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)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ПОБУДОВИ МОДЕЛЕЙ ОПТИМІЗАЦІЇ РОБОТИ КАМЕРИ ЗГОРЯННЯ АВІАДВИГУНА
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR DEVELOPING OPTIMISATION MODELS OF AN AIRCRAFT ENGINE COMBUSTION CHAMBER

Виконав(ла): студент 2 курсу, групи КНТ-213м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Системи штучного інтелекту

ЧЕРНЕЦОВ В. М.
(ПРІЗВИЩЕ та ініціали)
Керівник СУББОТІН С.О.
(ПРІЗВИЩЕ та ініціали)
Рецензент ІЛЬЯШЕНКО М.Б.
(ПРІЗВИЩЕ та ініціали)

2024

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(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)

ЧЕРНЕЦОВА Володимира Михайловича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів побудови моделей оптимизації роботи камери згоряння авіадвигуна. Research and Software Implementation of Methods for Developing Optimisation Models of an Aircraft Engine Combustion Chamber

керівник проєкту (роботи) д.т.н., проф., СУББОТІН Сергій Олександрович
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від "07" жовтня 2024 року № 398
2. Строк подання студентом проєкту (роботи) 02 грудня 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. Розробка програми. 3. Експериментальне дослідження.

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
1-3 Основна частина	СУББОТІН С.О., зав.каф.		
Нормоконтроль	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання “ 09 ” вересня 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.	3-4 тижні	Розділ 1
3	Розробка архітектурних підходів, моделей та	5-6 тижні	Розділ 2
	алгоритмів.		
4	Розробка програми.	7-8 тижні	Розділи 3
5	Експериментальне дослідження програми.	9 тиждень	ПЗ, Додатки
6	Оформлення пояснювальної записки та документів	10-11 тижні	
	до неї.		
7	Нормоконтроль та рецензування.	12 тиждень	
8	Захист роботи.	12 тиждень	

Студент(ка) _____ Володимир ЧЕРНЕЦОВ
(підпис) (Імя ПРІЗВИЩЕ)

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

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 103 с., 2 табл., 32 рис., 4 дод., 20 джерел.

КАМЕРА ЗГОРЯННЯ, АВІАЦІЙНІЙ ДВИГУН, КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, НЕГЛИБОКІ МОДЕЛІ, НЕЙРОННІ МЕРЕЖІ.

Об’єкт дослідження – процес прогнозування якості роботи запальника.

Предмет дослідження – методи якісного прогнозування керованими даними.

Мета роботи – підвищення рівня автоматизації прогнозування якості роботи запальника камери згоряння авіаційного двигуна.

Матеріали, методи та технічні засоби: методи машинного навчання, методи глибокого навчання нейромереж, об’єктно-орієнтоване програмування, мова програмування Python 3.8, персональний комп’ютер з процесором AMD Ryzen 5 3600 під управлінням операційної системи Linux Ubuntu 24.04.

Результати. Створено методи прогнозування якості роботи запальника. Розроблені методи реалізовано у бібліотеці функцій на мові програмування Python, які реалізують прогнозування якості роботи запальника. Визначені напрямкі подальшої експериментальної роботи в напрямку покращеня

Висновки. Розроблено методи та програмну систему для автоматичного прогнозування якості роботи запальника.

Галузь використання – будування авіаційних двигунів, машинобудування.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 103 pages, 2 tables, 32 figures, 4 appendixes, 20 sources.

COMBUSTION CHAMBER, AIRCRAFT ENGINE, CLASSIFICATION, MACHINE LEARNING, SHALLOW MODELS, NEURAL NETWORKS.

Object of the study is the prediction process of an igniter performance.

Subject of the study are methods for quality prediction based on supervised data.

The purpose of the work is to improve the level of automation in predicting the aircraft engine combustion chamber igniter performance.

Materials, methods, and technical tools: machine learning methods, deep learning neural network methods, object-oriented programming, Python 3.8 programming language, a personal computer with an AMD Ryzen 5 3600 processor running the Linux Ubuntu 24.04 operating system.

