Explorer	Accurint Crime Analysis	«Програмне забезпечення для аналізу і прогнозування злочинності»
Безкоштовність	+	+	–	+
Простота встановлення	+	+	+	+
Необхідність самостійного пошуку даних	+	–	–	–
Детальний аналіз	+	–	+	+
Наявність прогнозування	+	–	+	+
Наявність візуалізації	+	+	+	+

З результатів порівняння можна зробити висновок, що аналоги зазвичай пропонують користувачами простоту встановлення та наявність візуалізації, але не всі з них надають можливість користуватись продуктом безкоштовно, позбавляють користувача необхідності самому шукати дані, пропонують можливості більш глибокого аналізу та наявність прогнозування.

Саме тому актуальною залишається задача розробки нового ПЗ, яке буде мати все вищезгадане.

1.6 Технічне завдання

1.6.1 Призначення та область застосування програми

Програма, що буде розроблена, має на меті аналіз і прогнозування злочинності з використанням методів машинного навчання. Основні завдання цієї програми полягають у наданні користувачам візуалізації результатів, відсутності необхідності шукати необхідні дані власноруч, можливості більш глибокого аналізу та наявності прогнозування.

Основна область застосування програми це сфера аналізу злочинності. Програма допоможе користувачам більш детально заглибитись у аналіз та прогнозування злочинності – візуалізує основні патерни, покаже чи зв’язані між собою певні атрибути у вибірках, відбере ті, які найбільше впливають на результат, а також проведе прогнозування злочинності на основі наявних даних. Таким чином, програма дасть користувачам можливість більш глибоко проаналізувати і спрогнозувати злочинну діяльність, відокремити патерни.

1.6.2 Підстави для розробки

У сучасних умовах машинне навчання постійно використовують у сфері криміналістики для виявлення тенденцій у скоєнні злочинів, передбачення нових, аналізу результатів. Існуючі аналоги виявляються неефективними через наступні фактори: недоступність, необхідність обтяжувати користувача пошуком даних, відсутність більш глибокого аналізу патернів та вибору ознак, відсутність прогнозування.

Ці фактори вимагають розробки доступного ПЗ для аналізу і прогнозування злочинності, яке б вирішило ці проблеми – користувачу більше не потрібно було б самостійно шукати дані, була б наявна візуалізація патернів та вибору ознак, а також програма вміла б прогнозувати злочинну діяльність на основі вхідних даних.

1.6.3 Призначення розробки

Призначенням розробки є створення ПЗ, яке надає можливість аналізу і прогнозування злочинності на основі відкритих даних з поліцейських департаментів.

1.6.4 Вимоги до функціональних характеристик

ПЗ має реалізовувати наступні характеристики:

– завантаження даних: отримання даних з відкритого ресурсу за допомогою API запитів;

– попередня обробка даних: зміна типів даних для необхідних атрибутів, розбиття оригінального атрибуту на кілька менших;

– відбір атрибутів: визначення та відбір релевантних ознак;

– побудова моделей алгоритмів класифікації та регресії;

– аналіз і порівняння результатів;

– візуалізація даних: представлення даних у зручній для користувача формі.

1.6.5 Вимоги до надійності

Вимоги до надійності ПЗ включають в себе:

– перевірку алгоритмів, що використовуються у програмі, на точність та стабільність результатів;

– виведення повідомлень для користувача у разі виникнення помилок;

– виведення повідомлень для користувача, які показують етапи роботи програми;

– виведення повідомлень для користувача, які показують результати роботи програми.

1.6.6 Умови експлуатації

Для експлуатації ПЗ необхідно мати стабільне підключення до мережі Інтернет для здійснення API запитів, встановлену необхідну версію Python, систему керування пакетами pip (якщо встановлено версію Python меншу за 3.4 і вище, то туди pip за замовченням не входить), необхідні бібліотеки для встановлення, які можна буде завантажити за допомогою pip та файлу requirements.txt.

1.6.7 Вимоги до складу та параметрів технічних засобів

Комп’ютер користувача для коректної роботи з ПЗ має відповідати таким вимогам:

– стабільне підключення до мережі Інтернет для здійснення API запитів;

– операційна система: Windows 10;

– версія Python(>=3.12.1) та pip(>=23.3.2);

– вільний дисковий простір для встановлення необхідних бібліотек і роботи програми без збоїв: мінімум 1 ГБ;

– оперативна пам’ять: мінімум 8 ГБ;

– процесор: не менше 4 ядер з тактовою частотою не менше 3,4 ГГц.

1.6.8 Необхідні стадії і терміни розробки

Розробка ПЗ для аналізу та прогнозування злочинності включає наступні стадії та терміни:

– аналіз вимог: визначення вимог до програми. Термін: 1 тиждень;

– проєктування системи: розробка архітектури та вибір технологічних засобів. Термін: 1 тиждень;

– реалізація та програмування: написання коду програми та

реалізація визначених функцій. Термін: 2 тижні;

– тестування та налагодження: проведення випробувань для виявлення та усунення помилок. Термін: 1 тиждень.

1.6.9 Види випробувань

Для забезпечення якості та надійності буде проведено наступні види випробувань:

– функціональні тести: перевірка відповідності функціональних вимог програми;

– інтеграційні тести: перевірка взаємодії між різними частинами програми;

– системні тести: тестування програми в цілому.

1.7 Висновки за розділом

У першому розділі було проаналізовано предметну область ПЗ, що розробляється. Було виявлено вплив злочинності на суспільство, проаналізовано використання ШІ у сфері криміналістики. Також було досліджено машинне навчання, його типи, алгоритми, описано роботу з API, порівняно аналоги, поставлено технічне завдання.

2 ОПИС ПРОГРАМИ

Другий розділ буде присвячено висвітленню питань основних вимог щодо реалізації ПЗ. Особливу увагу буде присвячено вибору технічних засобів: мови програмування, середовища розробки, а також бібліотек для машинного навчання, візуалізації даних, API запитів.

2.1 Опис основних вимог до програми

Програма має надати можливість аналізу та прогнозування злочинності за допомогою алгоритмів машинного навчання на основі відкритих даних з поліцейських департаментів.

Таким чином, серед функціональних вимог виділимо:

– завантаження даних;

– попередня обробка даних;

– відбір релевантних атрибутів;

– передача оброблених даних моделям алгоритмів класифікації та регресії;

– тестування моделей;

– аналіз і візуалізація результатів за метриками;

– візуалізація патернів.

2.2 Вибір засобів проєктування і розробки

2.2.1 Вибір мови програмування

При розробці ПЗ було використано мову програмування Python.

Python – це інтерпретована, об’єктно-орієнтована та високорівнева мова програмування загального призначення з суворою динамічною типізацією. Python також підтримує кілька парадигм програмування: структурне, об'єктно-орієнтоване, функціональне, імперативне і аспектно-

орієнтоване. Інтерпретованість означає, що початковий код при запуску не перетворюється одразу в машинний, як у компільованих мовах, а виконується рядок за рядком за допомогою спеціального інтерпретатора. Високорівневість – швидкість і зручність використання для людини за допомогою введення додаткових смислових конструкцій, які описують структури даних, операції над ними. Для виконання високорівневі мови перетворюються в машинний код. Низькорівневі мови наближені до машинного коду і є більш складними для їх використання у розробці, проте не містять зайві конструкції, що збільшує їх швидкодію. Динамічна типізація означає, що більша частина перевірок типів даних виконується під час виконання програми, а не під час компіляції. Це дозволяє змінити тип даних змінної у процесі виконання програми, проте може призвести до неочікуваних помилок з типами даних при роботі [30]-[31].

Python була розроблена в 1990 році Гвідо ван Россумом. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні на всіх основних платформах [30]-[31].

Нижче буде представлено порівняння Python з іншими мовами програмування для машинного навчання – Java та R – за такими факторами:

– швидкість: тут найшвидшою мовою є Java – компільована високорівнева, Python і R є інтерпретованими високорівневими мовами. Однак, незважаючи на швидкість, Java може потребувати більш об’ємного коду, ніж Python. Також, з наявності сучасних обчислювальним можливостей комп’ютерів, різниця у швидкості може бути майже або непомітною, все також може залежати від використаних бібліотек;

– наявність спеціалізованих бібліотек для машинного навчання та візуалізації даних: тут лідером є Python – наявні багато бібліотек, таких, як TensorFlow, Scikit-learn, Keras, MatPlotLib, Seaborn та багато інших. Java має трохи менше бібліотек для машинного навчання і візуалізації даних – основні: Weka, DeepLearning4j, Apache OpenNLP, JFreeChart. Так як мова R призначена

для аналізу та візуалізації даних, то також має такі – ggplot2, Plotly, lattice, Ranger;

– зручність для розв’язання математичних проблем: тут лідерами є Python та R, бо перша має такі бібліотеки, як NumPy та SciPy, а друга оптимізована для статистичних обчислень та математичних моделей. Java має бібліотеки для математичних обчислень, наприклад – Apache Commons Math, але є менш популярною для виконання цих завдань;

– зручність для розширення системи та поєднання з іншими мовами програмування: тут лідерами є Python та Java;

– наявність середовищ розробки: Python має багато IDE(Integrated Development Environment) – Jupyter Notebook, PyCharm, VS (Visual Studio), Spyder та інші. Найпопулярнішими для Java є Eclipse, NetBeans, Intellij IDEA. R в основному використовують з потужним початковим середовищем – RStudio.

Таким чином нижче представлено порівняння мов програмування Python, Java та R за вибраними критеріями для використання у розроблюваному ПЗ (табл. 2.1).

Таблиця 2.1 – Порівняння мов програмування за критеріями

Критерій	Мова програмування		
	Python	Java	R
1	2	3	4
Швидкість	+	++	+
Бібліотеки машинного навчання	++	++	++
Розв’язання математичних задач	++	+	++

Продовження таблиці 2.1

1	2	3	4
Потенціал розширення	++	++	+
Середовища розробки	++	+	+

Таким чином, проаналізувавши результати, Python було обрано мовою програмування, яка найбільше підходить саме під цей проєкт.

2.2.2 Вибір середовища розробки

PyCharm – це IDE для data science та веброзробки на Python від JetBrains. У PyCharm за замовчуванням є все необхідне, щоб якісно та зручно писати код, як новачкам, так і професійним програмістам. Середовище розробки надає повну підтримку у написанні коду на Python – контекстне автодоповнення, підказки типів, підсвічування синтаксису, перевірку на помилки, швидкий перегляд документації, також оснащений можливостями налагодження, інтеграцією з контролем версій та різноманітними бібліотеками (в тому числі, і для машинного навчання), інструментами для роботи з базами даних, інструменти для виконання коду [32]. Не зважаючи на наявність ліцензії, існує безкоштовна версія з тими ж самими, але дещо обмеженими функціями.

Для порівняння середовищ розробки для Python розглянемо наступні: PyCharm, VS, Spyder.

Нижче буде представлено порівняння середовищ розробки для Python за такими факторами:

– вимоги до ресурсів: Spyder потребує найменшої кількості ресурсів, оскільки PyCharm та VS – це IDE з більш розширеними можливостями, які зазвичай використовують для більш великих проєктів;

– спеціалізація для Python: тут лідером є PyCharm та Spyder. Не зважаючи на те, що середовище PyCharm підтримує і такі мови, як SQL, JavaScript, CSS, HTML, TypeScript та інші, першочерговою його специфікацією є робота з Python. VS не є спеціалізованим середовищем для Python та також підтримує багато мов програмування. Spyder спеціалізується на Python для роботи з науковими обчисленнями;

– розширені можливості редагування коду для Python, автодоповнення, відлагодження та інші: PyCharm є відмінним вибором для Python, VS теж має вищезгадані особливості. Spyder має менше таких можливостей, бо є легким IDE, спеціально розробленим для наукової розробки на Python, тоді як PyCharm вимагає більше ресурсів і є набагато потужнішим для програмування. Таким чином, Spyder підходить для спеціалізованої наукової розробки, проте є поганим вибором для роботи з великими проєктами;

– інтеграція з системами контролю версій: PyCharm та VS мають вбудовану та відмінну підтримку Git та інших систем контролю версій, а от Spyder не має такої вбудованої підтримки;

– підтримка розширень: VS та PyCharm мають багато розширень, а от у Spyder їх обмежена кількість;

– інтеграція з бібліотеками машинного навчання: лідером є PyCharm з вбудованою і спеціалізованою підтримкою бібліотек машинного навчання. VS потребує встановлення необхідних розширень для цього. Хоч Spyder і спеціалізується на науковій розробці, там присутня і підтримка бібліотек машинного навчання.

Таким чином представлено порівняння середовищ розробки для Python за вищезгаданими критеріями (табл. 2.2).

Як можна побачити, кожне середовище розробки знаходить своє застосування і його вибір залежить від цілей та задач, які перед собою ставить розробник. Таким чином, проаналізувавши результати, PyCharm було обрано середовищем розробки, яке найбільше підходить саме під цей проєкт.

Таблиця 2.2 – Порівняння середовищ розробки

Критерій	Середовище розробки		
	PyCharm	VS	Spyder
Вимоги до ресурсів	Великі(±)	Великі(±)	Низькі(++)
Спеціалізація для Python	++	+	++
Розширені можливості	++	++	±
Інтеграція з системами контролю версій	++	++	±
Підтримка розширень	++	++	±
Інтеграція з бібліотеками машинного навчання	++	+	++

2.2.3 Вибір бібліотек для роботи

2.2.3.1 Вибір бібліотеки для API запитів

Для здійснення API запитів було вирішено використовувати бібліотеку requests. Requests – це популярна бібліотека, яка допомагає полегшити здійснення HTTP-запитів. Вона дозволяє виконувати будь-які налаштування заголовків, файлів cookie та авторизації. Також, requests надає методи для отримання певного вмісту сторінки, наприклад, JSON, при надсиланні HTTP-запитів за допомогою Python. HTTP-запит повертає об’єкт Response з усіма даними відповіді (вміст, кодування, статус тощо) [33].

2.2.3.2 Вибір бібліотеки для машинного навчання

Для побудови моделей машинного навчання було обрано бібліотеку Scikit-learn, оскільки вона надає потрібні проєкту алгоритми.

З моменту випуску в 2007 році scikit-learn стала однією з найпопулярніших бібліотек машинного навчання. Scikit-learn надає алгоритми для завдань машинного навчання, включаючи класифікацію, регресію, кластеризацію, зменшення розмірності. Вона також надає модулі для попередньої обробки даних, оцінки моделей та іншого [34].

Scikit-learn побудовано на бібліотеках NumPy та SciPy. NumPy розширює Python для підтримки ефективних операцій з великими масивами та багатовимірними матрицями. SciPy надає модулі для наукових обчислень. Scikit-learn часто використовують у парі з бібліотекою для візуалізації matplotlib [34].

Серед переваг цієї бібліотеки можна виділити:

– наявність зрозумілої документації;

– наявність великої кількості алгоритмів машинного навчання;

– простота у використанні;

– можливість експериментувати з результатами алгоритму, змінивши лише декілька рядків коду;

– наявність оболонок для scikit-learn у інших бібліотеках Python;

– вбудовані набори даних, що дозволяють розробникам більше зосередитись на алгоритмах, ніж на отриманні та очищенні даних [34].

2.2.3.3 Вибір бібліотеки для візуалізації результатів

Для візуалізації результатів було обрано бібліотеку Seaborn, яка побудована на основі matplotlib.

Seaborn – це бібліотека для візуалізації даних у Python, яку побудовано на основі Matplotlib. Вона представляє інтерфейс високого рівня для створення

інформативної та привабливої статистичної графіки [35].

Вона пропонує низку методів візуалізації для статистичної графіки, зокрема:

– однофакторні та двофакторні графіки: гістограми, квадратичні графіки, діаграми розсіювання;

– регресія та категоріальні графіки: лінійна регресія, логістична регресія, категоричні діаграми розсіювання та стовпчасті діаграми;

– матричні графіки: теплові карти, карти кластерів;

– графіки часових рядів [35].

Seaborn розроблено для бездоганної роботи з Pandas, тож вона може обробляти великі та складні набори даних. Seaborn також дозволяє кастомізувати графіки, включаючи палітри, кольори, теми та стилі.

Загалом, Seaborn – потужна та зручна бібліотека для створення інформативної та привабливої графіки [35].

Серед переваг можна виділити:

– пропонує низку візуалізаційних технік, які оптимізовані для створення інформативної та привабливої графіки з налаштуванням кастомізації для кольорів, стилів та тем;

– зручний інтерфейс розроблено таким чином, щоб бути легким у використанні. Він дозволяє створювати складні візуалізації за допомогою лише кількох рядків коду;

– наявність низки вбудованих наборів даних, які можна використовувати для практики чи дослідження;

– інтеграція з Pandas;

– широкий спектр методів візуалізації [35].

До недоліків можна віднести:

– обмеженість у візуалізації даних: бібліотека призначена для візуалізації статистичних даних і не така гнучка для даних загального призначення;

– може бути важкою для навчання без знань бібліотеки Pandas чи

концепцій статичної візуалізації;

– обмеженість у параметрах налаштування графіків: хоча спектр параметрів доволі широкий, порівняно, наприклад, з Matplotlib, він може здаватись малим [35].

2.3 Вимоги до технічних засобів

Комп’ютер користувача для коректної роботи з системою має відповідати таким вимогам:

– мати стабільне підключення до мережі Інтернет;
– операційна система: Windows 10;
– версія Python(>=3.12.1) та pip(>=23.3.2);
– вільний дисковий простір – 1 ГБ;
– оперативна пам’ять: мінімум 8 ГБ;
– процесор: не менше 4 ядер з тактовою частотою не менше 3,4 ГГц.

2.4 Висновки за розділом

У другому розділі було проаналізовано технічні засоби, які будуть використовуватися під час розробки ПЗ. Середою розробки було обрано PyCharm через спеціалізацію під Python, потужність та інтеграцію з бібліотеками машинного навчання. У якості мови програмування було обрано Python з бібліотекою машинного навчання scikit-learn, через потужну базу вбудованих функцій та зручність у використанні. Для візуалізації даних було обрано бібліотеку Seaborn через потужність та зручність у створенні інформативної графіки. Для API запитів було обрано бібліотеку requests.

3 РОЗРОБКА ПРОГРАМИ

У цьому розділі було оглянуто алгоритм роботи програми, представлено опис особливостей реалізованих функцій, а також показано процес створення файлу залежностей.

3.1 Алгоритм роботи програми

Програму було розроблено для аналізу та прогнозування злочинності на основі даних, взятих з поліцейських департаментів.

Користувач має змогу побачити процес:

– реального отримання даних за допомогою API запитів: відокремлення даних, обробку статус коду;

– попередньої обробки даних: створення додаткових атрибутів, видалення непотрібних, перетворення типів даних, обробку пропущених значень, кодування міток;

– відбору атрибутів за допомогою фільтраційних та вбудованих методів: відображення обраних ознак, матриці кореляції методом Пірсона, дерев рішень та випадкового лісу, візуалізація і порівняння важливостей ознак методів дерев рішень і випадкового лісу, порівняння відібраних ознак;

– класифікації методами випадкового лісу, k-найближчих сусідів та мультиноміальної регресії: відокремлення необхідних атрибутів, розбиття вибірок, семплінгу екземплярів, тренування та тестування моделей і їх порівняння за метриками, знаходження відхилення від середнього результату точності, для методу k-найближчих сусідів вирахування ідеального значення k для параметру точності;

– візуалізація патернів даних (наприклад, кількості злочинів по роках).