Results. Methods for predicting of the igniter performance have been developed. The proposed methods are implemented in a library of functions written in Python, which perform prediction for the igniter. Directions for further experimental work improvement were defined.

Conclusions. Methods and a software system for automatic prediction of igniter performance have been developed.

Fields of application are aircraft engine construction and mechanical engineering.

ЗМІСТ

С.

Перелік скорочень та умовних позначень....8
Вступ....9
1 Аналіз предметної області....11
1.1 Постановка задачі....11
1.2 Опис підходів до вивчення бінарної класифікації....13
1.3 Неглибокі моделі....15
1.4 Нейромережевий підхід....24
1.5 Висновок за розділом 1....27
2 Розробка програми....28
2.1 Вимоги до програми....28
2.2 Вибір інструментарію розробки програмного забеспечення....28
2.3 Вибір підходів та шаблонів....30
2.4 Рекомендації до системних вимог....32
2.5 Структурна схема програми....32
2.6 Модуль аналізу вхідних даних....34
2.7 Модуль підготовки вхідних даних....35
2.8 Модуль тренування моделей....37
2.9 Модуль передбачення даних....38
2.10 Висновок за розділом 2....39
3 Експериментальне дослідження....41
3.1 Аналіз даних....41
3.2 Порівняння навчених моделей....45
3.3 Порівняння метрик моделей....50
3.4 Приклад запуску програми передбачення даних....51
3.4 Висновок за розділом 3....52
Висновки....53
Перелік джерел посилань....54
Додаток А Приклад файлу протоколу протоколу вимірювань....57

Додаток Б Набіри тренувальних даних.....59
Додаток В Текст програми.....69
Додаток Г Слайди презентації.....96

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

CART – Classification and Regression Trees (дерева класифікації та регресії);

CNN – Convolutional neural network (загорткова нейронна мережа);

KNN – *k*-nearest Neighbours (метод *k*-найближчих сусідів);

LASSO – Least Absolute Shrinkage and Selection Operator;

MLP – Multi Layer Perceptron (багатошарові нейронні мережі);

MAE – Mean Absolute Error (середня абсолютна похибка);

MSE – Mean Squared Error (середньоквадратична похибка);

SVM –Support Vector Machines (метод опорних векторів);

ДП «Івченко-Прогрес» – державне підпріємство «Івченко-Прогрес».

ВСТУП

Запуск авіаційного двигуна є критично важливим етапом його функціонування, оскільки від нього залежить безпека та надійність експлуатації авіаційної техніки. Процес запуску може піддаватися впливу різних факторів, таких як температура зовнішнього середовища, атмосферний тиск на вході двигуна (який обумовлений висотою над поверхнею моря у кожен конкретний момент здійснення польоту). Оскільки запуск та експлуатація двигуна здійснюється у дуже широкому діапазоні температур та висот, надійний запуск є критично важливим для авіаційної техніки. Запуск авіаційного двигуна марки АИ-450 здійснюється завдяки пристрою який називають запальник. Саме параметри запальника впливають на успішність запуску. Більшість параметрів є геометричними параметрами цього пристрою та проєктується із міркувань надійності та простоти пристрою. Деякі параметри можуть бути змінені, наприклад температура палива у запальнику або тиск палива.

У цій дипломній роботі розглянуто декілька математичних моделей які можуть передбачити можливість запуску двигуна згідно змінних параметрів пристрою. Оскільки кожен натурній експеримент із авіаційним двигуном є досить коштовною задачею, поставлена мета диплому – підвищення рівня автоматизації прогнозування якості роботи запальника камери згоряння авіаційного двигуна, яка зможе передбачити можливість запуску при змінних параметрах.

Для здійснення цієї задачі було отримано дані вимірювань на заводі ДП «Івченко-Прогрес» у цифровому вигляді. В подальшому вони були оброблені та на їх базі було побудовано декілька математичних моделей різними методами машинного навчання.

Для обробки даних було використано два базових набору методів глибокого навчання: неглибокі моделі (shallow models) та нейромережевий підхід. Також для пошуку найбільш оптимальних параметрів самих моделей

було використано моделі пошуку оптимальних значень методом прямого перебору.

Таким чином, тема даної магістерської роботи – актуальна і пов’язана з програмною реалізацією моделей оптимізації роботи камери згоряння авіадвигуна.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

Існує проблема запуску авіаційного двигуна в широкому діапазоні тисків і температур. Одним із ключових елементів у процедурі запуску й стабільної роботи двигуна є «запальник» – компонент камери згоряння, що відповідає за ініціювання процесу горіння. Його надійність і ефективність значно впливають на загальну працездатність двигуна. На рисунку 1.1 наведена фотографія вигляду запальника двигуна АИ-450.

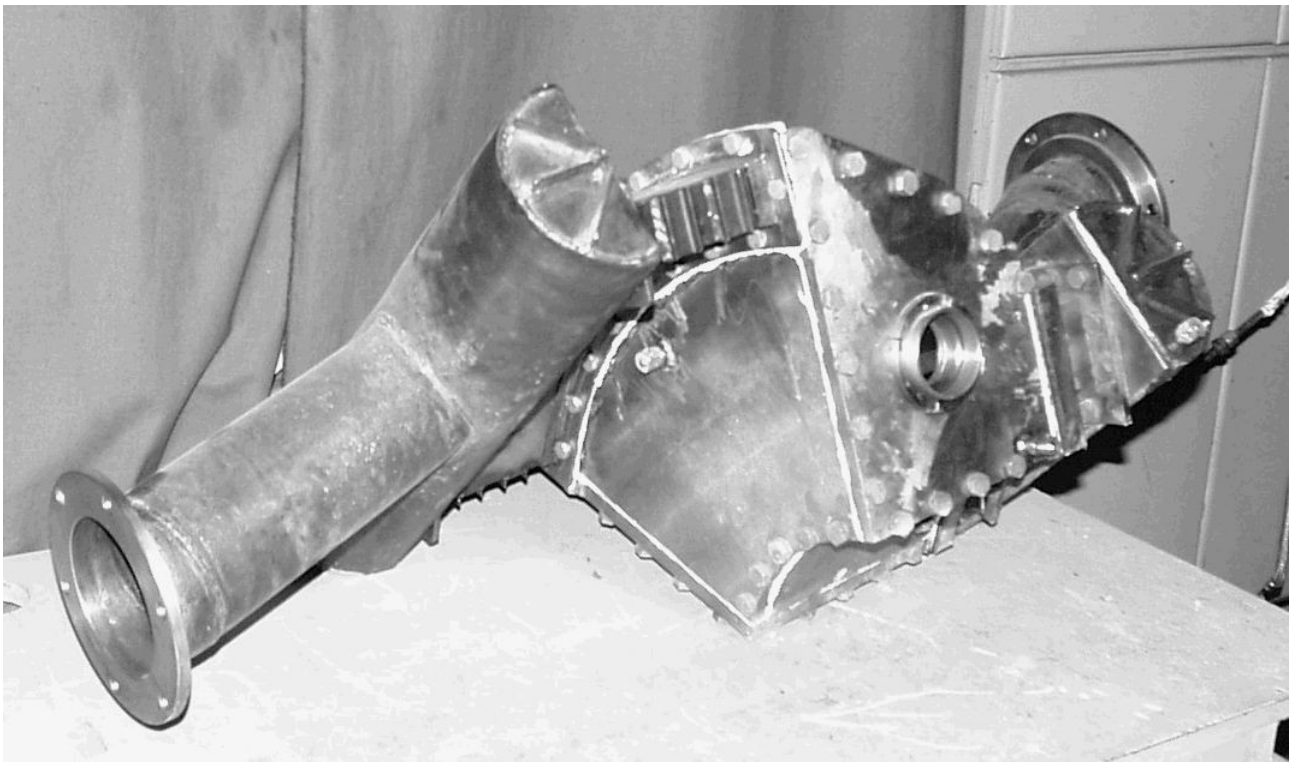


Рисунок 1.1 – Запальник авіаційного двигуна [1]

У 2014-2015 роках на ДП «Івченко-Прогрес» були проведені експериментальні дослідження роботи двигуна АІ-450 на тестовій установці випробувального стенда. Метою цих досліджень було вивчення впливу різних факторів на процес згоряння та збір експериментальних даних для подальшого аналізу. Зібрані дані дали змогу оцінити залежність поведінки камери згоряння від змінних навколишнього середовища (рис. 1.2) [1].