3.2 Опис реалізованих функцій програми

Основним файлом для роботи є main.py. Структуру програми реалізовано за допомогою функцій. Нижче представлено їх перелік(за черговістю виконання) і опис:

– data_load(): функція, реалізована для отримання даних за допомогою API запитів з параметрами. Якщо повернутий статус код дорівнює 200, тобто все гаразд, то дані буде записано у датафрейм, який після цього буде повернуто. Якщо код статусу відрізняється, то на екран буде виведено повідомлення про помилку, а роботу програми буде зупинено. При успішному завершенні роботи функції у консоль буде виведено відповідне повідомлення;

– data_preprocess(dataframe): функція, реалізована для попередньої обробки даних. Вона викликається тільки якщо було отримано датафрейм з попередньо викликаної функції – data_load(). Як аргумент, вона приймає датафрейм, який було попередньо отримано з функції data_load(). Всередині неї відбувається створення додаткових атрибутів, видалення непотрібних, перетворення типів даних, обробка пропущених значень, кодування міток. Також, у консоль виводяться повідомлення про етапи обробки. У кінці функція повертає оброблений датафрейм;

– feature_selection(dataframe): функція, реалізована для відбору атрибутів. Як аргумент вона приймає оброблений попередньою функцією датафрейм Всередині себе вона викликає інші дві функції – flt_methods(dataframe) та emd_methods(dataframe), у яких і відбувається сам відбір. У консоль виводяться повідомлення про результати фільтрації – атрибути, їх кількість, чи співпадають вони у різних методів та висновки;

– flt_methods(dataframe): функція, реалізована для відбору атрибутів, а також побудови матриці кореляції методом Пірсона, і яка викликається з функції feature_selection(). Як аргумент, вона приймає датафрейм. Всередині відбувається розбиття датафрейму на незалежні атрибути(або ознаки) та цільовий атрибут – тобто залежне значення, яке

потрібне спрогнозувати, відбір семи атрибутів з найбільшою кореляцією та їх вивід у консоль, будується кореляційна матриця за методом Пірсона, яка візуалізується, як карта тепла(heatmap). У кінці своєї роботи функція повертає список відібраних ознак;

– emb_methods(dataframe): функція, реалізована для відбору атрибутів вбудованими методами дерев рішень та випадкового лісу, і яка викликається з функції feature_selection(). Як аргумент, вона приймає датафрейм. Всередині відбувається розбиття датафрейму на незалежні атрибути(або ознаки) та цільовий атрибут, розбиття вибірки на тренувальну та тестувальну, тренування моделі, вивід відібраних атрибутів у консоль, візуалізація на графіку порівняння важливостей ознак, які були відібрані цими двома методами. У кінці роботи функція повертає два списки відібраних ознак;

– classification(dataframe): функція, реалізована для застосування алгоритмів класифікації. Як аргумент, вона приймає датафрейм. Всередині себе вона викликає дві функції різних алгоритмів класифікації – rf_classifier() та knn_classifier(). Функція повертає списки, які отримано з функцій rf_classifier() та knn_classifier();

– rf_classifier(dataframe): функція, реалізована для застосування алгоритму класифікації випадковий ліс. Як аргумент, вона приймає датафрейм. Всередині відбувається розбиття датафрейму на незалежні атрибути(або ознаки) та цільовий атрибут, видалення непотрібних атрибутів, розбиття вибірки на тренувальну і тестову, навчання моделей, прогнозування без семплінгу та з ним (oversampling, undersampling – методи для роботи з незбалансованими наборами даних), вивід у консоль і порівняння метрик частки успішності та помилки, точності, чутливості та середнього гармонійного для цих моделей, знаходиться відхилення поточної частки успішності від середньої частки успішності. Функція повертає список зі значенням найбільшої частки успішності та рядка, що показує якій з моделей це значення належить – з семплінгом чи без;

– knn_classifier(dataframe): функція, реалізована для застосування алгоритму класифікації k-найближчих сусідів. Як аргумент, вона приймає датафрейм. Всередині відбувається розбиття датафрейму на незалежні атрибути(або ознаки) та цільовий атрибут, видалення непотрібних атрибутів, навчання моделі та прогнозування без семплінгу та з ним, вивід у консоль метрик частки успішності та помилки, точності, чутливості та середнього гармонійного, знаходиться відхилення поточної частки успішності від середньої частки успішності, а також відбувається пошук ідеального k – тобто кількості сусідів, що виводиться у консоль. Функція повертає список зі значенням найбільшої частки успішності та рядка, що показує якій моделі це значення належить – з семплінгу чи без;

– regression(dataframe): функція, реалізована для застосування алгоритму мультиноміальної логістичної регресії. Як аргумент, вона приймає датафрейм. Всередині відбувається виклик функції multinomial_regr(). Функція повертає список, який отримано з функції multinomial_regr();

– multinomial_regr(dataframe): функція, реалізована для застосування алгоритму мультиноміальної логістичної регресії. Як аргумент, вона приймає датафрейм. Всередині відбувається видалення непотрібних атрибутів, розбиття вибірки на тренувальну та тестувальну, навчання моделей, прогнозування без семплінгу та з ним, вивід у консоль метрик частки успішності та помилки, точності, чутливості та середнього гармонійного для цих моделей, знаходиться відхилення поточної частки успішності від середньої частки успішності. Функція повертає список зі значенням найбільшої частки успішності та рядка, що показує якій з моделей це значення належить – з семплінгом чи без;

– accuracy_comparison(rf_lst, knn_lst, mlt_lst): функція, реалізована для візуалізації значень часток успішності для усіх трьох застосованих алгоритмів: випадкового лісу, k-найближчих сусідів та мультиноміальної логістичної регресії. Як аргументи, вона приймає списки зі значенням часток успішностей та рядків, що показують чи було використано семплінг для

кожного з цих алгоритмів.

– patterns(df): функція, реалізована для побудови графіків, які відображають певні патерни у наборі даних. Як аргумент, приймає датафрейм.

3.3 Опис особливостей реалізованих функцій

На рисунку 3.1 представлено функцію data_load().

```
def data_load():
    my_limit = 1000
    my_offset = 47150
    temp_dfs = list()
    i = 0
    while i < 20:
        url = 'https://data.lacity.org/resource/2nrs-mtv8.json?$limit={}&$offset={}'
        endpoint = url.format(my_limit, my_offset)
        response = requests.get(endpoint)
        if response.status_code == 200:
            data = response.json()
            temp_dfs.append(pd.DataFrame(data))
        else:
            print('Error has occurred. Please try again')
            return 0
        i += 1
        my_offset += 47150
    big_df = pd.concat(temp_dfs)
    return big_df
```

Рисунок 3.1 – Функція data_load()

Дані, які необхідні для роботи програми, завантажуються за допомогою API запитів. Для цього було імпортовано бібліотеку requests. Змінні my_limit та my_offset використовуються, як параметри у запиті. Вони визначають максимальну кількість записів, яку буде повернуто за раз, і зсув від початку набору даних. Так як оригінальний набір даних дуже великий і налічує 944 000 записів, то було прийнято рішення завантажити лише 20 000. При чому, було збережено приблизний розподіл кількості злочинів по роках, щоб дані були відібрані без упереджень. Якщо код статусу запиту дорівнює 200, то повернуті дані записуються до тимчасового списку у вигляді датафреймів(для чого було імпортовано бібліотеку pandas). Якщо виникла помилка, то відповідне

повідомлення виводиться на екран та програма припиняє роботу. У кінці програма поєднує всі малі датафрейми з тимчасового списку у один великий і повертає його.

На рисунку 3.2 представлено функцію data_preprocess().

Для того, щоб мати змогу ефективно застосувати алгоритми машинного навчання, отримані дані треба спочатку обробити. Для цього і створено цю функцію. Всередині стовбець атрибуту дати злочину спочатку перетворено на тип даних datetime. Це зроблено для того, щоб можна було зручно отримати значення року, місяця, цифри дня та день тижня. Ці стовбці додано до датафрейму. Також стовбець з часом злочину також спочатку перетворено на тип даних int. Після цього з датафрейму видаляються непотрібні стовбці – дата злочину, дата звітності, пересічна вулиця. Потім заповнюються пропущені значення, а також інші числові стовбці конвертуються до типу даних int або float. Категоріальні стовбці конвертуються за допомогою кодування міток у LabelEncoder(), що отримано з бібліотеки scikit-learn.

```
def data_preprocess(dataframe):
    pd.set_option('future.no_silent_downcasting', True)
    pd.set_option('display.max_columns', 28)
    pd.set_option('display.max_rows', 28)
    dataframe['date_occ'] = pd.to_datetime(dataframe['date_occ'])
    dataframe['Year'] = dataframe['date_occ'].dt.year
    dataframe['Month'] = dataframe['date_occ'].dt.month
    dataframe['Day'] = dataframe['date_occ'].dt.day
    dataframe['Weekday'] = dataframe['date_occ'].apply(lambda x: x.weekday())
    dataframe['Time'] = dataframe['time_occ'].astype(int)
    dataframe['Time'] = dataframe['Time'].apply(lambda x: x / 100).astype(int)

    dataframe.drop(columns=['date_occ', 'time_occ', 'date_rptd', 'part_1_2', 'cross_street'], inplace=True)
    print('\nThe unnecessary columns were removed.')
    [dataframe.pop(x) for x in ['area_name', 'mocodes', 'premis_desc', 'weapon_desc', 'status_desc', 'location']]
    dataframe.fillna({'weapon_used_cd': 0, 'crm_cd_2': 0, 'crm_cd_3': 0, 'premis_cd': 0,
                      'vict_sex': 0, 'vict_descent': 0}, inplace=True)
    print('Missing values were filled with 0.')
    dataframe['vict_sex'] = dataframe['vict_sex'].replace({'H': 0, 'X': 0})
    dataframe['vict_descent'] = dataframe['vict_descent'].replace({'X': 0, '-': 0})

    cols_to_conv = ['dr_no', 'area', 'rpt_dist_no', 'crm_cd', 'vict_age', 'premis_cd',
                    'weapon_used_cd', 'crm_cd_1', 'crm_cd_2', 'crm_cd_3']
    dataframe[cols_to_conv] = dataframe[cols_to_conv].astype(int)
    dataframe['lat'] = dataframe['lat'].astype(float)
    dataframe['lon'] = dataframe['lon'].astype(float)
    dataframe['crm_cd_desc'] = dataframe['crm_cd_desc'].astype(str)
    print('Features were converted to int and float.')

    categorical_features = ['vict_sex', 'vict_descent', 'status']
    dataframe[categorical_features] = dataframe[categorical_features].astype(str)
    inv_values = []
    values = []
    dict_lst = {}
    encoded = []
    label_encoder = LabelEncoder()
    for feature in categorical_features:
        inv_values.append(dataframe[feature].value_counts().keys().to_list())
        dataframe[feature] = label_encoder.fit_transform(dataframe[feature])
        values.append(dataframe[feature].value_counts().keys().to_list())
    for i, inv_value_lst in enumerate(inv_values):
        for j, inv_value in enumerate(inv_value_lst):
            dict_lst.update({values[i][j]: inv_value})
        encoded.append(dict_lst.copy())
        dict_lst.clear()
    print('Labels were encoded.')

    for i in range(len(encoded)):
        print(f" Feature {categorical_features[i]}: {encoded[i]}")
    return dataframe
```

Рисунок 3.2 – Функція data_preprocess()

Функцію feature_selection() представлено на рисунку 3.3.

```
def feature_selection(dataframe):
    print('Feature selection:')
    flt_selected = flt_methods(dataframe)
    dt_selected, rf_selected = emb_methods(dataframe)
    features_selected = set(flt_selected) & set(dt_selected) & set(rf_selected)
    print(' Features selected by three methods:', features_selected)
    print(' Amount of features selected by three methods:', len(features_selected), '\n')
    print(' Feature importances and the correlation coefficients are small enough to make a
    print(' But some of the following columns will be dropped for each classification and r
```

Рисунок 3.3 – Функція feature_selection()

У цій функції відбувається виклик необхідних методів відбору ознак та порівнюються результати, робляться висновки. Як можна буде побачити, методи не зафіксували великої кореляції між атрибутами або їх важливостей. Тому було прийнято рішення власноруч видаляти непотрібні атрибути для алгоритмів.

Функцію flt_methods() представлено на рисунку 3.4.

```
def flt_methods(dataframe):
    X = dataframe.drop(columns=['crm_cd', 'crm_cd_desc'])
    y = dataframe['crm_cd']
    X.columns = [x.capitalize() for x in X.columns]
    X = X.drop(columns=['Dr_no', 'Rpt_dist_no', 'Status', 'Crm_cd_1', 'Crm_cd_2', 'Crm_cd_3'])

    fs = SelectKBest(score_func=f_regression, k=7)
    x_selected = fs.fit_transform(X, y)
    selected_features = fs.get_support()
    print(' Features selected by Pearson method:', list(X.columns[selected_features]))

    corr_matrix = X.corr(method='pearson')
    f, ax1 = plt.subplots(figsize=(13, 9))
    sns.heatmap(corr_matrix, annot=True, linewidths=.5, ax=ax1)
    ax1.set_title('Correlation matrix by Pearson\'s method', fontsize=16, pad=14)
    plt.show()
    return list(X.columns[selected_features])
```

Рисунок 3.4 – Функція flt_methods()

Ця функція призначена для відбору атрибутів і відображення матриці кореляції методом Пірсона. Спочатку датафрейм розбивається на незалежні і цільовий(залежний) атрибути. Необхідні стовбці видаляються з

датафрейму – офіційний номер файлу справи, звітній округ, статус злочину, а також коди злочину. За допомогою методу SelectKBest було відібрано 7 найбільш значущих ознак, які виведено на консоль. За допомогою методу corr() створено матрицю кореляції за методом Пірсона, яку потім візуалізовано за допомогою бібліотеки seaborn. Список значущих атрибутів повернуто назад.

Функцію emb_methods() представлено на рисунку 3.5.

```
def emb_methods(dataframe):
    X = dataframe.drop(columns=['crm_cd_desc', 'crm_cd'])
    y = dataframe['crm_cd_desc']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
    X_train.columns = [x.capitalize() for x in X_train.columns]
    X_test.columns = [x.capitalize() for x in X_test.columns]
    X_train, X_test = X_train.drop(columns=['Dr_no', 'Rpt_dist_no', 'Status', 'Crm_cd_1', 'Crm_cd_2',
                                            'Crm_cd_3']), X_test.drop(columns=['Dr_no',
                                            'Rpt_dist_no', 'Status', 'Crm_cd_1', 'Crm_cd_2', 'Crm_cd_3'])

    X_train_dt = X_train[:2000]
    X_test_dt = X_test[:2000]
    y_train_dt = y_train[:2000]
    y_test_dt = y_test[:2000]

    dt_fs = SelectFromModel(DecisionTreeClassifier(random_state=5), threshold='median')
    dt_fs.fit(X_train_dt, y_train_dt)
    dt_selected = list(X_train_dt.columns[(dt_fs.get_support())])
    dt_importances = pd.Series(dt_fs.estimator_.feature_importances_)

    rf_fs = SelectFromModel(RandomForestClassifier(n_estimators=50, random_state=5), threshold='median')
    rf_fs.fit(X_train, y_train)
    rf_selected = list(X_train.columns[(rf_fs.get_support())])
    rf_importances = pd.Series(rf_fs.estimator_.feature_importances_)

    print(' Features selected by Decision Tree:', dt_selected)
    print(' Features selected by Random Forest:', rf_selected)

    for_viz = pd.DataFrame()
    for_viz['Features'] = X_train.columns
    for_viz['Dec_tree'] = dt_importances
    for_viz['Ran_forest'] = rf_importances

    plt.figure(figsize=(13, 9))
    bar_width = 0.35
    index = np.arange(len(for_viz['Features']))
    plt.bar(index, for_viz['Dec_tree'], bar_width, label='Dec_tree', color='#96B5DE')
    plt.bar(index + bar_width, for_viz['Ran_forest'], bar_width, label='Ran_forest', color='#F2DA8D')
    plt.xlabel('Features', fontsize=14, labelpad=8)
    plt.ylabel('Importance', fontsize=14, labelpad=5)
    plt.title('Feature importances using Decision Tree and Random Forest', fontsize=16, pad=14)
    plt.xticks(index + bar_width / 2, for_viz['Features'], rotation=45)
    plt.legend()
    plt.show()
    return dt_selected, rf_selected
```

Рисунок 3.5 – Функція emb_methods()

Цю функцію створено для відбору значущих ознак за допомогою методів дерев рішень і випадкового лісу. Спочатку датафрейм розбивається на незалежні і цільовий(залежний) атрибути, видаляються зайві стовбці. За допомогою методу train_test_split вибірка розбивається на тренувальну і

тестувальну. Це потрібно для того, щоб результати алгоритму були протестовані на раніше не бачених ним записах. Це підвищить реалістичність прогнозів. За допомогою методу SelectFromModel відбувається відбір ознак на основі їх важливості за допомогою класифікаторів Decision Tree та Random Forest. Параметр threshold дорівнює «median», що значить, що порогом для вибору ознак є медійне значення важливості ознак. Будуть відібрані тільки ті ознаки, важливість яких перевищує медійну. За допомогою методу feature_importances_ отримано числові значення важливостей ознак для обох методів, які потім візуалізовано на одному графіку. Назад функція повертає списки відібраних ознак.

Функцію classification() представлено на рисунку 3.6.

```
def classification(dataframe):
    print('Classification:')
    rf_lst = rf_classifier(dataframe)
    knn_lst = knn_classifier(dataframe)
    return rf_lst, knn_lst
```

Рисунок 3.6 – Функція classification()

У цій функції відбувається виклик необхідних функцій з алгоритмами класифікації. Функція повертає списки зі значеннями найбільших часток успішності та рядків, які показують якій з моделей це значення належить – з семплінгом чи без.

Функцію rf_classifier() представлено на рисунку 3.7.

Цю функцію створено для проведення класифікації методом випадкового лісу. Як і зазвичай, спочатку відбувається розбиття датафрейму, видалення непотрібного стовбця з номером справи, розбиття вибірки. За допомогою методів RandomOverSample та RandomUnderSample(для яких імпортовано бібліотеку imblearn) відбувається семплінг вибірки. Ми робимо це тому, що наш набір даних незбалансований – одні класи представлено великою кількістю записів, а інші дуже малою. В даному випадку

RandomOverSampler збільшує кількість записів лише для найменш представлених класів шляхом випадкового дублювання зразків, а RandomUnderSampler зменшує кількість записів у найбільш представлених класах шляхом випадкового видалення. Але, як ми зможемо побачити у результатах, що буде виведено на консоль, – у даному випадку це не зіграло великої ролі у прогнозуванні. За допомогою методу KFold вибірка розбивається на 10 піднаборів. За допомогою cross_val_score() модель буде навчено на дев'яти піднаборах і протестовано на одному. Це робиться для оцінки нашої моделі. Середня частка успішності потім виводиться у консоль і порівнюється з отриманими власноруч нами. Функція повертає список зі значенням найбільшої частки успішності та рядка, що показує якій з моделей це значення належить – з семплінгом чи без.

```
def rf_classifier(dataframe):
    X = dataframe.drop(columns=['crm_cd', 'crm_cd_desc'])
    y = dataframe['crm_cd_desc']
    X.columns = [x.capitalize() for x in X.columns]
    X = X.drop(columns=['Dr_no'])
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2)

    rf_model = RandomForestClassifier(n_estimators=50, random_state=5)
    rf_model.fit(X_train, y_train)
    y_pred = rf_model.predict(X_test)
    print('Random forest:')
    c1 = classification_report(y_test, y_pred, zero_division=1)

    over_sampler = RandomOverSampler(sampling_strategy='minority')
    under_sampler = RandomUnderSampler(sampling_strategy='majority')
    model_with_sampling = Pipeline([('over_sampler', over_sampler), ('under_sampler', under_sampler), ('lr', rf_model)])
    model_with_sampling.fit(X_train, y_train)
    y_pred_sampling = model_with_sampling.predict(X_test)
    c2 = classification_report(y_test, y_pred_sampling, zero_division=1)

    c1_lines = c1.split('\n')
    c1_class_names = []
    c1_class_recall = []
    for line in c1_lines[2:-4]:
        values = line.split()
        if len(values) >= 3:
            c1_class_names.append(values[:-4])
            c1_class_recall.append(float(values[-3]))

    c2_lines = c2.split('\n')
    c2_class_names = []
    c2_class_recall = []
    for line in c2_lines[2:-4]:
        values2 = line.split()
        if len(values2) >= 3:
            c2_class_names.append(values2[:-4])
            c2_class_recall.append(float(values2[-3]))

    df_metrics = pd.DataFrame()
    df_metrics['Average metrics'] = ['Accuracy', 'Missclassification', 'Precision', 'Recall', 'F1-score']
    df_metrics['No_sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred)),
                                 '{0:.4f}'.format(zero_one_loss(y_test, y_pred)),
                                 '{0:.4f}'.format(precision_score(y_test, y_pred, average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(recall_score(y_test, y_pred, average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(f1_score(y_test, y_pred, average='weighted'))]
    df_metrics['Sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(zero_one_loss(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(precision_score(y_test, y_pred_sampling, average='weighted', zero_division=1)),
                              '{0:.4f}'.format(recall_score(y_test, y_pred_sampling, average='weighted', zero_division=1)),
                              '{0:.4f}'.format(f1_score(y_test, y_pred_sampling, average='weighted'))]
    df_metrics.set_index('Average metrics', inplace=True)
    print(' ', df_metrics, '\n')
    sum_c1, sum_c2, sum_ = 0.0, 0.0, 0.0
    for s in c1_class_recall:
        sum_c1 += s
    for s in c2_class_recall:
        sum_c2 += s
    sum_ = sum_c1 - sum_c2
    print('No_sampling dataset recall deviation from sampling_dataset:', sum_)
    if sum_ > 0:
        print('Recall in the no_sampling dataset is higher than in the sampling_dataset.\n')
    else:
        print('Recall in the no_sampling dataset is lower than in the sampling_dataset.\n')

    print('Counting CV scores...')
    k_folds = KFold(n_splits=10)
    scores = cross_val_score(rf_model, X, y, cv=k_folds)
    print(' Cross Validation Scores: ', list(scores))
    print(' Average CV Score: ', '{0:.4f}'.format(scores.mean()))
    print(' Deviation from average cv score:', '{0:.4f}'.format(accuracy_score(y_test, y_pred) - scores.mean()), '\n')
    if (float(df_metrics.loc['Accuracy', 'No_sampling']) - float(df_metrics.loc['Accuracy', 'Sampling'])) > 0:
        return [df_metrics.loc['Accuracy', 'No_sampling'], 'No_sampling rf']
    else:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'Sampling rf']
```

Рисунок 3.7 – Функція rf_classifier()

Функцію knn_classifier() представлено на рисунку 3.8.

```
def knn_classifier(dataframe):
    X = dataframe.drop(columns=['crm_cd', 'crm_cd_desc'])
    y = dataframe['crm_cd']
    X.columns = [x.capitalize() for x in X.columns]
    X = X.drop(columns=['Dr_no', 'Rpt_dist_no', 'Status', 'Crm_cd_2', 'Crm_cd_3'])
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2)
    knn = KNeighborsClassifier(n_neighbors=15, weights='distance')
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)

    over_sampler = RandomOverSampler(sampling_strategy='minority')
    under_sampler = RandomUnderSampler(sampling_strategy='majority')
    model_with_sampling = Pipeline([('over_sampler', over_sampler), ('under_sampler', under_sampler), ('lr', knn)])
    model_with_sampling.fit(X_train, y_train)
    y_pred_sampling = model_with_sampling.predict(X_test)

    print('KNN:')

    df_metrics = pd.DataFrame()
    df_metrics['Metrics'] = ['Accuracy', 'Missclassification', 'Precision', 'Recall', 'F1-score']
    df_metrics['No_sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred)),
                                 '{0:.4f}'.format(zero_one_loss(y_test, y_pred)),
                                 '{0:.4f}'.format(precision_score(y_test, y_pred, average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(recall_score(y_test, y_pred, average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(f1_score(y_test, y_pred, average='weighted'))]
    df_metrics['Sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(zero_one_loss(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(precision_score(y_test, y_pred_sampling, average='weighted', zero_division=1)),
                              '{0:.4f}'.format(recall_score(y_test, y_pred_sampling, average='weighted', zero_division=1)),
                              '{0:.4f}'.format(f1_score(y_test, y_pred_sampling, average='weighted'))]
    df_metrics.set_index('Metrics', inplace=True)
    print(' ', df_metrics, '\n')

    print('Counting CV scores...')
    k_folds = KFold(n_splits=10)
    scores = cross_val_score(knn, X, y, cv=k_folds)
    print(' Cross Validation Scores: ', list(scores))
    print(' Average CV Score: ', '{0:.4f}'.format(scores.mean()))
    print(' Deviation from average cv score:', '{0:.4f}'.format(accuracy_score(y_test, y_pred) - scores.mean()), '\n')

    print('Calculating perfect k..')
    perfect_k = 0.0
    perfect_k_number = 0
    for k in range(1, 30):
        knn = KNeighborsClassifier(n_neighbors=k)
        scores = cross_val_score(knn, X_train, y_train, cv=5)
        if scores.mean() > perfect_k:
            perfect_k = scores.mean()
            perfect_k_number = k
    print(f' The perfect k is {perfect_k_number} with accuracy =', '{0:.4f}'.format(perfect_k), '\n')
    if (float(df_metrics.loc['Accuracy', 'No_sampling']) - float(df_metrics.loc['Accuracy', 'Sampling'])) > 0:
        return [df_metrics.loc['Accuracy', 'No_sampling'], 'No_sampling knn']
    elif (float(df_metrics.loc['Accuracy', 'No_sampling']) - float(df_metrics.loc['Accuracy', 'Sampling'])) < 0:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'Sampling knn']
    else:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'knn']
```

Рисунок 3.8 – Функція knn_classifier()

Цю функцію створено для проведення класифікації методом k-найближчих сусідів. Як і зазвичай, спочатку відбувається розбиття датафрейму, видалення непотрібних стовбців з офіційним номером файлу справи, звітнім округом, статусом злочину, а також кодами злочину, вибірка розбивається на тренувальну і тестувальну. За допомогою методів RandomOverSample та RandomUnderSample відбувається семплінг вибірки. Модель навчається і тестується, метрики виводяться у консоль. За допомогою KFold розбиваємо вибірку на 10 піднаборів і порівнюємо отримане середнє значення частки успішності з отриманими власноруч. Також відбувається пошук ідеального значення k-сусідів шляхом порівняння того ж середнього

значення частки успішності. Функція повертає список зі значенням найбільшої частки успішності та рядка, що показує якій з моделей це значення належить.

Функцію regression() представлено на рисунку 3.9.

```
def regression(dataframe):
    mlt_lst = multinomial_regr(dataframe)
    return mlt_lst
```

Рисунок 3.9 – Функція regression()

У цій функції відбувається виклик необхідної функції з алгоритмом мультиноміальної логістичної регресії. Функція повертає список зі значенням найбільшої частки успішності та рядка, що показує якій з моделей це значення належить.

Функцію multinomial_regr() представлено на рисунку 3.10.

```
def multinomial_regr(dataframe):
    print('Multinomial regression:')
    X = dataframe.drop(columns=['crm_cd', 'crm_cd_desc', 'dr_no', 'rpt_dist_no', 'status', 'crm_cd_2', 'crm_cd_3'])
    y = dataframe['crm_cd']
    X.columns = [x.capitalize() for x in X.columns]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    model = LogisticRegression(multi_class='multinomial', solver='saga', max_iter=3000, penalty='l2',
                               C=0.5, tol=0.001, random_state=42)
    model2 = LogisticRegression(multi_class='multinomial', solver='saga', tol=0.001)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    test_score = model.score(X_test, y_test)

    over_sampler = RandomOverSampler(sampling_strategy='minority')
    under_sampler = RandomUnderSampler(sampling_strategy='majority')
    model_with_sampling = Pipeline([('over_sampler', over_sampler), ('under_sampler', under_sampler), ('lr', model2)])
    model_with_sampling.fit(X_train, y_train)
    y_pred_sampling = model_with_sampling.predict(X_test)

    df_metrics = pd.DataFrame()
    df_metrics['Metrics'] = ['Accuracy', 'Missclassification', 'Precision', 'Recall', 'F1-score']
    df_metrics['No_sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred)),
                                 '{0:.4f}'.format(zero_one_loss(y_test, y_pred)),
                                 '{0:.4f}'.format(precision_score(y_test, y_pred, average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(recall_score(y_test, y_pred, average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(f1_score(y_test, y_pred, average='weighted'))]
    df_metrics['Sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(zero_one_loss(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(precision_score(y_test, y_pred_sampling, average='weighted', zero_division=1)),
                              '{0:.4f}'.format(recall_score(y_test, y_pred_sampling, average='weighted', zero_division=1)),
                              '{0:.4f}'.format(f1_score(y_test, y_pred_sampling, average='weighted'))]
    df_metrics.set_index('Metrics', inplace=True)
    print(' ', df_metrics, '\n')
    if float(df_metrics.loc['Recall', 'No_sampling']) - float(df_metrics.loc['Recall', 'Sampling']):
        print('Recall in the no_sampling dataset is higher than in the sampling_dataset.\n')
    else:
        print('Recall in the no_sampling dataset is lower than in the sampling_dataset.\n')

    print('Counting CV scores...')
    scores = cross_val_score(model, X_train, y_train, cv=3)
    print(f' Scores for each fold: {scores}')
    print(f' Mean score:', '{0:.4f}'.format(scores.mean()))
    print(' Deviation from average cv score:', '{0:.4f}'.format(test_score - scores.mean()), '\n')
    if (float(df_metrics.loc['Accuracy', 'No_sampling']) - float(df_metrics.loc['Accuracy', 'Sampling'])) > 0:
        return [df_metrics.loc['Accuracy', 'No_sampling'], 'No_sampling mlt']
    else:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'Sampling mlt']
```

Рисунок 3.10 – Функція multinomial_regr()

Цю функцію створено для проведення мультиноміальної логістичної регресії. Як і зазвичай, спочатку відбувається розбиття датафрейму, видалення непотрібних стовбців з офіційним номером файлу справи, звітнім округом, статусом злочину, а також коди злочину, вибірка розбивається на тренувальну і тестувальну. За допомогою методів RandomOverSample та RandomUnderSample відбувається семплінг вибірки. Знову відбувається порівняння двох моделей – з семплінгом та без, результат виводиться у консоль. Також, за допомогою KFold розбиваємо вибірку на 10 піднаборів і порівнюємо отримане середнє значення частки успішності з отриманими власноруч. Функція повертає список зі значенням найбільшої частки успішності та рядка, що показує якій з моделей це значення належить.

Функцію accuracy_comparison() представлено на рисунку 3.11.

```
def accuracy_comparison(rf_lst, knn_lst, mlt_lst):
    lst_x, lst_y = [], []
    lst_x.append(rf_lst[1])
    lst_x.append(knn_lst[1])
    lst_x.append(mlt_lst[1])
    lst_y.append(float(rf_lst[0]))
    lst_y.append(float(knn_lst[0]))
    lst_y.append(float(mlt_lst[0]))
    print(lst_x)
    print(lst_y)
    sns.set_theme(style='whitegrid', context='notebook')
    plt.figure(figsize=(13, 9))
    sns.barplot(x=np.array(lst_x), y=np.array(lst_y), color='#96B5DE')
    plt.xlabel('The highest accuracy of each method', fontsize=14, labelpad=8)
    plt.ylabel('Accuracy', fontsize=14, labelpad=5)
    plt.title('Accuracy comparison ', fontsize=16, pad=14)
    plt.show()
```

Рисунок 3.11 – Функція accuracy_comparison()

Цю функцію створено для візуалізації результатів часток успішностей, отриманих після використання трьох методів – випадкових лісів, k-найближчих сусідів та мультиноміальної логістичної регресії.

Частину функції patterns() представлено на рисунку 3.12. Графіки було створено за допомогою бібліотек Seaborn та Matplotlib.

```
def patterns(df):
    print('Some interesting patterns:')
    sns.set_theme(style='whitegrid', context='notebook')
    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(13, 9))
    plt.subplots_adjust(hspace=0.4, wspace=0.3)
    sns.barplot(x=np.array(df['Year'].value_counts().keys().to_list()),
                y=np.array(df['Year'].value_counts().values), ax=ax1,
                color='#96B5DE')
    ax1.set_xlabel('Year', fontsize=14, labelpad=8)
    ax1.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax1.set_title('Crime count by year', fontsize=16, pad=14)

    sns.barplot(x=np.array(df['Month'].value_counts().keys().to_list()),
                y=np.array(df['Month'].value_counts().values), ax=ax2,
                color='#F2DA8D')
    ax2.set_xlabel('Month', fontsize=14, labelpad=8)
    ax2.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax2.set_title('Crime count by month', fontsize=16, pad=14)

    sns.lineplot(x=np.array(df['Day'].value_counts().keys().to_list()),
                 y=np.array(df['Day'].value_counts().values), ax=ax4,
                 color='#7CDA97')
    ax4.set_xlabel('Day', fontsize=14, labelpad=8)
    ax4.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax4.set_title('Crime count by day', fontsize=16, pad=14)
```

Рисунок 3.12 – Функція patterns()

Цю функцію створено для візуалізації знайдених патернів у наборі даних. Для цього використовується бібліотека Seaborn.

3.4 Файл залежностей проєкту

У файлі requirements.txt було реалізовано перелік необхідних бібліотек для проєкту і їх версій. Файл було створено за допомогою бібліотеки pipreqs. Його створено для автоматизації пошуку імпортів у проєкті і створення на їх основі файлу залежностей. Цей процес представлено на рисунку 3.13.

```
PS D:\serious_stuff\uni\4.2\diploma\project> pipreqs .
INFO: Not scanning for jupyter notebooks.
WARNING: Import named "numpy" not found locally. Trying to resolve it at the PyPI server.
WARNING: Import named "numpy" was resolved to "numpy:1.26.4" package (https://pypi.org/project/numpy/).
Please, verify manually the final list of requirements.txt to avoid possible dependency confusions.
WARNING: Import named "pandas" not found locally. Trying to resolve it at the PyPI server.
WARNING: Import named "pandas" was resolved to "pandas:2.2.2" package (https://pypi.org/project/pandas/).
Please, verify manually the final list of requirements.txt to avoid possible dependency confusions.
WARNING: Import named "Requests" not found locally. Trying to resolve it at the PyPI server.
WARNING: Import named "Requests" was resolved to "requests:2.32.2" package (https://pypi.org/project/requests/).
Please, verify manually the final list of requirements.txt to avoid possible dependency confusions.
WARNING: Import named "seaborn" not found locally. Trying to resolve it at the PyPI server.
WARNING: Import named "seaborn" was resolved to "seaborn:0.13.2" package (https://pypi.org/project/seaborn/).
Please, verify manually the final list of requirements.txt to avoid possible dependency confusions.
INFO: Successfully saved requirements file in .\requirements.txt
```

Рисунок 3.13 – Створення файлу залежностей проєкту

Сам файл requirements.txt представлено на рисунку 3.14.

```
imbalanced_learn==0.12.2
imblearn==0.0
matplotlib==3.8.4
numpy==1.26.4
pandas==2.2.2
Requests==2.32.2
scikit_learn==1.4.2
seaborn==0.13.2
```

Рисунок 3.14 – Файл requirements.txt

3.5 Висновки за розділом

У цьому розділі були розглянуті ключові аспекти функціонування програми. Було прописано алгоритм роботи програми, описано та обґрунтовано особливості всіх створених функцій – методи, імпортовані бібліотеки. Також, було представлено алгоритм створення файлу requirements.txt, який допоможе користувачу у один рядок встановити всі необхідні для правильної роботи проєкту залежності.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

У цьому розділі було оглянуто можливості використання програми з точки зору звичайного користувача.

4.1 Вимоги для експлуатації

Функції системи описано у ТЗ п. 1.6.4. Вимоги до технічних засобів наведено у ТЗ п. 1.6.7.

Програму не вийде запустити одразу, якщо на комп'ютері не встановлено необхідне забезпечення.

Спочатку треба переконатись, що операційною системою є Windows 10 та присутнє стабільне підключення до мережі Інтернет. Так як дані будуть повертатись за допомогою API запитів, то нестабільне підключення може призвести до помилок у отриманні даних, а таким чином і у всій подальшій роботі програми.

Також потрібно мати вільний дисковий простір мінімум 1ГБ для встановлення необхідних бібліотек та просто роботи програми без збоїв. Процесор і оперативна пам'ять теж потрібні відповідати мінімальним вимогам – не менше 4 ядер та мінімум 8ГБ. Якщо ж комп'ютер не має таких характеристик, то запуск програми буде дуже довгим та ресурсозатратним або взагалі неможливим.

Якщо все вищезгадане наявне, то тепер потрібно скачати Python. Для цього потрібно перейти на офіційний сайт https://www.python.org/. Там у категорії «Download» вибрати останню наявну версію Python. Це показано на рисунку 4.1.

Після цього буде відкрито сторінку, де розказано про зміни у цій версії, а також представлено можливість завантажити Python. Щоб завантажити Python, натисніть на «Windows Installer(64 bit)» у розділі «Files». Це представлено на рисунку 4.2.