Рисунок 1.2 – Загальний вигляд запальника двигуна на тестовій установці [1]

Паралельно на кафедрі теорії авіаційних двигунів (ТАД) виконувалися чисельні експерименти, результатом яких стало створення регресійної моделі для прогнозування параметрів роботи запалювальних пристроїв двигуна (рис.1.3) [2].

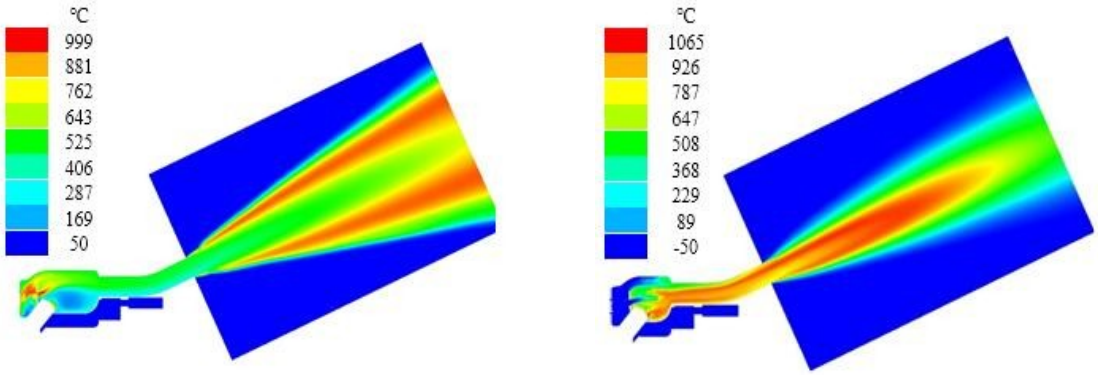


Рисунок 1.3 – Приклад модельованих полів температур для різних комбінацій початкових та граничних умов [2]

Попри отримані результати, створені моделі все ще мають обмеження, оскільки вони недостатньо враховують складну природу фізичних процесів у камері згоряння.

Через високу складність і витратність практичного експериментального підходу виникає потреба у створенні методів, які дозволять прогнозувати ефективність запальника без фізичного запуску двигуна. Це не лише скоротить витрати ресурсів, але й прискорить процес оптимізації, дозволяючи швидше впроваджувати вдосконалення.

Розвиток сучасних технологій, зокрема методів машинного навчання, відкриває нові можливості для побудови більш точних і універсальних моделей. Таким чином, постає задача розробки інструменту, який дозволить проводити аналіз і моделювання процесів роботи камери згоряння з урахуванням більшої кількості змінних і складних залежностей між ними.

Метою цієї дипломної роботи є підвищення рівня автоматизації прогнозування якості роботи запальника камери згоряння авіаційного двигуна а саме – створення програмного забезпечення, яке дозволяє використовувати різні алгоритми машинного навчання для побудови моделей роботи камери згоряння двигуна АІ-450. Програма повинна забезпечувати можливість навчання моделей по нових даних, отриманих у ході нових експериментів, їх оцінки та порівняння ефективності кожної моделі. Використання такого інструменту дозволить інженерам не лише прогнозувати поведінку камери згоряння, але й визначати оптимальні налаштування експериментальної установки для отримання нових значень. Таким чином, експерименти можливо буде проводити не навмання, а направлено, ще збереже кошти та час.

1.2 Опис підходів до вивчення бінарної класифікації

Завдання бінарної класифікації у контексті оптимізації роботи камери згоряння авіадвигуна передбачає визначення належності кожного із вимірювань стану системи до одного з двох станів: «горить» або «не горить» запальник.