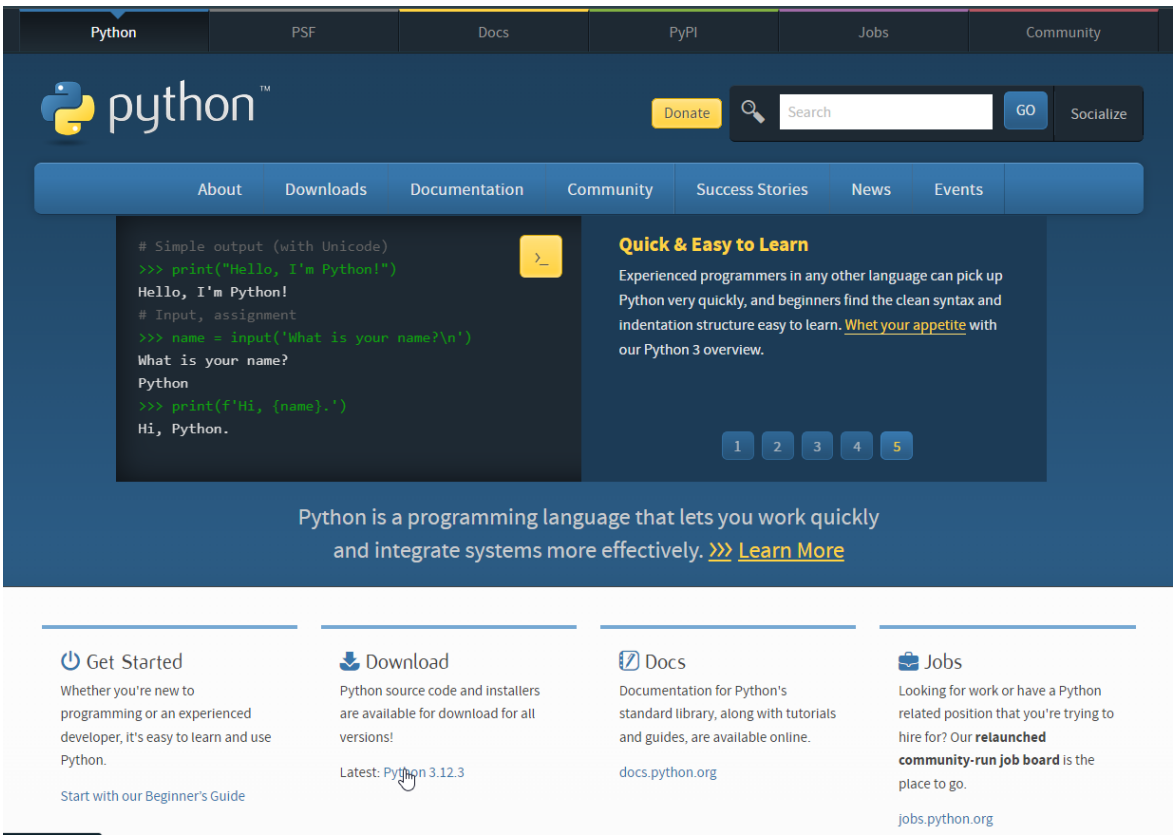
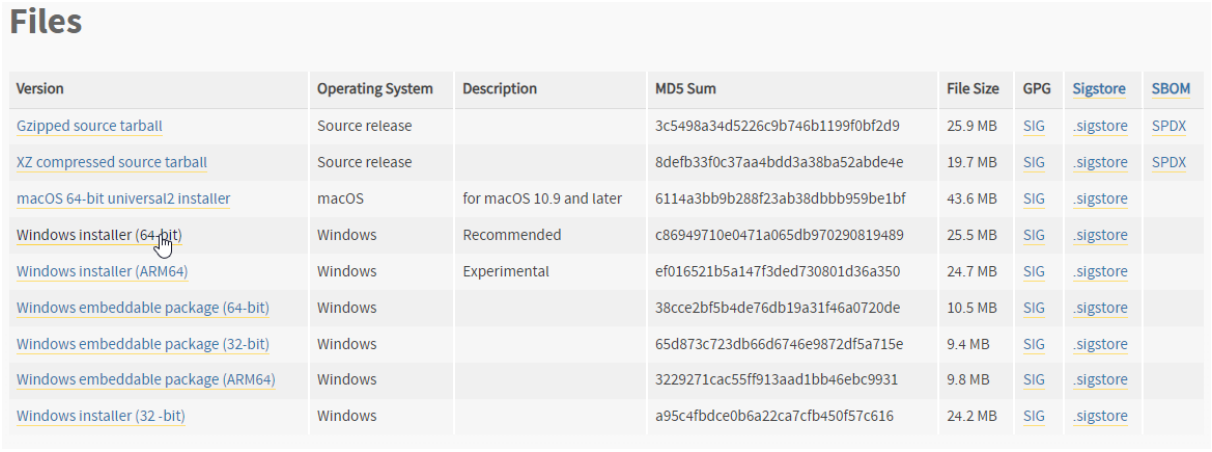


Рисунок 4.1 – Вибір останньої версії Python



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG	Sigstore	SBOM
Gzipped source tarball	Source release		3c5498a34d5226c9b746b1199f0bf2d9	25.9 MB	SIG	.sigstore	SPDX
XZ compressed source tarball	Source release		8defb33f0c37aa4bdd3a38ba52abde4e	19.7 MB	SIG	.sigstore	SPDX
macOS 64-bit universal2 installer	macOS	for macOS 10.9 and later	6114a3bb9b288f23ab38dbbb959be1bf	43.6 MB	SIG	.sigstore	
Windows installer (64-bit)	Windows	Recommended	c86949710e0471a065db970290819489	25.5 MB	SIG	.sigstore	
Windows installer (ARM64)	Windows	Experimental	ef016521b5a147f3ded730801d36a350	24.7 MB	SIG	.sigstore	
Windows embeddable package (64-bit)	Windows		38cce2bf5b4de76db19a31f46a0720de	10.5 MB	SIG	.sigstore	
Windows embeddable package (32-bit)	Windows		65d873c723db66d6746e9872df5a715e	9.4 MB	SIG	.sigstore	
Windows embeddable package (ARM64)	Windows		3229271cac55ff913aad1bb46ebc9931	9.8 MB	SIG	.sigstore	
Windows installer (32-bit)	Windows		a95c4fbdce0b6a22ca7cfb450f57c616	24.2 MB	SIG	.sigstore	

Рисунок 4.2 – Вибір інсталятора

Далі потрібно вибрати місце на комп’ютері, куди потрібно встановити файл. Після чого треба відкрити цей файл. На рисунку 4.3 показано приклад запущеного інсталятору. Кнопка «Install Now» почне завантаження Python у директорію за замовченням. Цей спосіб встановлення є рекомендованим. Проте, якщо хочеться встановити Python у іншу директорію, то потрібно натиснути «Customize installation». Перед тим, як завантажувати Python

наполегливо рекомендується поставити галочки у два чекбокси знизу – це ще зіграє свою роль у майбутньому. Після цього можна натискати кнопку «Install Now» та чекати завершення встановлення.

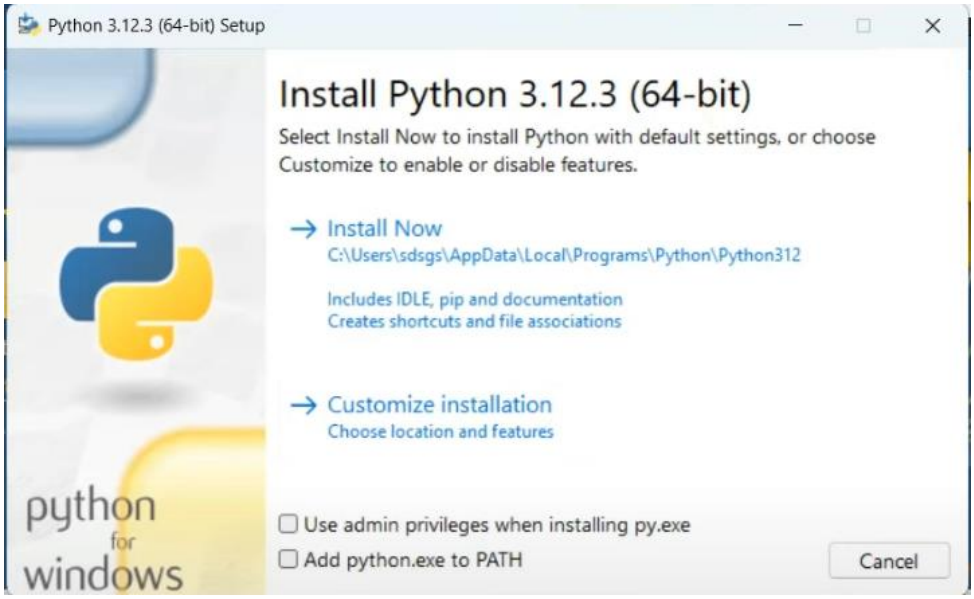


Рисунок 4.3 – Інсталятор Python

Тепер перевіримо чи дійсно Python встановлено правильно. Для цього спочатку відкриємо командний рядок. Щоб зробити це, натиснемо сполучення клавіш Win+R, після чого введемо у вікно команду cmd. Це показано на рисунку 4.4.

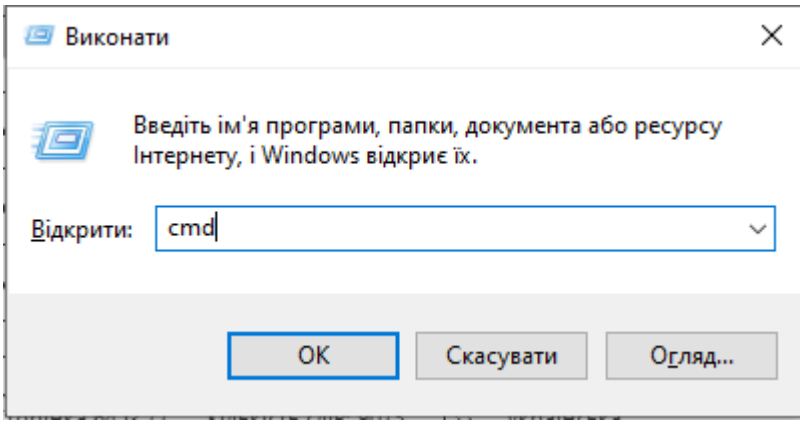


Рисунок 4.4 – Відкриття командного рядка

Після цього можна натиснути кнопку ОК і буде відкрито командний рядок, що представлено на рисунку 4.5.

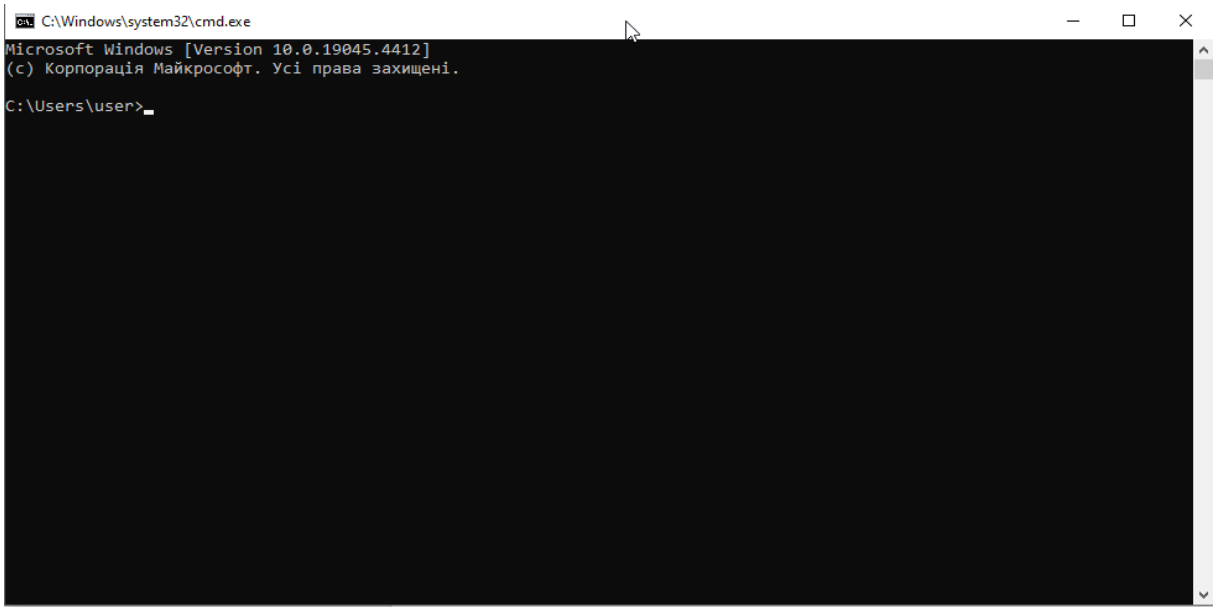


Рисунок 4.5 – Відкритий командний рядок

Наступна команда спрацює, якщо при встановленні Python було поставлено галочку у чекбокс «add python.exe to PATH». Якщо це не було зроблено, то додавати python.exe до змінних оточень доведеться самостійно. Тож, введемо у командному рядку python, що представлено на рисунку 4.6.

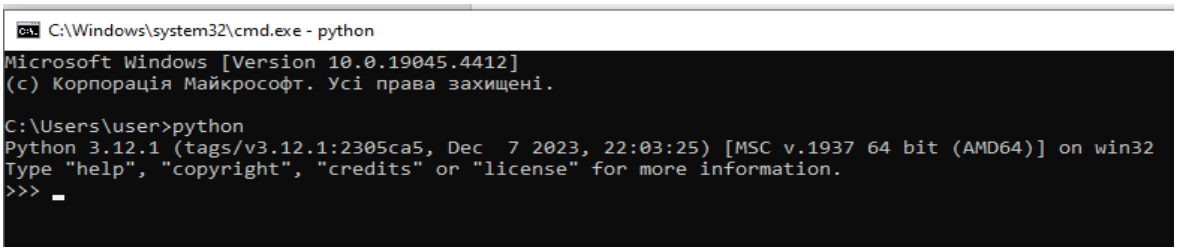


Рисунок 4.6 – Перевірка версії Python

Як можна побачити, на консоль виведено встановлену версію Python – у мене це 3.12.1. Якщо у консоль виведено рядок з помилкою, то Python було встановлено неправильно і треба повторити всі кроки.

Зверніть увагу – якщо буде помилково встановлено версію Python нижчу за 3.4, то у комплект не буде входити менеджер пакетів pip і його потрібно буде встановити самостійно. Проте рекомендується використовувати новіші версії Python.

Розглянемо структуру папки проєкту – її представлено на рисунку 4.7. Потрібно скачати цю папку собі на комп'ютер і перейти у командному рядку до цієї директорії.

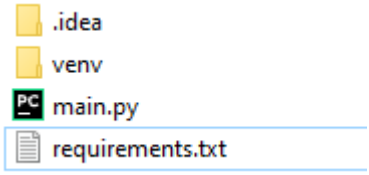


Рисунок 4.7 – Папка проєкту

До того, як запускати основний файл – main.py, потрібно встановити необхідні бібліотеки з файлу requirements.txt, без яких програма не буде працювати. Зробити це зручно і швидко можна за допомогою команди pip install -r requirements.txt у pip. Це показано на рисунку 4.8. У мене ці бібліотеки вже встановлено, що й написано у консолі, але у тих , хто вперше їх завантажує, буде розпочато процес завантаження. Дочекайтеся повідомлення про успішну інсталяцію, що буде означати, що бібліотеки встановлено успішно.

```
D:\serious_stuff\uni\4.2\diploma\project>pip install -r requirements.txt
Requirement already satisfied: imbalanced_learn==0.12.2 in c:\users\user\appdata\l
Requirement already satisfied: imblearn==0.0 in c:\users\user\appdata\local\progra
Requirement already satisfied: matplotlib==3.8.4 in c:\users\user\appdata\local\pr
Requirement already satisfied: numpy==1.26.4 in c:\users\user\appdata\local\progra
Requirement already satisfied: pandas==2.2.2 in c:\users\user\appdata\local\progra
Collecting Requests==2.32.2 (from -r requirements.txt (line 6))
  Downloading requests-2.32.2-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: scikit_learn==1.4.2 in c:\users\user\appdata\local\
Requirement already satisfied: seaborn==0.13.2 in c:\users\user\appdata\local\prog
Requirement already satisfied: scipy>=1.5.0 in c:\users\user\appdata\local\progra
Requirement already satisfied: joblib>=1.1.1 in c:\users\user\appdata\local\progra
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\user\appdata\loca
3.4.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\user\appdata\local\pro
Requirement already satisfied: cycler>=0.10 in c:\users\user\appdata\local\progra
Requirement already satisfied: fonttools>=4.22.0 in c:\users\user\appdata\local\pr
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\user\appdata\local\pr
Requirement already satisfied: packaging>=20.0 in c:\users\user\appdata\local\prog
Requirement already satisfied: pillow>=8 in c:\users\user\appdata\local\programs\p
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\user\appdata\local\pro
Requirement already satisfied: python-dateutil>=2.7 in c:\users\user\appdata\loca
ost0)
Requirement already satisfied: pytz>=2020.1 in c:\users\user\appdata\local\progra
Requirement already satisfied: tzdata>=2022.7 in c:\users\user\appdata\local\prog
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\user\appdata\l
2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\user\appdata\local\progra
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\user\appdata\local\p
Requirement already satisfied: certifi>=2017.4.17 in c:\users\user\appdata\local\p
Requirement already satisfied: six>=1.5 in c:\users\user\appdata\local\programs\py
) (1.16.0)
Downloading requests-2.32.2-py3-none-any.whl (63 kB)
   ---------------------------------------- 63.9/63.9 kB 570.9 kB/s eta 0:00:00
Installing collected packages: Requests
  Attempting uninstall: Requests
    Found existing installation: requests 2.31.0
    Uninstalling requests-2.31.0:
      Successfully uninstalled requests-2.31.0
Successfully installed Requests-2.32.2
```

Рисунок 4.8 – Встановлення бібліотек з requirements.txt

4.2 Експлуатація програми

Після успішного встановлення усього вищезгаданого можна запускати програму. Для цього потрібно у командному рядку у тій самій директорії проєкту прописати команду python main.py. Це представлено на рисунку 4.9.

```
D:\serious_stuff\uni\4.2\diploma\project>python main.py
```

Рисунок 4.9 – Запуск програми

Тут почнеться виконання програми. У консоль буде виведено повідомлення про процес виконання. Після запуску буде виведено

повідомлення про початок виконання, а після успішного отримання даних за допомогою API запитів(за це відповідає функція data_load()) буде виведено відповідний рядок. Це можна побачити на рисунку 4.10.

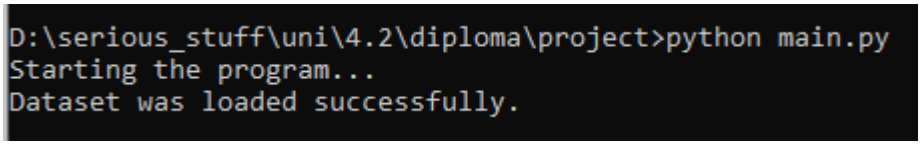


Рисунок 4.10 – Початок виконання та отримання даних

Потім у консоль буде виведено результати попередньої обробки даних, за яку відповідає функція data_preprocess(), що й представлено на рисунку 4.11. Виведено повідомлення про видалення непотрібних атрибутів, заповнення пропущених значень нулями, перетворення типів даних, успішне кодування міток, самі мітки і їх коди та повідомлення про успішну обробку даних.

Рисунок 4.11 – Попередня обробка даних

Після цього у консоль буде виведено 7 найзначніших ознак та відобразиться матриця кореляції ознак методом Пірсона, за що відповідає функція flt_methods(). Матрицю представлено на рисунку 4.12. Після її закриття програма продовжить свою роботу.

Буде виведено графік, який порівнює важливість ознак, яку вирахувано за допомогою методів дерев рішень та випадкового лісу, що прописано у функції emd_methods(). Графік представлено на рисунку 4.13. Після закриття графіка програма продовжує роботу.

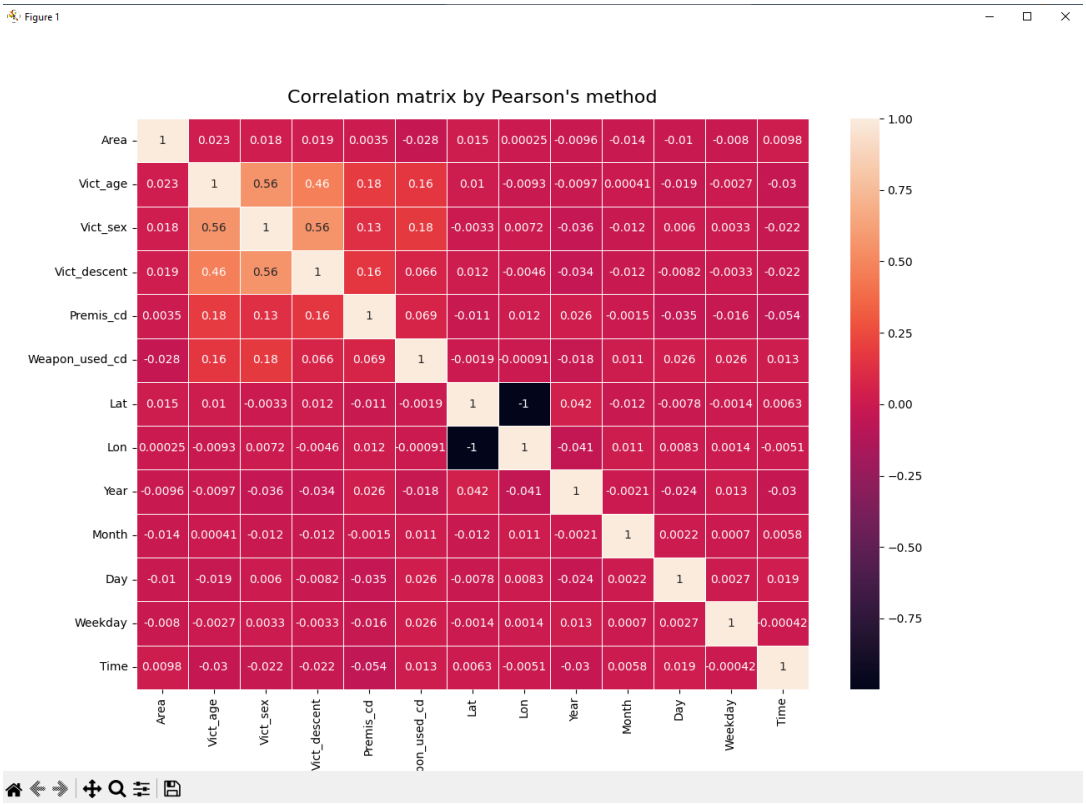


Рисунок 4.12 – Матриця кореляції ознак

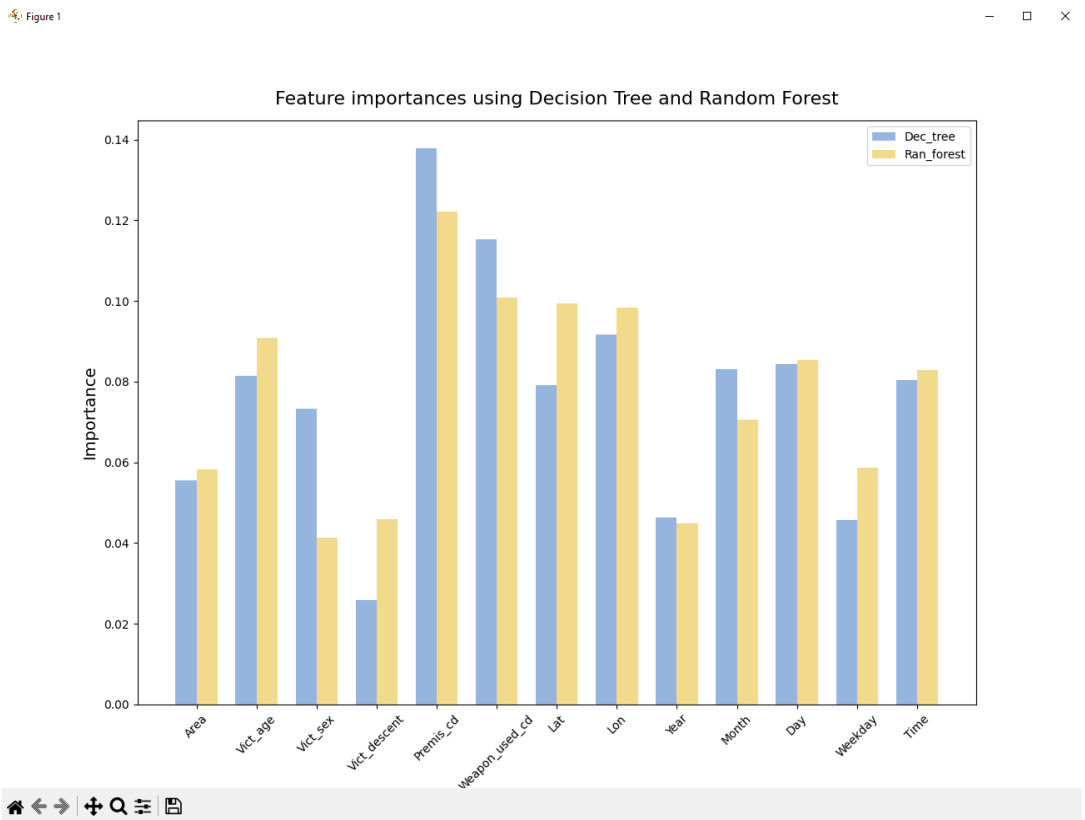


Рисунок 4.13 – Порівняння важливостей ознак

На екран буде виведено результати відбору ознак, що представлено на рисунку 4.14 – це функція feature_selection(). Методи показали низьку кореляцію між ознаками і важливість атрибутів, тому було прийнято рішення не видаляти ознаки, базуючись на їх результатах. При використанні алгоритмів машинного навчання буде видалятись частина або всі з наступних атрибутів – офіційний номер файлу справи, звітний округ, статус злочину, а також коди злочину.

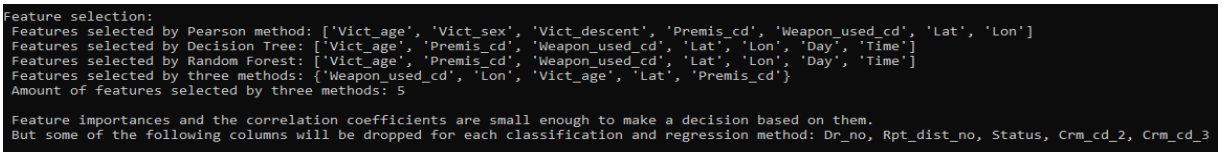


Рисунок 4.14 – Результати відбору ознак

Наступними на консоль буде виведено результати класифікації за допомогою алгоритму випадкового лісу – функція rf_classifier(). У вигляді датафрейму виведено порівняння метрик моделей без семплінгу та з ним. Як можна побачити, результати майже не змінились. Нижче вирахувано відхилення від середньої частки успішності – воно також дуже незначне, а також виведено на екран частки успішності, отримані за допомогою методу KFold(). Все це представлено на рисунку 4.15.

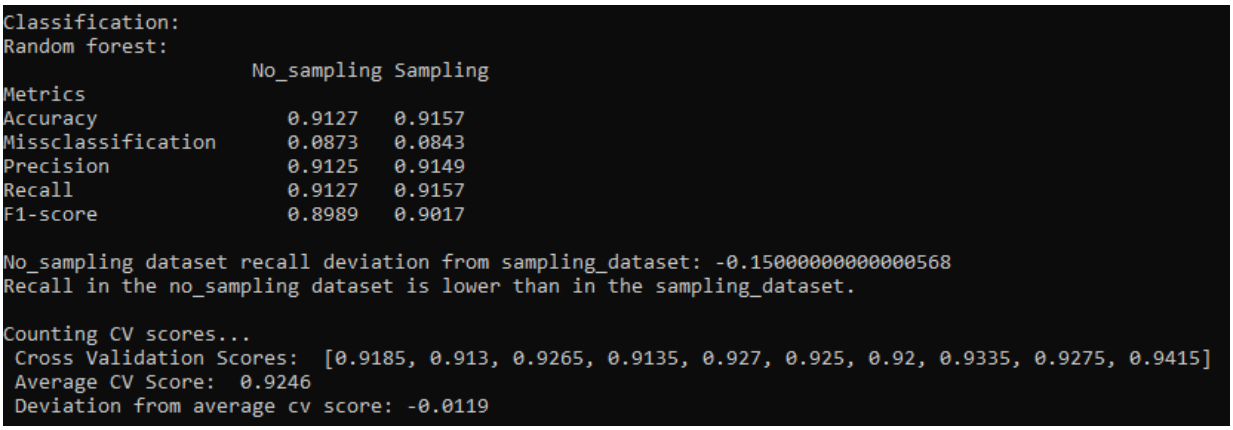


Рисунок 4.15 – Класифікація методом випадкового лісу

Після цього на консоль буде виведено результати класифікації за допомогою алгоритму k-найближчих сусідів – функція knn_classifier(). У вигляді датафрейму виведено повірняння метрик моделей без семплінгу та з ним. Відображено метрики моделі, вирахувано відхилення від середньої частки успішності – незначне, як і у попереднього алгоритму. Нижче виведено ідеальну кількість k-сусідів для цього алгоритмі, базуючись на порівнянні часток успішності. Це представлено на рисунку 4.16.

```
KNN:
                    No_sampling Sampling
Metrics
Accuracy                 0.7752   0.7752
Missclassification       0.2248   0.2248
Precision                0.7795   0.7795
Recall                   0.7752   0.7752
F1-score                 0.7548   0.7548

Counting CV scores...
 Cross Validation Scores:  [0.7655, 0.766, 0.7775, 0.77, 0.805, 0.7975, 0.7785, 0.796, 0.7875, 0.796]
 Average CV Score:  0.7840
 Deviation from average cv score: -0.0088

Calculating perfect k..
 The perfect k is 4 with accuracy = 0.7762
```

Рисунок 4.16 – Класифікація методом k-найближчих сусідів

За цим буде виведено у консоль результати класифікації за допомогою алгоритму мультиноміальної логістичної регресії – функція multinomial_regr(). У вигляді датафрейму виведено порівняння метрик моделей без семплінгу та з ним, порівняно значення чутливостей. Під кінець, як і для інших методів, вирахувано відхилення від середньої частки успішності – воно є більш значним, ніж у попередніх методів, але все ще малим. Все це представлено на рисунку 4.17.

Після цього буде візуалізовано на графіку частки успішностей кожного з виконаних раніше алгоритмів та позначенням його виду – з семплінгом чи без. Це представлено на рисунку 4.18. Для цього використано функцію accuracy_comparison().

```
Multinomial regression:
                    No_sampling Sampling
Metrics
Accuracy                 0.6538   0.5887
Missclassification       0.3462   0.4113
Precision                0.6429   0.5863
Recall                   0.6538   0.5887
F1-score                 0.5955   0.5154

Recall in the no_sampling dataset is higher than in the sampling_dataset.

Counting CV scores...
 Scores for each fold: [0.62116992 0.62695522 0.62516074]
 Mean score: 0.6244
 Deviation from average cv score: 0.0294
```

Рисунок 4.17 – Мультиноміальна логістична регресія

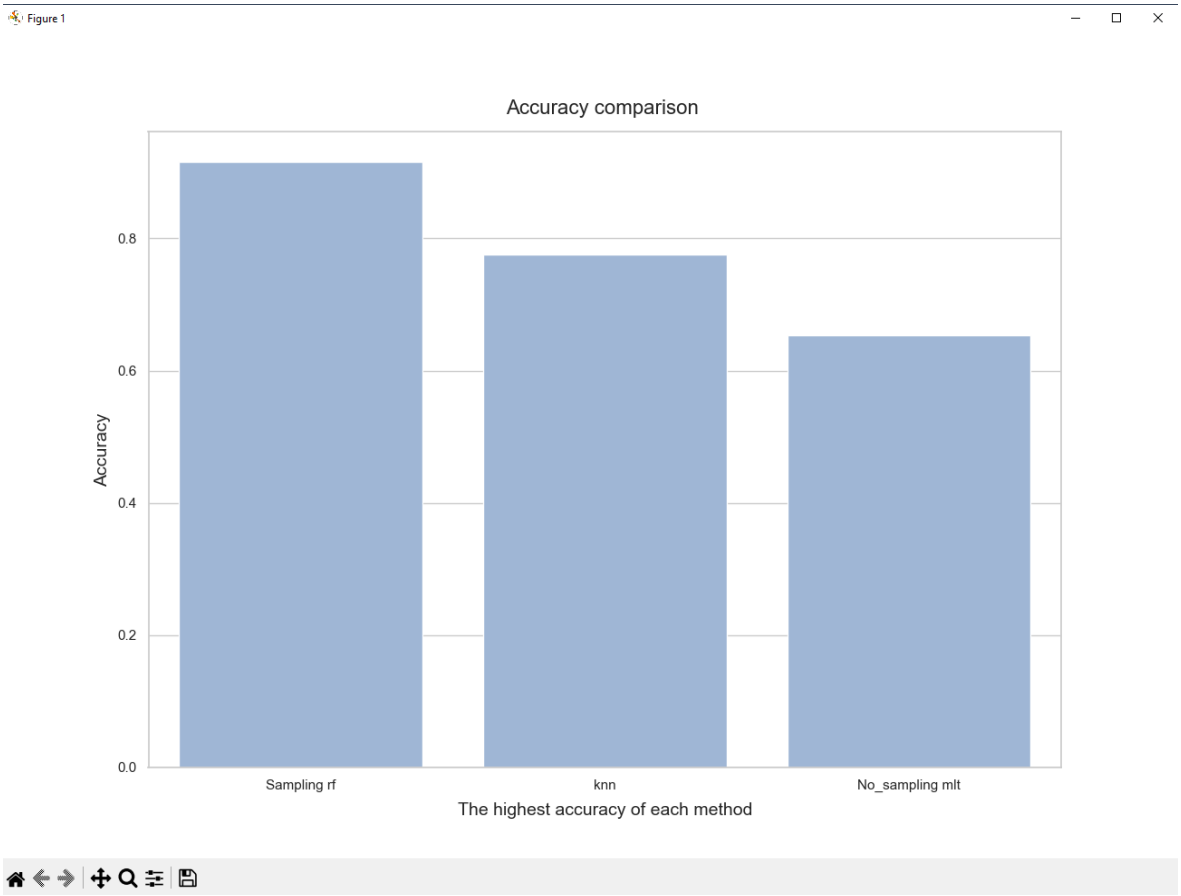


Рисунок 4.18 – Порівняння часток успішностей

У кінці буде виведено графіки, які відображають патерни злочинності з набору даних – наприклад, кількість злочинів по роках, місяцях і т.і. Їх представлено на рисунках 4.19-4.21.

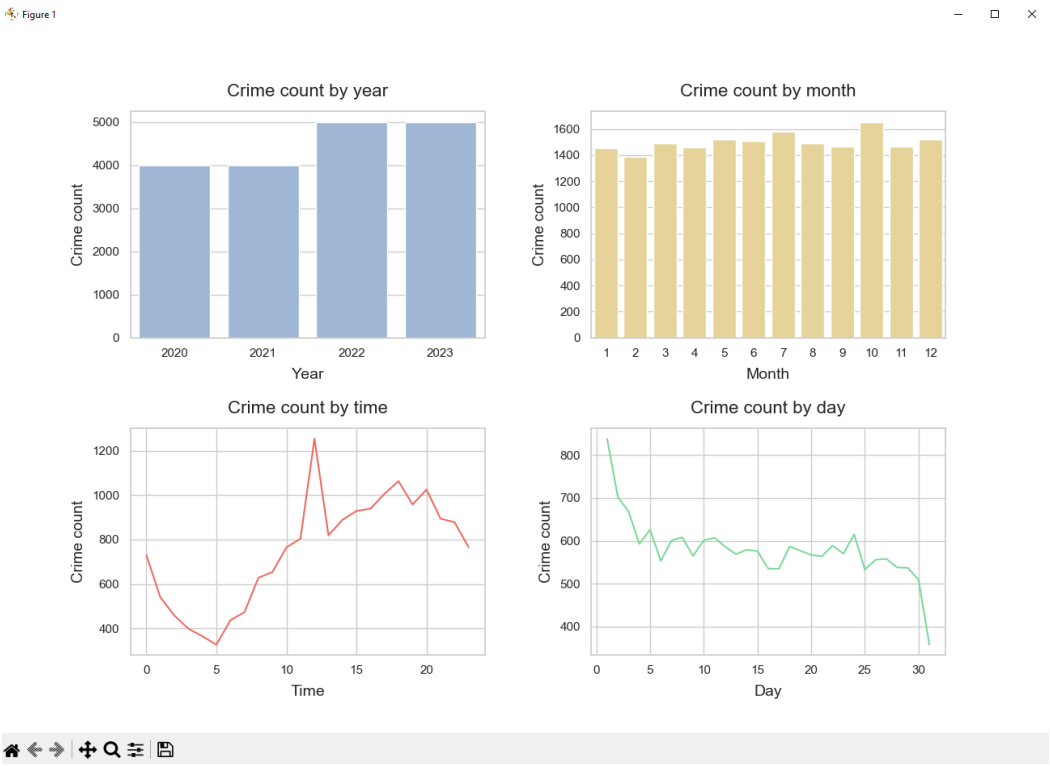


Рисунок 4.19 – Патерни злочинності

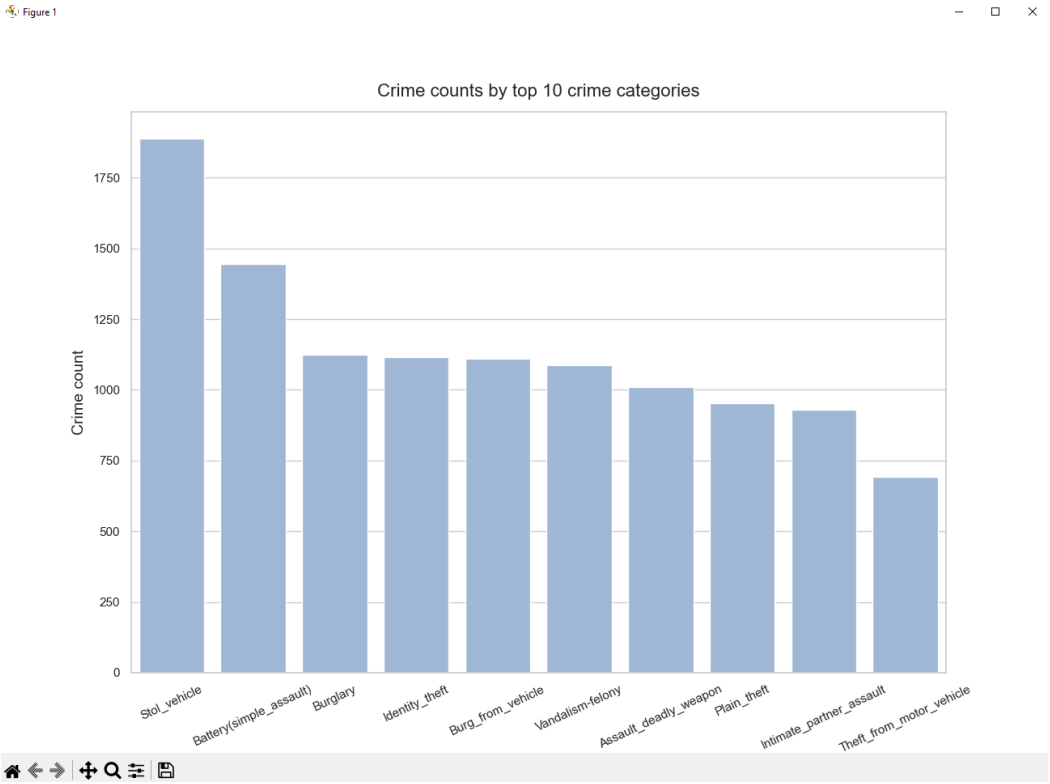


Рисунок 4.20 – Патерни злочинності

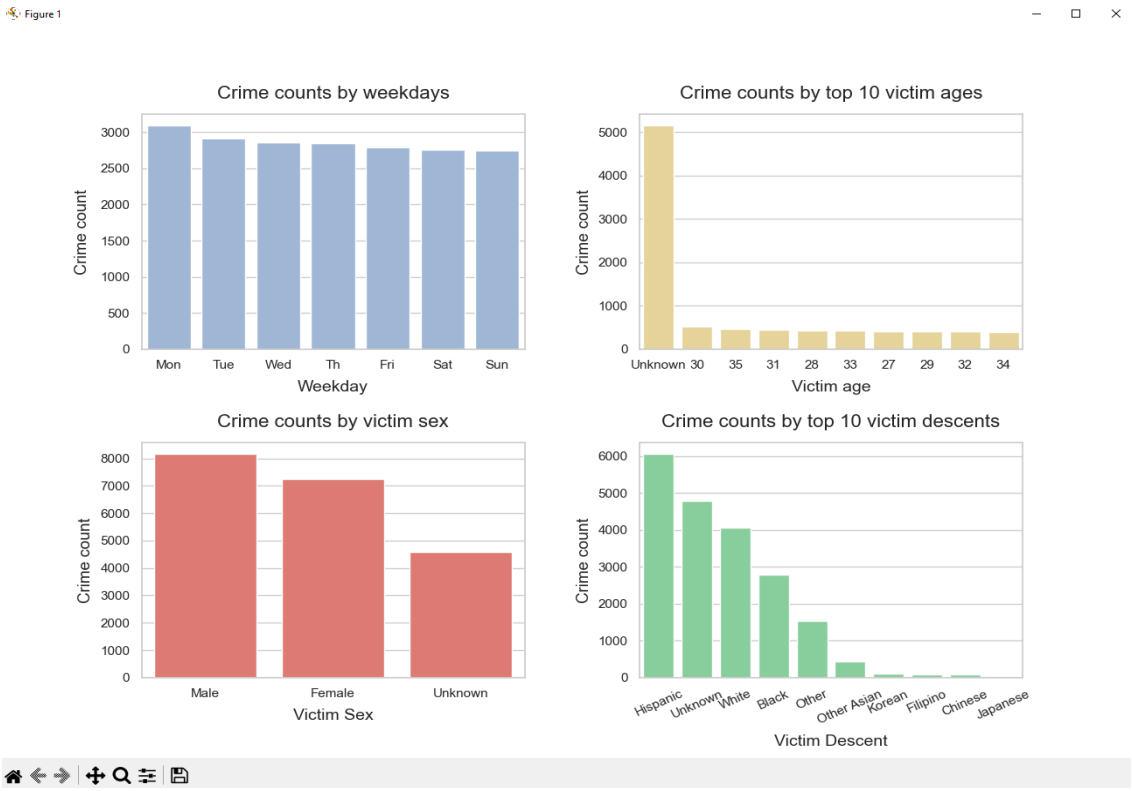


Рисунок 4.21 – Патерни злочинності

Після чого програма завершує свою роботу.