Згідно вивчених протоколів, та після експертної консультації із спеціалістом, який займається цією роботою на ДП «Івченко-Прогрес» означено, що всі стани, які не позначені як «горить», треба вважати як «не горить». Треба зазначити що при деяких станах системи, розпалювання відбувається, або не стабільно, або не у спроєктованих для цього місцях, чого також треба запобігти. Такі стани також означені станом «не горить».

Існує декілька загальних методів дослідження даних із метою бінарної класифікації із використанням моделей машинного навчання: неглибокі моделі та нейроні мережі.

До методів неглибоких моделей машинного навчання належать:

- логістична регресія (Logistic Regression);
- дерева рішень (Decision Trees);
- метод опорних векторів (Support Vector Machines, SVM);
- гаусів наївний баєсів класифікатор (Gauss Naive Classificator);
- алгоритм k-найближчих сусідів (k-Nearest Neighbors, KNN).

Вони дозволяють отримати відносно точні результати при обмежених ресурсах і незначних обсягах даних. Їхньою ключовою перевагою є швидкість навчання та висока інтерпретованість. Наприклад, логістична регресія, як один із базових підходів, забезпечує зрозумілий математичний апарат для оцінки впливу факторів на кінцевий результат, а дерева прийняття рішень дають інтуїтивно зрозумілий графічний вигляд.

Іншою важливою групою методів є нейронні мережі, які пропонують потужні інструменти для аналізу складних і багатовимірних даних. Наприклад, багатошарові нейронні мережі (Multi-Layer Perceptron, MLP) можуть знаходити приховані залежності між факторами, що впливають на роботу запальника, дозволяючи враховувати складну природу процесу згоряння. Водночас нейромережеві моделі вимагають значно більших обчислювальних ресурсів і обсягів даних, що обмежує їхнє використання в умовах, коли дані є обмеженими. Для задач із великою кількістю параметрів або складною

структурою залежностей ці підходи можуть бути ефективними, але потребують ретельної оптимізації архітектури мережі та процесу навчання.

Також треба запобігати, або, враховувати, що моделі можуть бути перенавченими. Перенавчання моделі (або overfitting) – це ситуація, коли модель машинного навчання настільки добре адаптується до тренувальних даних, що починає запам'ятовувати деталі і випадкові шуми, замість того щоб виявляти загальні закономірності. У результаті модель демонструє високу точність на тренувальних даних, але погано працює на нових, невідомих даних (на тестових або в реальних умовах) [3].

1.3 Неглибокі моделі

1.3.1 Логістична регресія

Логістична регресія – це метод статистичного моделювання, який використовується для бінарної класифікації. Цей метод є різновидом лінійної регресії та застосовується у задачах класифікації, наприклад, для визначення, чи належить певне спостереження до категорії «так» чи «ні». Так само, як і модель лінійної регресії, модель логістичної регресії обчислює зважену суму вхідних ознак (додатково враховуючи вільний член), але замість того, щоб безпосередньо повертати отриманий результат, як це робить лінійна регресія, вона видає логістичну функцію цього результату. Формулу логістичної функції наведено у формулі (1.1):

$$\sigma(t)=\frac{1}{1+e^{(-t)}}, \tag{1.1}$$

та відповідний графік наведено на рисунку 1.4 [4].