У підсумку, програмою виконано основне призначення. Вона пропонує детальну візуалізацію результатів, сама здійснює пошук даних, більш глибоко аналізує дані та може прогнозувати. Як можна побачити з результатів, трьома методами досягнуто різних значень часток успішності. Найуспішнішою виявилась модель випадкового лісу з семплінгом з часткою у 0,916, хоча і випередження моделі без семплінгу склало усього 0,003. Моделю k-найближчих сусудів показано результат у 0,775 частки успішності. Це значення не змінюється при використанні семплінга та без нього. Мультиноміальною логістичною регресією показано результат у 0,654 частки успішності. У цьому випадку розрив від набору даних без семплінгу складає вже 0,03. Найвищу чутливість також продемонстровано алгоритмом випадкового лісу. Методом KFold показано, що побудовано якісні моделі, які добре справляються зі своїми задачами.

4.3 Тестування програми

Програму було протестовано на відповідність функціональним вимогам. Вона повністю відповідає їм, тому що забезпечує:

– завантаження даних;
– попередню обробку даних;
– відбір атрибутів;
– тестування моделей алгоритмів класифікації та регресії;
– аналіз, візуалізацію і порівняння результатів;
– візуалізації патернів.

Програма виконує свої функції та працює без збоїв, функції не конфліктують одна з одною.

ВИСНОВКИ

В процесі виконання дипломної роботи, як результат, було розроблене ПЗ для аналізу та прогнозування злочинності за допомогою методів машинного навчання. Проведено порівняльне ознайомлення з аналогами, виконано аналіз технічної літератури та матеріалів, пов’язаних зі злочинністю, її впливом на суспільство, API, використанням машинного навчання – його категорій, алгоритмів, метрик. Для алгоритмів представлено пояснення, порівняння переваг та недоліків, було представлено формули для метрик та визначено технічне завдання. Також, було обрано мову програмування, середовище розробки та бібліотеки, було обґрунтовано вибір кожного пункту і порівняно можливі варіанти. Детально та в доступній формі описано процес встановлення Python та усіх необхідних бібліотек. Було визначено основні вимоги для експлуатації, проведено тестування програми, представлено повний процес експлуатації програми з необхідними скріншотами та поясненнями, підбито підсумки виконання програми.

Наукова цінність роботи полягає в тому, що було представлено ПЗ, яке вдосконалило аспекти всіх проаналізованих аналогів. Програма повністю відповідає функціональним вимогам.

Програма має практичну цінність, оскільки надає користувачам можливість мати змогу більш детально аналізувати дані, будувати більш складні моделі, прогнозувати результати, а також візуалізувати їх.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Empirical Analysis for Crime Prediction and Forecasting Using Machine Learning and Deep Learning Techniques [Electronic resource]. – Access mode: https://ieeexplore.ieee.org/abstract/document/9424589.

2. Pana T. N. Machine Learning and Crime [Electronic resource]. – Access mode: https://www.linkedin.com/pulse/machine-learning-crime-tudor-n-pana/.

3. Garnham, A. Artificial Intelligence | An Introduction [Electronic resource] / A. Garnham. – Oxfordshire : Taylor & Francis, 2017. – 300 p. – Access mode: https://www.google.com.ua/books/edition/Artificial_Intelligence/qFI8DwAAQBAJ?hl=uk&gbpv=1&dq=what+is+artificial+intelligence&printsec=frontcover.

4. Campedelli, G. M. Machine Learning for Criminology and Crime Research [Electronic resource] / G. M. Campedelli. – New York : Routledge, 2022. – 194 p. – Access mode: https://www.google.com.ua/books/edition/Machine_Learning_for_Criminology_and_Cri/lSpsEAAAQBAJ?hl=uk&gbpv=1&dq=machine+learning+in+crime&printsec=frontcover.

5. The Growing Role of AI in Criminal Justice [Electronic resource]. – Access mode: https://www.mastersinai.org/industries/criminal-justice/.

6. What Is Machine Learning (ML)? | IBM [Electronic resource]. – Access mode: https://www.ibm.com/topics/machine-learning.

7. Sahota N. Machine Learning: Definition, Types, Advantages & More [Electronic resource]. – Access mode: https://www.linkedin.com/pulse/machine-learning-definition-types-advantages-more-neil-sahota-萨冠军-/.

8. Senthilselvi, A. Machine Learning [Electronic resource] / A. Senthilselvi, B. J Chelliah, S. Senthil Pandi. – Tamil Nadu : Shanlax Publications, 2021. – 269 p. – Access mode:

https://www.google.com.ua/books/edition/Machine_Learning/vUpgEAAAQBAJ?hl=uk&gbpv=1&dq=machine+learning&printsec=frontcover.

9. Kiptoon D. Feature Selection in Machine Learning [Electronic resource]. – Access mode: https://medium.com/@jdkiptoon/feature-selection-in-machine-learning-20417d052b80.

10. Feature Selection Techniques in Machine Learning (Updated 2024) | Analytics Vidhya [Electronic resource]. – Access mode: https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/.

11. Chandra V. MACHINE LEARNING [Electronic resource] / V. Chandra, A. Hareendran. – Delhi : PHI Learning Private Limited, 2021. – 600 p. – Access mode: https://www.google.com.ua/books/edition/MACHINE_LEARNING/b_wdEAAAQBAJ?hl=uk&gbpv=0.

12. What is Decision Tree? [A Step-by-Step Guide] | Analytics Vidhya [Electronic resurce]. – Access mode: https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/.

13. Decision Tree in Machine Learning – GeeksforGeeks [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/decision-tree-introduction-example/.

14. What is Correlation Analysis? – GeeksforGeeks [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/what-is-correlation-analysis/.

15. Correlation in machine learning – All you need to know [Electronic resource]. – Access mode: https://medium.com/@abdallahashraf90x/all-you-need-to-know-about-correlation-for-machine-learning-e249fec292e9.

16. Bhasin H. Machine Learning for Beginners – 2nd Edition [Electronic resource] / H. Bhasin. – London : BPB Publications, 2023. – 384 p. – Access mode: https://www.google.com.ua/books/edition/Machine_Learning_for_Beginners/0xPdEAAAQBAJ?hl=uk&gbpv=0.

17. Олійник, А. О. Методичні вказівки до лабораторних робіт з дисципліни «Інтелектуальний аналіз даних» для студентів спеціальності 121 «Інженерія програмного забезпечення» та спеціальності 122 «Комп'ютерні науки» / А. О. Олійник, С. О. Субботін, Є. М. Федорченко, Т. В. Федорончак, М. О. Андреєв, В. В. Козлов – Запоріжжя: НУ«Запорізька політехніка», 2021. – 105 с.

18. Rhys H. Machine Learning with R, the Tidyverse, and MLR [Electronic resource] / H. Rhys. – New York : Manning, 2020. – 536 p. – Access mode: https://www.google.com.ua/books/edition/Machine_Learning_with_R_the_tidyverse_an/qjszEAAAQBAJ?hl=uk&gbpv=0.

19. Brownlee J. Imbalanced Classification with Python [Electronic resource] / J. Brownlee. – Machine Learning Mastery, 2020. – 463 p. – Access mode: https://www.google.com.ua/books/edition/Imbalanced_Classification_with_Python/jaXJDwAAQBAJ?hl=uk&gbpv=0.

20. Brownlee J. Ensemble Learning Algorithms With Python [Electronic resource] / J. Brownlee. – Machine Learning Mastery, 2021. – 450 p. – Access mode: https://www.google.com.ua/books/edition/Ensemble_Learning_Algorithms_With_Python/IUkrEAAAQBAJ?hl=uk&gbpv=0.

21. Information Resources Management Association. Research Anthology on Machine Learning Techniques, Methods, and Applications [Electronic resource] / Information Resources Management Association. – Hershley : IGI Global, 2022. – 1516 p. – Access mode: https://www.google.com.ua/books/edition/Research_Anthology_on_Machine_Learning_T/fPZ7EAAAQBAJ?hl=uk&gbpv=0.

22. What is the k-nearest neighbors algorithm? |IBM [Electronic resource]. – Access mode: https://www.ibm.com/topics/knn.

23. Montgomery D. C. Introduction to Linear Regression Analysis [Electronic resource] / D. C. Montgomery, E. A. Peck, G. Vining. – New Jersey : Wiley, 2015. – 672 p. – Access mode:

https://www.google.com.ua/books/edition/Introduction_to_Linear_Regression_Analys/27kOCgAAQBAJ?hl=uk&gbpv=0.

24. Multinomial Logistic Regression | IBM [Electronic resource]. – Access mode: https://www.ibm.com/docs/en/spss-statistics/29.0.0?topic=regression-multinomial-logistic.

25. Biehl M. API Architecture [Text] / M. Biehl. – CreateSpace Independent Publishing Platform, 2015. – 190 p. – Access mode: https://www.google.com.ua/books/edition/API_Architecture/6D64DwAAQBAJ?hl=uk&gbpv=0.

26. Mastering API Calls: What they Are & Ensuring Their Security | Noname Security [Electronic resource]. – Access mode: https://nonamesecurity.com/learn/what-is-an-api-call/.

27. Installing RapidMiner Studio – Altair RapidMiner Documentation [Electronic resource]. – Access mode: https://docs.rapidminer.com/10.2/studio/installation/index.html.

28. Crime trend [Electronic resource]. – Access mode: https://cde.ucr.cjis.gov/LATEST/webapp/#/pages/explorer/crime/crime-trend.

29. Accurint Crime Analysis | LexisNexis Risk Soutions [Electronic resource]. – Access mode: https://risk.lexisnexis.com/products/accurint-crime-analysis.

30. Raschka S. Python Machine Learning [Electronic resource] / S. Raschka, V. Mirjalili. – Birmingham : Packt Publishing, 2019. – 772 p. – Access mode: https://www.google.com.ua/books/edition/Python_Machine_Learning/sKXIDwAAQBAJ?hl=uk&gbpv=0.

31. Python — Вікіпедія [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Python#%D0%9E%D0%B1'%D1%94%D0%BA%D1%82%D0%BD%D0%BE%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0

%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F.

32. PyCharm Features – JetBrains Python IDE [Electronic resource]. – Access mode: https://www.jetbrains.com/pycharm/features/.

33. Galea A. Applied Deep Learning with Python [Electronic resource] / A. Galea, L. Capelo. – Birmingham, Packt Publishing, 2018. – 334 p. – Access mode: https://www.google.com.ua/books/edition/Applied_Deep_Learning_with_Python/dPFsDwAAQBAJ?hl=uk&gbpv=0.

34. Hackeling G. Mastering Machine Learning with Scikit-learn [Electronic resource] / G. Hackeling. – Birmingham, Packt Publishing, 2017. – 254 p. – Access mode: https://www.google.com.ua/books/edition/Mastering_Machine_Learning_with_scikit_l/9-ZDDwAAQBAJ?hl=uk&gbpv=0.

35. Durmus M. A Hands-On Introduction to Essential Python Libraries and Frameworks (With Code Samples) [Electronic resource] / M. Durmus. – Murat Dermus, 2023. – Access mode: https://www.google.com.ua/books/edition/A_Hands_On_Introduction_to_Essential_Pyt/x4q0EAAAQBAJ?hl=uk&gbpv=0.

ДОДАТОК А

Технічне завдання

А.1 Призначення та область застосування програми

Програма, що буде розроблена, має на меті аналіз і прогнозування злочинності з використанням методів машинного навчання.

Основна область застосування програми це сфера криміналістики.

А.2 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Аналіз і прогнозування злочинності на основі методів машинного навчання», затверджене наказом Національного університету «Запорізька політехніка» № 168 від 24 квітня 2024 р.

А.3 Призначення розробки

Призначенням розробки є створення ПЗ, яке надає можливість аналізу і прогнозування злочинності на основі відкритих даних з поліцейських департаментів.

А.4 Вимоги до функціональних характеристик

ПЗ має реалізовувати наступні характеристики:

– завантаження даних: отримання даних з відкритого ресурсу за допомогою API запитів;

– попередня обробка даних: зміна типів даних для необхідних атрибутів, розбиття оригінального атрибуту на кілька менших.

– відбір атрибутів: визначення та відбір релевантних ознак,

– побудова моделей алгоритмів класифікації та регресії;

– аналіз і порівняння результатів;

– візуалізація даних: представлення даних у зручній для користувача формі.

А.5 Вимоги до надійності

Вимоги до надійності ПЗ включають в себе:

– перевірку алгоритмів, що використовуються у програмі, на точність та стабільність результатів;

– виведення повідомлень для користувача у разі виникнення помилок;

– виведення повідомлень для користувача, які показують етапи роботи програми;

– виведення повідомлень для користувача, які показують результати роботи програми.

А.6 Умови експлуатації

Для експлуатації ПЗ необхідно мати стабільне підключення до мережі Інтернет для здійснення API запитів, встановлену необхідну версію Python, систему керування пакетами pip(якщо встановлено версію Python меншу за 3.4 і вище, у які pip за замовченням входить), необхідні бібліотеки для встановлення, які можна буде завантажити за допомогою pip та файлу requirements.txt.

А.7 Вимоги до складу та параметрів технічних засобів

Комп’ютер користувача для коректної роботи з ПЗ має відповідати таким вимогам:

– стабільне підключення до мережі Інтернет для здійснення API запитів;

– операційна система: Windows 10;

– версія Python(>=3.12.1) та pip(>=23.3.2);

– вільний дисковий простір для встановлення необхідних бібліотек і роботи програми без збоїв: мінімум 1 ГБ;

– оперативна пам’ять: мінімум 8 ГБ;

– процесор: не менше 4 ядер з тактовою частотою не менше 3,4 ГГц.

А.8 Необхідні стадії і терміни розробки

Розробка ПЗ для аналізу та прогнозування злочинності включає наступні стадії та терміни:

– аналіз вимог: визначення вимог до програми. Термін: 1 тиждень;

– проєктування системи: розробка архітектури та вибір технологічних засобів. Термін: 1 тиждень;

– реалізація та програмування: написання коду програми та реалізація визначених функцій. Термін: 2 тижні;

– тестування та налагодження: проведення випробувань для виявлення та усунення помилок. Термін: 1 тиждень.

А.9 Види випробувань

Для забезпечення якості та надійності проведуться наступні види випробувань:

– функціональні тести: перевірка відповідності функціональних вимог програми;

– інтеграційні тести: перевірка взаємодії між різними частинами програми;

– системні тести: тестування програми в цілому.