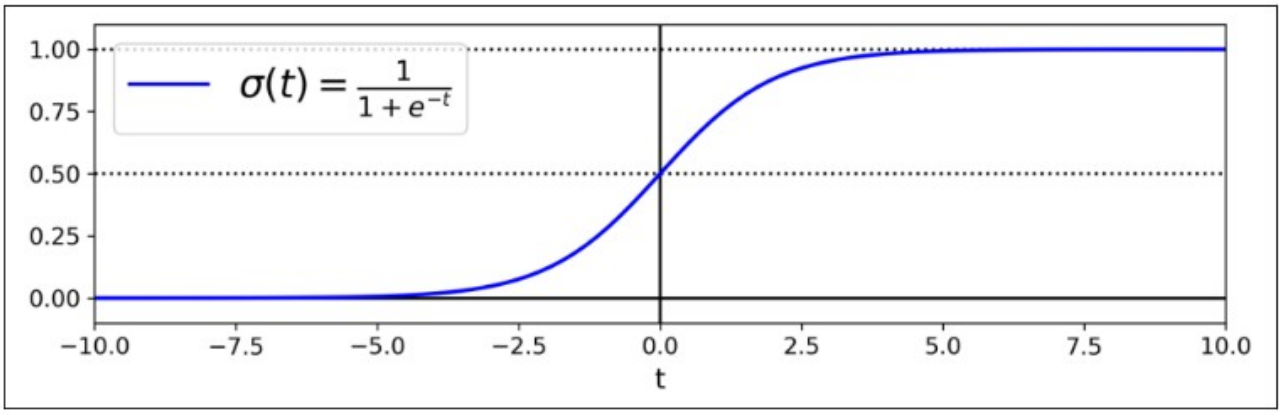


Рисунок 1.4 – Графік логістичної функції

Передбачення відбувається за правилом, наведеному у формулі (1.2):

$$y = \begin{cases} 0, \text{якщо } p < 0,5; \\ 1, \text{якщо } p \geq 0,5. \end{cases} \quad (1.2)$$

Для запобігання перенавчанню моделі (overfitting) та поліпшення її здатності узагальнювати результати на нових даних використовується техніка, яка називається регуляризація. Вона додає так званий штраф до функції втрат, що допомагає контролювати складність моделі, зменшуючи вплив надмірно великих коефіцієнтів.

У логістичній регресії застосовуються три типи регуляризації: Lasso та Ridge.

Lasso (Least Absolute Shrinkage and Selection Operator) – оператор найменшого абсолютного стиснення і вибору отримало свою назву через основні принципи, які лежать в його основі, яка підкреслює основну властивість цього підходу: одночасно скорочувати (shrinkage) та відбирати (selection) ознаки. Це особливо корисно у задачах з великою кількістю змінних, де важливо визначити, які з них мають найбільше значення для прогнозу, та виключити зайві [3]- [4].

Функція втрат із Lasso-регуляризацією наведена у формулі (1.3):

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} ¿\theta_i \vee ¿¿. \quad (1.3)$$

MSE (Mean Squared Error) - це міра середньоквадратичної помилки, яка використовується для оцінки якості моделі у задачах регресії. Вона обчислює середнє значення квадратів різниць між фактичними та передбаченими значеннями. Формула MSE наведена у формулі (1.4):

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \quad (1.4)$$

де n – кількість спостережень у наборі даних;

y_i – фактичне значення для i-го спостереження;

$\hat{y}_i$– передбачене значення для i-го спостереження.

Ridge-регуляризація – це метод, який додає до функції втрат штраф, пропорційний сумі квадратів коефіцієнтів моделі. Вона є поширеним підходом для зменшення перенавчання, особливо в ситуаціях, коли модель схильна до занадто сильного підстроювання під дані навчання. Ridge-регуляризація намагається зменшити величини коефіцієнтів моделі, але на відміну від Lasso-регуляризації, вона не «занулює» коефіцієнти. Замість цього вона «стискає» всі коефіцієнти ближче до нуля, зменшуючи їх вплив. Таким чином, модель стає менш чутливою до окремих ознак із великим значенням.

Завдяки квадратичному характеру штрафу, Ridge-регуляризація «м'яко» коригує коефіцієнти, без занулення. Це підходить, коли всі ознаки важливі для моделі, але їхній вплив потрібно згладити [3]- [4].

Функція втрат із Ridge-регуляризацєю наведена у формулі (1.5):

$$J(\theta) = MSE(\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2. \quad (1.5)$$

ElasticNet – це метод регуляризації, який об'єднує властивості Lasso - регуляризації та Ridge-регуляризації. Він додає до функції втрат обидва штрафні члени: як суму абсолютних значень коефіцієнтів (Lasso), так і суму квадратів коефіцієнтів (Ridge). Цей метод забезпечує баланс між відбором ознак і згладжуванням значень коефіцієнтів [5].