ДОДАТОК Б

Код програми

```
import sys

import requests

import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model._cd_fast import ConvergenceWarning
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectFromModel, SelectKBest,
f_regression
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, zero_one_loss,
precision_score, \
    recall_score, f1_score
from sklearn.preprocessing import LabelEncoder

from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline

import warnings

warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=ConvergenceWarning)
```

```
def emb_methods(dataframe):
    X = dataframe.drop(columns=['crm_cd_desc', 'crm_cd'])
    y = dataframe['crm_cd_desc']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)
    X_train.columns = [x.capitalize() for x in X_train.columns]
    X_test.columns = [x.capitalize() for x in X_test.columns]
    X_train, X_test = X_train.drop(columns=['Dr_no', 'Rpt_dist_no', 'Status',
'Crm_cd_1', 'Crm_cd_2',
                                            'Crm_cd_3']), X_test.drop(columns=['Dr_no',
                                            'Rpt_dist_no', 'Status', 'Crm_cd_1', 'Crm_cd_2',
'Crm_cd_3'])
    X_train_dt = X_train[:2000]
    X_test_dt = X_test[:2000]
    y_train_dt = y_train[:2000]
    y_test_dt = y_test[:2000]

    dt_fs = SelectFromModel(DecisionTreeClassifier(random_state=5),
threshold='median')
    dt_fs.fit(X_train_dt, y_train_dt)
    dt_selected = list(X_train_dt.columns[(dt_fs.get_support())])
    dt_importances = pd.Series(dt_fs.estimator_.feature_importances_)

    rf_fs = SelectFromModel(RandomForestClassifier(n_estimators=50,
random_state=5), threshold='median')
    rf_fs.fit(X_train, y_train)
    rf_selected = list(X_train.columns[(rf_fs.get_support())])
    rf_importances = pd.Series(rf_fs.estimator_.feature_importances_)
```

```
    print(' Features selected by Decision Tree:', dt_selected)
    print(' Features selected by Random Forest:', rf_selected)

    for_viz = pd.DataFrame()
    for_viz['Features'] = X_train.columns
    for_viz['Dec_tree'] = dt_importances
    for_viz['Ran_forest'] = rf_importances

    plt.figure(figsize=(13, 9))
    bar_width = 0.35
    index = np.arange(len(for_viz['Features']))
    plt.bar(index, for_viz['Dec_tree'], bar_width, label='Dec_tree',
color='#96B5DE')
    plt.bar(index + bar_width, for_viz['Ran_forest'], bar_width, label='Ran_forest',
color='#F2DA8D')
    plt.xlabel('Features', fontsize=14, labelpad=8)
    plt.ylabel('Importance', fontsize=14, labelpad=5)
    plt.title('Feature importances using Decision Tree and Random Forest',
fontsize=16, pad=14)
    plt.xticks(index + bar_width / 2, for_viz['Features'], rotation=45)
    plt.legend()
    plt.show()
    return dt_selected, rf_selected

def flt_methods(dataframe):
    X = dataframe.drop(columns=['crm_cd', 'crm_cd_desc'])
    y = dataframe['crm_cd']
    X.columns = [x.capitalize() for x in X.columns]
    X = X.drop(columns=['Dr_no', 'Rpt_dist_no', 'Status', 'Crm_cd_1', 'Crm_cd_2',
'Crm_cd_3'])
```

```
    fs = SelectKBest(score_func=f_regression, k=7)
    x_selected = fs.fit_transform(X, y)
    selected_features = fs.get_support()
    print(' Features selected by Pearson method:',
list(X.columns[selected_features]))

    corr_matrix = X.corr(method='pearson')
    f, ax1 = plt.subplots(figsize=(13, 9))
    sns.heatmap(corr_matrix, annot=True, linewidths=.5, ax=ax1)
    ax1.set_title('Correlation matrix by Pearson\'s method', fontsize=16, pad=14)
    plt.show()
    return list(X.columns[selected_features])
def feature_selection(dataframe):
    print('Feature selection:')
    flt_selected = flt_methods(dataframe)
    dt_selected, rf_selected = emb_methods(dataframe)
    features_selected = set(flt_selected) & set(dt_selected) & set(rf_selected)
    print(' Features selected by three methods:', features_selected)
    print(' Amount of features selected by three methods:', len(features_selected),
'\n')
    print(' Feature importances and the correlation coefficients are small enough to
make a decision based on them.')
    print(' But some of the following columns will be dropped for each classification
and regression method: Dr_no, Rpt_dist_no, Status, Crm_cd_2, Crm_cd_3\n')

def knn_classifier(dataframe):
    X = dataframe.drop(columns=['crm_cd', 'crm_cd_desc'])
    y = dataframe['crm_cd']
    X.columns = [x.capitalize() for x in X.columns]
```

```
    X = X.drop(columns=['Dr_no', 'Rpt_dist_no', 'Status', 'Crm_cd_2', 'Crm_cd_3'])
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=2)
    knn = KNeighborsClassifier(n_neighbors=15, weights='distance')
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)

    over_sampler = RandomOverSampler(sampling_strategy='minority')
    under_sampler = RandomUnderSampler(sampling_strategy='majority')
    model_with_sampling = Pipeline([('over_sampler', over_sampler),
('under_sampler', under_sampler), ('lr', knn)])
    model_with_sampling.fit(X_train, y_train)
    y_pred_sampling = model_with_sampling.predict(X_test)

    print('KNN:')
    df_metrics = pd.DataFrame()
    df_metrics['Metrics'] = ['Accuracy', 'Missclassification', 'Precision', 'Recall', 'F1-
score']
    df_metrics['No_sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred)),
                                 '{0:.4f}'.format(zero_one_loss(y_test, y_pred)),
                                 '{0:.4f}'.format(precision_score(y_test, y_pred,
average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(recall_score(y_test, y_pred,
average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(f1_score(y_test, y_pred, average='weighted'))]
    df_metrics['Sampling'] = ['{0:.4f}'.format(accuracy_score(y_test,
y_pred_sampling)),
                              '{0:.4f}'.format(zero_one_loss(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(precision_score(y_test, y_pred_sampling,
average='weighted', zero_division=1)),
```

```
                    '{0:.4f}'.format(recall_score(y_test, y_pred_sampling,
average='weighted', zero_division=1)),
                    '{0:.4f}'.format(f1_score(y_test, y_pred_sampling,
average='weighted'))]
    df_metrics.set_index('Metrics', inplace=True)
    print(' ', df_metrics, '\n')

    print('Counting CV scores...')
    k_folds = KFold(n_splits=10)
    scores = cross_val_score(knn, X, y, cv=k_folds)
    print(' Cross Validation Scores: ', list(scores))
    print(' Average CV Score: ', '{0:.4f}'.format(scores.mean()))
    print(' Deviation from average cv score:', '{0:.4f}'.format(accuracy_score(y_test,
y_pred) - scores.mean()), '\n')

    print('Calculating perfect k..')
    perfect_k = 0.0
    perfect_k_number = 0
    for k in range(1, 30):
        knn = KNeighborsClassifier(n_neighbors=k)
        scores = cross_val_score(knn, X_train, y_train, cv=5)
        if scores.mean() > perfect_k:
            perfect_k = scores.mean()
            perfect_k_number = k
    print(f' The perfect k is {perfect_k_number} with accuracy =',
'{0:.4f}'.format(perfect_k), '\n')
    if (float(df_metrics.loc['Accuracy', 'No_sampling']) -
float(df_metrics.loc['Accuracy', 'Sampling'])) > 0:
        return [df_metrics.loc['Accuracy', 'No_sampling'], 'No_sampling knn']
    elif (float(df_metrics.loc['Accuracy', 'No_sampling']) -
```

```
float(df_metrics.loc['Accuracy', 'Sampling'])) < 0:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'Sampling knn']
    else:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'knn']

def rf_classifier(dataframe):
    X = dataframe.drop(columns=['crm_cd', 'crm_cd_desc'])
    y = dataframe['crm_cd_desc']
    X.columns = [x.capitalize() for x in X.columns]
    X = X.drop(columns=['Dr_no'])
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=2)

    rf_model = RandomForestClassifier(n_estimators=50, random_state=5)
    rf_model.fit(X_train, y_train)
    y_pred = rf_model.predict(X_test)
    print('Random forest:')
    c1 = classification_report(y_test, y_pred, zero_division=1)

    over_sampler = RandomOverSampler(sampling_strategy='minority')
    under_sampler = RandomUnderSampler(sampling_strategy='majority')
    model_with_sampling = Pipeline([('over_sampler', over_sampler),
('under_sampler', under_sampler), ('lr', rf_model)])
    model_with_sampling.fit(X_train, y_train)
    y_pred_sampling = model_with_sampling.predict(X_test)
    c2 = classification_report(y_test, y_pred_sampling, zero_division=1)

    c1_lines = c1.split('\n')
    c1_class_names = []
    c1_class_recall = []
```

```
    for line in c1_lines[2:-4]:
        values = line.split()
        if len(values) >= 3:
            c1_class_names.append(values[:-4])
            c1_class_recall.append(float(values[-3]))

    c2_lines = c2.split('\n')
    c2_class_names = []
    c2_class_recall = []
    for line in c2_lines[2:-4]:
        values2 = line.split()
        if len(values2) >= 3:
            c2_class_names.append(values2[:-4])
            c2_class_recall.append(float(values2[-3]))

    df_metrics = pd.DataFrame()
    df_metrics['Metrics'] = ['Accuracy', 'Missclassification', 'Precision', 'Recall', 'F1-
score']
    df_metrics['No_sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred)),
                                 '{0:.4f}'.format(zero_one_loss(y_test, y_pred)),
                                 '{0:.4f}'.format(precision_score(y_test, y_pred,
average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(recall_score(y_test, y_pred,
average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(f1_score(y_test, y_pred, average='weighted'))]
    df_metrics['Sampling'] = ['{0:.4f}'.format(accuracy_score(y_test,
y_pred_sampling)),
                              '{0:.4f}'.format(zero_one_loss(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(precision_score(y_test, y_pred_sampling,
average='weighted', zero_division=1)),
```

```
                    '{0:.4f}'.format(recall_score(y_test, y_pred_sampling,
average='weighted', zero_division=1)),
                    '{0:.4f}'.format(f1_score(y_test, y_pred_sampling,
average='weighted'))]
    df_metrics.set_index('Metrics', inplace=True)
    print(' ', df_metrics, '\n')
    sum_c1, sum_c2, sum_ = 0.0, 0.0, 0.0
    for s in c1_class_recall:
        sum_c1 += s
    for s in c2_class_recall:
        sum_c2 += s
    sum_ = sum_c1 - sum_c2
    print('No_sampling dataset recall deviation from sampling_dataset:', sum_)
    if sum_ > 0:
        print('Recall in the no_sampling dataset is higher than in the
sampling_dataset.\n')
    else:
        print('Recall in the no_sampling dataset is lower than in the
sampling_dataset.\n')

    print('Counting CV scores...')
    k_folds = KFold(n_splits=10)
    scores = cross_val_score(rf_model, X, y, cv=k_folds)
    print(' Cross Validation Scores: ', list(scores))
    print(' Average CV Score: ', '{0:.4f}'.format(scores.mean()))
    print(' Deviation from average cv score:', '{0:.4f}'.format(accuracy_score(y_test,
y_pred) - scores.mean()), '\n')
    if (float(df_metrics.loc['Accuracy', 'No_sampling']) -
float(df_metrics.loc['Accuracy', 'Sampling'])) > 0:
        return [df_metrics.loc['Accuracy', 'No_sampling'], 'No_sampling rf']
```

```
    elif (float(df_metrics.loc['Accuracy', 'No_sampling']) -
float(df_metrics.loc['Accuracy', 'Sampling'])) < 0:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'Sampling rf']
    else:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'rf']

def classification(dataframe):
    print('Classification:')
    rf_lst = rf_classifier(dataframe)
    knn_lst = knn_classifier(dataframe)
    return rf_lst, knn_lst

def multinomial_regr(dataframe):
    print('Multinomial regression:')
    X = dataframe.drop(columns=['crm_cd', 'crm_cd_desc', 'dr_no', 'rpt_dist_no',
'status', 'crm_cd_2', 'crm_cd_3'])
    y = dataframe['crm_cd']
    X.columns = [x.capitalize() for x in X.columns]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

    model = LogisticRegression(multi_class='multinomial', solver='saga',
max_iter=3000, penalty='l2',
                               C=0.5, tol=0.001, random_state=42)
    model2 = LogisticRegression(multi_class='multinomial', solver='saga',
tol=0.001)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    test_score = model.score(X_test, y_test)
```

```
    over_sampler = RandomOverSampler(sampling_strategy='minority')
    under_sampler = RandomUnderSampler(sampling_strategy='majority')
    model_with_sampling = Pipeline([('over_sampler', over_sampler),
('under_sampler', under_sampler), ('lr', model2)])
    model_with_sampling.fit(X_train, y_train)
    y_pred_sampling = model_with_sampling.predict(X_test)

    df_metrics = pd.DataFrame()
    df_metrics['Metrics'] = ['Accuracy', 'Missclassification', 'Precision', 'Recall', 'F1-
score']
    df_metrics['No_sampling'] = ['{0:.4f}'.format(accuracy_score(y_test, y_pred)),
                                 '{0:.4f}'.format(zero_one_loss(y_test, y_pred)),
                                 '{0:.4f}'.format(precision_score(y_test, y_pred,
average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(recall_score(y_test, y_pred,
average='weighted', zero_division=1)),
                                 '{0:.4f}'.format(f1_score(y_test, y_pred, average='weighted'))]
    df_metrics['Sampling'] = ['{0:.4f}'.format(accuracy_score(y_test,
y_pred_sampling)),
                              '{0:.4f}'.format(zero_one_loss(y_test, y_pred_sampling)),
                              '{0:.4f}'.format(precision_score(y_test, y_pred_sampling,
average='weighted', zero_division=1)),
                              '{0:.4f}'.format(recall_score(y_test, y_pred_sampling,
average='weighted', zero_division=1)),
                              '{0:.4f}'.format(f1_score(y_test, y_pred_sampling,
average='weighted'))]
    df_metrics.set_index('Metrics', inplace=True)
    print(' ', df_metrics, '\n')
    if float(df_metrics.loc['Recall', 'No_sampling']) - float(df_metrics.loc['Recall',
'Sampling']):
```

```
        print('Recall in the no_sampling dataset is higher than in the
sampling_dataset.\n')
    else:
        print('Recall in the no_sampling dataset is lower than in the
sampling_dataset.\n')

    print('Counting CV scores...')
    scores = cross_val_score(model, X_train, y_train, cv=3)
    print(f' Scores for each fold: {scores}')
    print(f' Mean score:', '{0:.4f}'.format(scores.mean()))
    print(' Deviation from average cv score:', '{0:.4f}'.format(test_score -
scores.mean()), '\n')
    if (float(df_metrics.loc['Accuracy', 'No_sampling']) -
float(df_metrics.loc['Accuracy', 'Sampling'])) > 0:
        return [df_metrics.loc['Accuracy', 'No_sampling'], 'No_sampling mlt']
    elif (float(df_metrics.loc['Accuracy', 'No_sampling']) -
float(df_metrics.loc['Accuracy', 'Sampling'])) < 0:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'Sampling mlt']
    else:
        return [df_metrics.loc['Accuracy', 'Sampling'], 'mlt']

def regression(dataframe):
    mlt_lst = multinomial_regr(dataframe)
    return mlt_lst
def data_preprocess(dataframe):
    pd.set_option('future.no_silent_downcasting', True)
    pd.set_option('display.max_columns', 28)
    pd.set_option('display.max_rows', 28)
    dataframe['date_occ'] = pd.to_datetime(dataframe['date_occ'])
    dataframe['Year'] = dataframe['date_occ'].dt.year
```

```
    dataframe['Month'] = dataframe['date_occ'].dt.month
    dataframe['Day'] = dataframe['date_occ'].dt.day
    dataframe['Weekday'] = dataframe['date_occ'].apply(lambda x: x.weekday())
    dataframe['Time'] = dataframe['time_occ'].astype(int)
    dataframe['Time'] = dataframe['Time'].apply(lambda x: x / 100).astype(int)

    dataframe.drop(columns=['date_occ', 'time_occ', 'date_rptd', 'part_1_2',
'cross_street'], inplace=True)
    print('\nThe unnecessary columns were removed.')
    [dataframe.pop(x) for x in ['area_name', 'mocodes', 'premis_desc',
'weapon_desc', 'status_desc', 'location']]
    dataframe.fillna({'weapon_used_cd': 0, 'crm_cd_2': 0, 'crm_cd_3': 0,
'premis_cd': 0,
                      'vict_sex': 0, 'vict_descent': 0}, inplace=True)
    print('Missing values were filled with 0.')
    dataframe['vict_sex'] = dataframe['vict_sex'].replace({'H': 0, 'X': 0})
    dataframe['vict_descent'] = dataframe['vict_descent'].replace({'X': 0, '-': 0})

    cols_to_conv = ['dr_no', 'area', 'rpt_dist_no', 'crm_cd', 'vict_age', 'premis_cd',
                    'weapon_used_cd', 'crm_cd_1', 'crm_cd_2', 'crm_cd_3']
    dataframe[cols_to_conv] = dataframe[cols_to_conv].astype(int)
    dataframe['lat'] = dataframe['lat'].astype(float)
    dataframe['lon'] = dataframe['lon'].astype(float)
    dataframe['crm_cd_desc'] = dataframe['crm_cd_desc'].astype(str)
    print('Features were converted to int and float.')

    categorical_features = ['vict_sex', 'vict_descent', 'status']
    dataframe[categorical_features] = dataframe[categorical_features].astype(str)
    inv_values = []
    values = []
```

```
    dict_lst = {}
    encoded = []
    label_encoder = LabelEncoder()
    for feature in categorical_features:
        inv_values.append(dataframe[feature].value_counts().keys().to_list())
        dataframe[feature] = label_encoder.fit_transform(dataframe[feature])
        values.append(dataframe[feature].value_counts().keys().to_list())
    for i, inv_value_lst in enumerate(inv_values):
        for j, inv_value in enumerate(inv_value_lst):
            dict_lst.update({values[i][j]: inv_value})
        encoded.append(dict_lst.copy())
        dict_lst.clear()
    print('Labels were encoded.')

    for i in range(len(encoded)):
        print(f" Feature {categorical_features[i]}: {encoded[i]}")
    return dataframe
def data_load():
    my_limit = 1000
    my_offset = 47150
    temp_dfs = list()
    i = 0
    while i < 20:
        url = 'https://data.lacity.org/resource/2nrs-mtv8.json?$limit={}&$offset={}'
        endpoint = url.format(my_limit, my_offset)
        response = requests.get(endpoint)
        if response.status_code == 200:
            data = response.json()
            temp_dfs.append(pd.DataFrame(data))
        else:
```

```
            print('Error has occurred. Please try again')
            return 0
        i += 1
        my_offset += 47150
    big_df = pd.concat(temp_dfs)
    return big_df

def accuracy_comparison(rf_lst, knn_lst, mlt_lst):
    lst_x, lst_y = [], []
    lst_x.append(rf_lst[1])
    lst_x.append(knn_lst[1])
    lst_x.append(mlt_lst[1])
    lst_y.append(float(rf_lst[0]))
    lst_y.append(float(knn_lst[0]))
    lst_y.append(float(mlt_lst[0]))
    sns.set_theme(style='whitegrid', context='notebook')
    plt.figure(figsize=(13, 9))
    sns.barplot(x=np.array(lst_x), y=np.array(lst_y), color='#96B5DE')
    plt.xlabel('The highest accuracy of each method', fontsize=14, labelpad=8)
    plt.ylabel('Accuracy', fontsize=14, labelpad=5)
    plt.title('Accuracy comparison ', fontsize=16, pad=14)
    plt.show()

def patterns(df):
    print('Some interesting patterns:')
    sns.set_theme(style='whitegrid', context='notebook')
    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(13, 9))
    plt.subplots_adjust(hspace=0.4, wspace=0.3)
    sns.barplot(x=np.array(df['Year'].value_counts().keys().to_list()),
                y=np.array(df['Year'].value_counts().values), ax=ax1,
```

```
            color='#96B5DE')
    ax1.set_xlabel('Year', fontsize=14, labelpad=8)
    ax1.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax1.set_title('Crime count by year', fontsize=16, pad=14)

    sns.barplot(x=np.array(df['Month'].value_counts().keys().to_list()),
                y=np.array(df['Month'].value_counts().values), ax=ax2,
                color='#F2DA8D')
    ax2.set_xlabel('Month', fontsize=14, labelpad=8)
    ax2.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax2.set_title('Crime count by month', fontsize=16, pad=14)

    sns.lineplot(x=np.array(df['Day'].value_counts().keys().to_list()),
                 y=np.array(df['Day'].value_counts().values), ax=ax4,
                 color='#7CDA97')
    ax4.set_xlabel('Day', fontsize=14, labelpad=8)
    ax4.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax4.set_title('Crime count by day', fontsize=16, pad=14)

    sns.lineplot(x=np.array(df['Time'].value_counts().keys().to_list()),
                 y=np.array(df['Time'].value_counts().values), ax=ax3,
                 color='#EF6A61')
    ax3.set_xlabel('Time', fontsize=14, labelpad=8)
    ax3.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax3.set_title('Crime count by time', fontsize=16, pad=14)
    plt.show()

    tmp_cd_desc = ['Stol_vehicle', 'Battery(simple_assault)', 'Burglary',
'Identity_theft', 'Burg_from_vehicle',
                   'Vandalism-felony', 'Assault_deadly_weapon', 'Plain_theft',
```

```
'Intimate_partner_assault',
                'Theft_from_motor_vehicle']
    plt.figure(figsize=(13, 9))
    sns.barplot(x=tmp_cd_desc,
                y=np.array(df['crm_cd_desc'].value_counts().values[:10]),
                color='#96B5DE')
    plt.xlabel('Crime category', fontsize=14, labelpad=8)
    plt.ylabel('Crime count', fontsize=14, labelpad=5)
    plt.title('Crime counts by top 10 crime categories', fontsize=16, pad=14)
    plt.xticks(rotation=25)
    plt.show()

    f, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(13, 9))
    plt.subplots_adjust(hspace=0.4, wspace=0.3)
    sns.barplot(x=np.array(['Mon', 'Tue', 'Wed', 'Th', 'Fri', 'Sat', 'Sun']),
                y=np.array(df['Weekday'].value_counts().values), ax=ax1,
                color='#96B5DE')
    ax1.set_xlabel('Weekday', fontsize=14, labelpad=8)
    ax1.set_ylabel('Crime count', fontsize=14, labelpad=8)
    ax1.set_title('Crime counts by weekdays', fontsize=16, pad=14)

    tmp_vict_age = []
    for age in df['vict_age'].value_counts().keys().to_list()[:10]:
        if age == 0:
            tmp_vict_age.append('Unknown')
        else:
            tmp_vict_age.append(age)

    sns.barplot(x=tmp_vict_age,
                y=np.array(df['vict_age'].value_counts().values[:10]), ax=ax2,
```

```
            color='#F2DA8D')
    ax2.set_xlabel('Victim age', fontsize=14, labelpad=8)
    ax2.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax2.set_title('Crime counts by top 10 victim ages', fontsize=16, pad=14)

    sns.barplot(x=['Male', 'Female', 'Unknown'],
            y=np.array(df['vict_sex'].value_counts().values), ax=ax3,
            color='#EF6A61')
    ax3.set_xlabel('Victim Sex', fontsize=14, labelpad=8)
    ax3.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax3.set_title('Crime counts by victim sex', fontsize=16, pad=14)

    tmp_vict_descent = ['Hispanic', 'Unknown', 'White', 'Black', 'Other', 'Other
Asian',
                    'Korean', 'Filipino', 'Chinese', 'Japanese']
    sns.barplot(x=tmp_vict_descent,
            y=np.array(df['vict_descent'].value_counts().values[:10]), ax=ax4,
            color='#7CDA97')
    ax4.set_xlabel('Victim Descent', fontsize=14, labelpad=8)
    ax4.set_ylabel('Crime count', fontsize=14, labelpad=5)
    ax4.set_title('Crime counts by top 10 victim descents', fontsize=16, pad=14)
    plt.xticks(rotation=25)
    plt.show()


if __name__ == '__main__':
    print('Starting the program...')
    df = data_load()
    if isinstance(df, int):
        sys.exit('Failed to load dataset.')
```

```
elif isinstance(df, pd.DataFrame):
    print('Dataset was loaded successfully.')
    df = data_preprocess(df)
    if isinstance(df, pd.DataFrame):
        print('Dataset was preprocessed successfully.\n')
    else:
        sys.exit('Failed to preprocess dataset.\n')
feature_selection(df)
rf_lst, knn_lst = classification(df)
mlt_lst = regression(df)
accuracy_comparison(rf_lst, knn_lst, mlt_lst)
patterns(df)
print('The program is executed.')
```

ДОДАТОК В

Слайди презентації

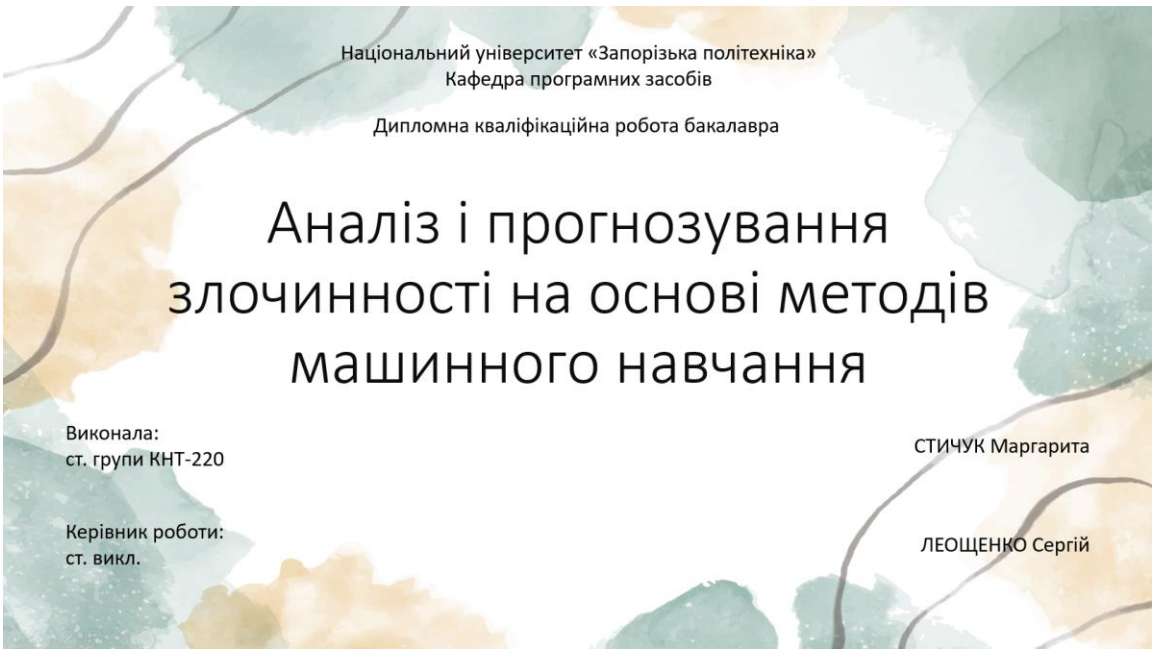


Рисунок В.1 – Слайд №1

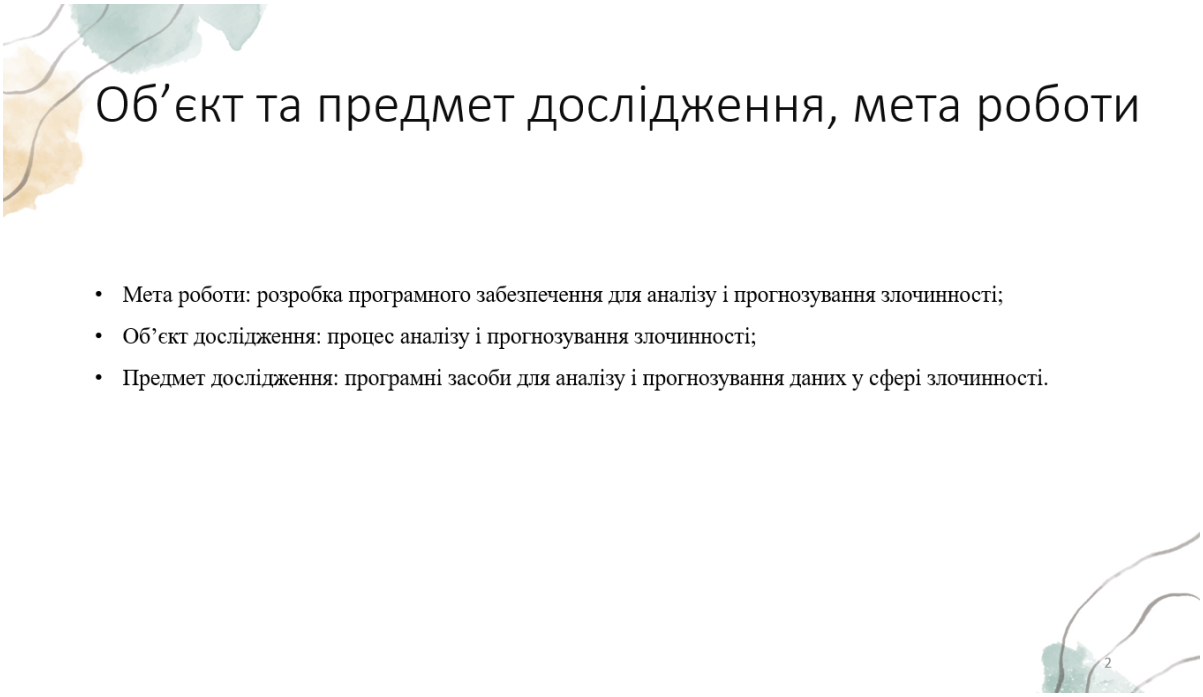


Рисунок В.2 – Слайд №2

Порівняння аналогів

Критерій порівняння	RapidMiner Studio	Crime Data Explorer	Accurint Crime Analysis	Мій застосунок
Безкоштовність	+	+	-	+
Простота встановлення	+	+	+	+
Необхідність самостійного пошуку даних	+	-	-	-
Детальний аналіз	+	-	+	+
Наявність прогнозування	+	-	+	+
Наявність візуалізації	+	+	+	+

3

Рисунок В.3 – Слайд №3

Вибір мови програмування

Критерій	Python	Java	R
Швидкість	+	++	+
Бібліотеки машинного навчання	++	++	++
Розв'язання математичних задач	++	+	++
Потенціал розширення	++	++	+
Середовища розробки	++	+	+

4

Рисунок В.4 – Слайд №4

Вибір середовища розробки

Критерій	PyCharm	VS	Spyder
Вимоги до ресурсів	Великі(±)	Великі(±)	Низькі(++)
Спеціалізація для Python	++	+	++
Розширені можливості	++	++	±
Інтеграція з VCS	++	++	±
Підтримка розширень	++	++	±
Інтеграція з бібліотеками машинного навчання	++	+	++

5

Рисунок В.5 – Слайд №5

Машинне навчання

Штучний інтелект (ШІ) – це підхід до вивчення розумної поведінки, який заснований на припущенні, що інтелект можна відтворити. Однією з цілей є вивчення та розуміння того, як працює людський інтелект, іншою – його імітація за допомогою комп'ютера.

Машинне навчання – це галузь ШІ та комп'ютерних наук, яка фокусується на використанні даних та алгоритмів, що дозволяють ШІ імітувати спосіб навчання людини, поступово підвищуючи його точність.

Рисунок В.6 – Слайд №6

Вимоги до технічних засобів

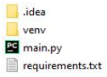
- стабільне підключення до мережі Інтернет;
- операційна система: Windows 10;
- процесор: не менше 4 ядер з тактовою частотою не менше 3,4 ГГц;
- оперативна пам’ять: мінімум 8 ГБ;
- вільний дисковий простір – 1 ГБ;
- версія Python(>=3.12.1) та pip(>=23.3.2).

7

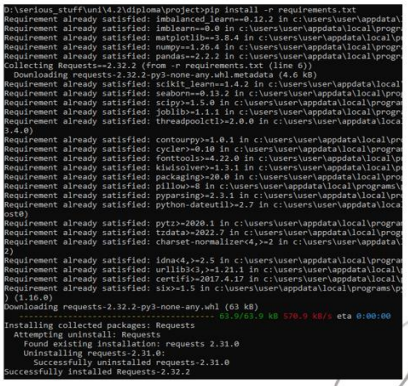
Рисунок В.7 – Слайд №7

Встановлення необхідних бібліотек

Після завантаження проєкту потрібно спочатку встановити необхідні бібліотеки. Це можна зробити за допомогою pip та файлу requirements.txt так, як показано на рисунках.




```
imbalanced_learn==0.12.2
imblearn==0.0
matplotlib==3.8.4
numpy==1.26.4
pandas==2.2.2
Requests==2.32.2
scikit_learn==1.4.2
seaborn==0.13.2
```



8

Рисунок В.8 – Слайд №8

Запуск програми

Для запуску програми потрібно, знаходячись у папці проєкту, прописати python main.py.

```
D:\serious_stuff\uni\4.2\diploma\project>python main.py
Starting the program...
Dataset was loaded successfully.
```

Рисунок 4 – Запуск програми

```
The unnecessary columns were removed.
Missing values were filled with 0.
Features were converted to int and float.
Labels were encoded.
 Feature vict_sex: {2: 'M', 1: 'F', 0: '0'}
 Feature vict_descent: {7: 'H', 0: '0', 15: 'W',
 Feature status: {3: 'IC', 1: 'AO', 0: 'AA', 4:
Dataset was preprocessed successfully.
```

Рисунок 5 – Попередня обробка даних

```
Feature selection:
 Features selected by Pearson method: ['Vict_age', 'Vict_sex', 'Vict_descent', 'Premis_cd', 'Weapon_used_cd', 'Lat', 'Lon']
 Features selected by Decision Tree: ['Vict_age', 'Premis_cd', 'Weapon_used_cd', 'Lat', 'Lon', 'Day', 'Time']
 Features selected by Random Forest: ['Vict_age', 'Premis_cd', 'Weapon_used_cd', 'Lat', 'Lon', 'Day', 'Time']
 Features selected by three methods: {'Vict_age', 'Premis_cd', 'Lon', 'Weapon_used_cd', 'Lat'}
 Amount of features selected by three methods: 5

 Feature importances and the correlation coefficients are small enough to make a decision based on them.
 But some of the following columns will be dropped for each classification and regression method: Dr_no, Rpt_dist_no, Status, Crm_cd_2, Crm_cd_3
```

Рисунок 6 – Відбір атрибутів

9

Рисунок В.9 – Слайд №9

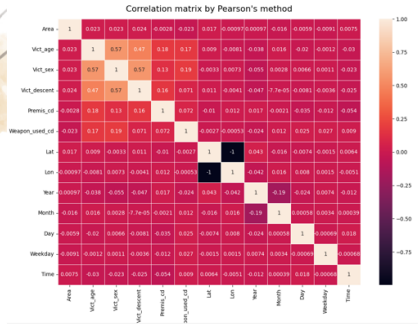


Рисунок 7 – Матриця кореляції Пірсона

На рисунках представлено матрицю кореляцій між атрибутами методом Пірсона, а також графік, на якому порівнюються важливості ознак, вирахувані за допомогою методів Random Forest та Decision Tree.

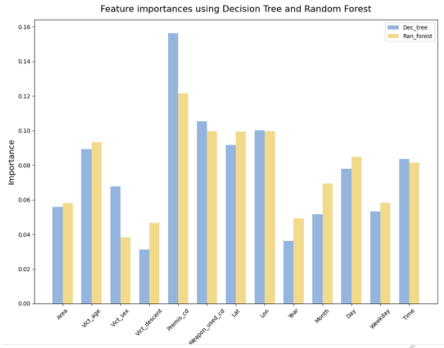


Рисунок 8 – Графік важливостей ознак

10

Рисунок В.10 – Слайд №10

Випадковий ліс

На рисунку представлено результати класифікації методом випадкового лісу.

```
Classification:
Random forest:
                    No_sampling Sampling
Metrics
Accuracy                 0.9127   0.9157
Missclassification       0.0873   0.0843
Precision                0.9125   0.9149
Recall                   0.9127   0.9157
F1-score                 0.8989   0.9017

No_sampling dataset recall deviation from sampling_dataset: -0.15000000000000568
Recall in the no_sampling dataset is lower than in the sampling_dataset.

Counting CV scores...
 Cross Validation Scores:  [0.9185, 0.913, 0.9265, 0.9135, 0.927, 0.925, 0.92, 0.9335, 0.9275, 0.9415]
 Average CV Score:  0.9246
 Deviation from average cv score: -0.0119
```

Рисунок 9 – Метод випадкового лісу

11

Рисунок В.11 – Слайд №11

К-найближчих сусідів

На рисунку представлено результати класифікації методом k-найближчих сусідів.

```
KNN:
                    No_sampling Sampling
Metrics
Accuracy                 0.7752   0.7752
Missclassification       0.2248   0.2248
Precision                0.7795   0.7795
Recall                   0.7752   0.7752
F1-score                 0.7548   0.7548

Counting CV scores...
 Cross Validation Scores:  [0.7655, 0.766, 0.7775, 0.77, 0.805, 0.7975, 0.7785, 0.796, 0.7875, 0.796]
 Average CV Score:  0.7840
 Deviation from average cv score: -0.0088

Calculating perfect k..
 The perfect k is 4 with accuracy = 0.7762
```

Рисунок 10 – Метод k-найближчих сусідів

12

Рисунок В.12 – Слайд №12

Мультиноміальна логістична регресія

На рисунку представлено результати класифікації методом мультиноміальної логістичної регресії.

```
Multinomial regression:
                    No_sampling Sampling
Metrics
Accuracy                 0.6538   0.5887
Missclassification       0.3462   0.4113
Precision                0.6429   0.5863
Recall                   0.6538   0.5887
F1-score                 0.5955   0.5154

Recall in the no_sampling dataset is higher than in the sampling_dataset.

Counting CV scores...
 Scores for each fold: [0.62116992 0.62695522 0.62516074]
 Mean score: 0.6244
 Deviation from average cv score: 0.0294
```

Рисунок 11 – Метод мультиноміальної логістичної регресії

13

Рисунок В.13 – Слайд №13

Порівняння методів

На рисунку представлено порівняння часток успішності використаних методів.

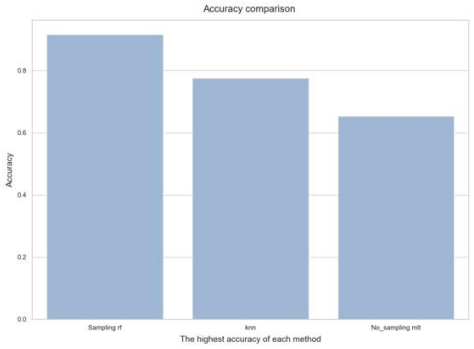


Рисунок 12 – Порівняння методів

14

Рисунок В.14 – Слайд №14

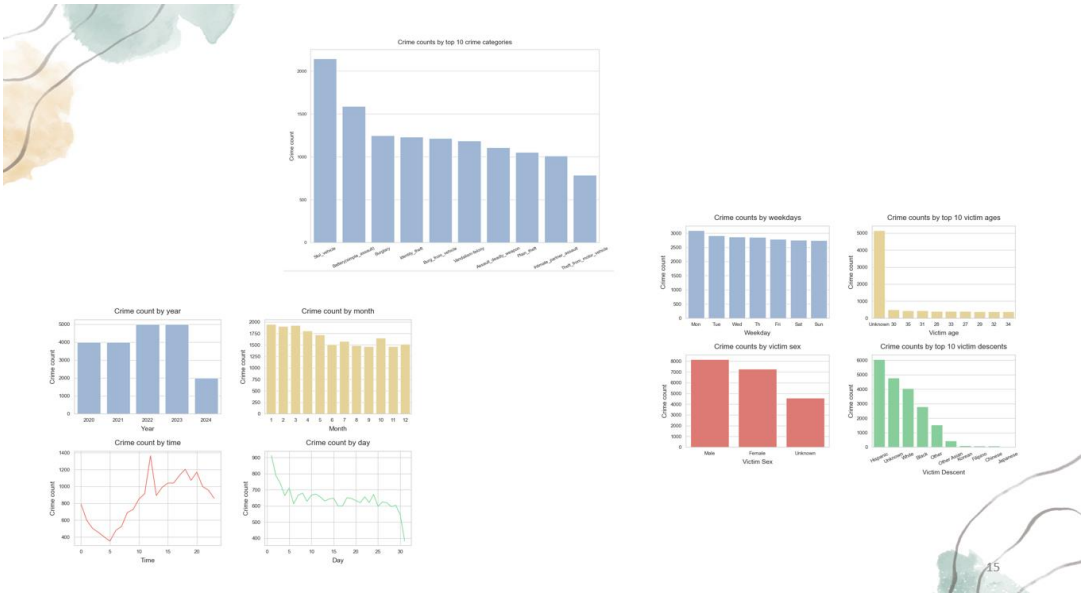
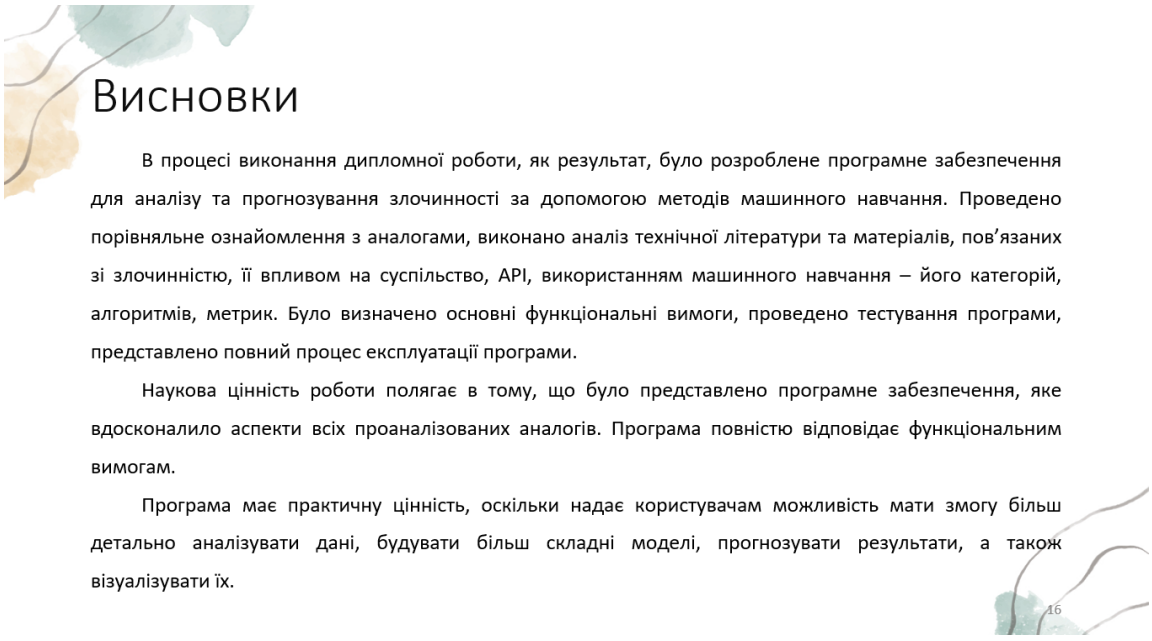


Рисунок В.15 – Слайд №15



Висновки

В процесі виконання дипломної роботи, як результат, було розроблене програмне забезпечення для аналізу та прогнозування злочинності за допомогою методів машинного навчання. Проведено порівняльне ознайомлення з аналогами, виконано аналіз технічної літератури та матеріалів, пов'язаних зі злочинністю, її впливом на суспільство, API, використанням машинного навчання – його категорій, алгоритмів, метрик. Було визначено основні функціональні вимоги, проведено тестування програми, представлено повний процес експлуатації програми.

Наукова цінність роботи полягає в тому, що було представлено програмне забезпечення, яке вдосконалило аспекти всіх проаналізованих аналогів. Програма повністю відповідає функціональним вимогам.

Програма має практичну цінність, оскільки надає користувачам можливість мати змогу більш детально аналізувати дані, будувати більш складні моделі, прогнозувати результати, а також візуалізувати їх.

Рисунок В.16 – Слайд №16