Функція втрат із ElasticNet-регуляризацією наведена у формулі (1.6):

$$J(\theta) = MSE(\theta) + r\alpha \sum_{i=1}^{n} \theta_i \vee + \alpha \frac{1-r}{2} \sum_{i=1}^{n} \theta_i^2. \quad (1.6)$$

Як видно із формули 1.6, ElasticNet поєднує Lasso та Ridge, за винятком, що у формулу додано так званий параметр мікс-співвідношення, який дозволяє точніше підлаштувати ENet до набору даних. Якщо *r=1*, то ElasticNet стає чистою Lasso-регуляризацією, якщо *r=0*, то ElasticNet стає чистою Ridge-регуляризацією [5].

1.3.2 Метод k-найближчих сусідів

Алгоритм k-найближчих сусідів (k-nearest Neighbours, KNN) – це простий, але потужний метод класифікації, який базується на ідеї пошуку найближчих сусідів у просторі ознак. Він належить до алгоритмів навчання без моделі, оскільки не будує явної моделі під час навчання, а запам'ятовує всі дані.

При класифікації KNN визначає клас об’єкта, орієнтуючись на класи його k-найближчих сусідів у просторі ознак. У випадку регресії прогнозується середнє значення цільової змінної для цих сусідів.

Для задачі класифікації клас нового спостереження визначається голосуванням сусідів: обирається клас, який зустрічається найчастіше серед k-сусідів [6].

Також слід зазначити що ознаки в набори даних цієї дипломної роботи мають різні діапазони, тому для нормальної роботи алгоритму необхідно виконувати нормалізацію даних.

Для кожного нового спостереження обчислюється відстань до кожного об’єкта з навчального набору. В дипломній роботі використано наступні метрики відстані: евклідова, мангеттенська та метрика Мінковського [7].

Евклідова метрика визначає прямолінійну відстань між двома точками у просторі. Формула евклідової метрики відстані наведена у формулі (1.7):

$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}. \quad (1.7)$$

Мангеттенська метрика визначає суму абсолютних різниць між координатами. Формула мангеттенської метрики наведена у формулі (1.8):

$$d(x,y)=\sum_{i=1}^{n} ¿ x_i-y_i \vee ¿ ¿. \quad (1.8)$$

Метрика Міньковського узагальнює Евклідову та Манхеттенську відстані, вводячи параметр *p*, який визначає порядок норми. Формула метрики Міньковського наведена у формулі (1.9):

$$d(x,y)=(\sum_{i=1}^{n} ¿ x_i-y_i \vee ¿^p)^{\frac{1}{p}} ¿. \quad (1.9)$$

1.3.3 Гаусів наївний баєсівський класифікатор

Гаусів наївний баєсівський класифікатор є частиною сімейства наївних баєсівських алгоритмів, які підходять для роботи з безперервними даними. Гаусів наївний баєсівський класифікатор припускає, що ознаки розподілені

згідно з нормальним розподілом. Він навчається, обчислюючи середнє значення та стандартне відхилення кожної ознаки для кожного класу.

Під час прогнозування гаусів наївний баєсівський класифікатор порівнює кожну нову точку даних зі статистикою для кожного класу. Для кожного класу обчислюється ймовірність належності об'єкта, виходячи з його ознак і збережених статистичних параметрів. Потім обирається клас із найвищою ймовірністю. Цей підхід забезпечує простоту та швидкість, але його точність може знижуватися, якщо припущення про нормальний розподіл даних не відповідають реальності [8].

Формула щільності ймовірності Гаусового розподілу наведена у формулі (1.10):

$$P(x_i|y) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \qquad (1.10)$$

де x – значення ознаки x, для якого обчислюється ймовірність;

μ – середнє значення ознаки x для класу y;

σ – стандартне відхилення ознаки x для класу y;

σ^2 – дисперсія ознаки x для класу y.

1.3.4 Метод опорних векторів

Метод опорних векторів (Support Vector Machines, SVM) – це потужний алгоритм для класифікації, регресії та задач виявлення аномалій. Його основна ідея полягає в пошуку гіперплощини, яка максимально розділяє точки різних класів у просторі ознак. SVM відомий своєю ефективністю для задач із невеликою кількістю даних і високою розмірністю простору ознак. Це досягається шляхом використання лише тих точок даних, які лежать найближче до гіперплощини розділення – вони називаються опорними векторами. У випадках, коли дані нелінійно роздільні, SVM використовує ядерні функції

(наприклад, радіальну базисну функцію, поліноміальне або лінійне ядро), щоб перетворити ознаки в простір вищої розмірності, де розділення стає можливим.

Під час навчання метод опорних векторів визначає, наскільки важливим є кожне із навчальних спостережень для представлення межі розділення між двома класами. Зазвичай тільки підмножина навчальних точок має значення для визначення цієї межі – це точки, які лежать на межі між класами. Вони називаються опорними векторами і дають назву методу опорних векторів.

Розділювальна гіперплощина задається рівнянням наведеним у формулі (1.11) [9]:

$$f(x) = wx + b. \tag{1.11}$$

Мета методу SVM визначається за формулою (1.12):

$$minimize \frac{1}{2} w^T w = minimize \frac{1}{2} ¿ \vee w \vee ¿^2, \tag{1.12}$$

за умов $y_i(w^T x_i + b) \geq 1 \; для \; 1, 2, 3 \ldots, m,$

де w – вектор ваг гіперплощини;

b – зсув;

x_i – точки даних;

y_i – мітки класів (1 або 0).

1.3.5 Дерево прийняття рішень

Дерева прийняття рішень (Decision Trees, або CART - Classification and Regression Trees) – це інтуїтивно зрозумілий і потужний алгоритм для класифікації та регресії, який будує модель у вигляді дерева. Алгоритм поступово розділяє дані на підгрупи, використовуючи умови на значення ознак,

поки всі підгрупи не стануть максимально однорідними (з точки зору цільової змінної) [10].

Під час навчання алгоритм визначає, яка ознака найбільше зменшує невизначеність у розподілі цільової змінної. Вузли дерева створюються шляхом поділу даних за цією ознакою. Процес триває рекурсивно, розділяючи дані на менші підгрупи в кожному вузлу. Розгалуження тривають доти, поки не досягнуто умов зупинки, таких як мінімальний розмір групи або максимальна глибина дерева.

Для передбачення нових спостережень алгоритм проходить від кореневого вузла до листа дерева, вибираючи шлях залежно від умов, визначених у кожному вузлу. Лист, у якому зупиняється спостереження, містить результат – клас або значення регресії [11]-[12].

У якості критерію вибору вузла (розщеплення) використовують інформаційну ентропію або критерій Gini.

Інформаційна ентропія – це міра невизначеності даних, яка використовується при побудові дерев рішень. Ентропія характеризує середній ступінь непередбачуваності стану джерела повідомлення.

Формула інформаційної ентропії наведена у (1.13) [13]:

$$H(S) = -\sum_{i=1}^{c} \frac{1}{N} \log_2 \left(\frac{N!}{\prod N_i!} \right), \tag{1.13}$$

де N – кількість об'єктів;

$H(S)$ – ентропія множини S;

C – кількість класів;

p_i – частота (ймовірність) належності об'єкта до класу i в множині S.

Критерій Gini - визначає ймовірність того, що об’єкт, вибраний випадково з набору даних, буде неправильно класифікований, якщо його класифікувати випадково відповідно до розподілу класів [10].

Формула критерію Gini наведена у (1.14) [14]:

$$gini(T) = 1 - \sum_{i=1}^{n} p_i^2, \tag{1.14}$$

де T – поточний вузол;

p_j – імовірність класу j у вузлу T;

n – кількість класів.

Приклад графічного виду дерева прийняття рішень наведено на рисунку 1.5 [15].

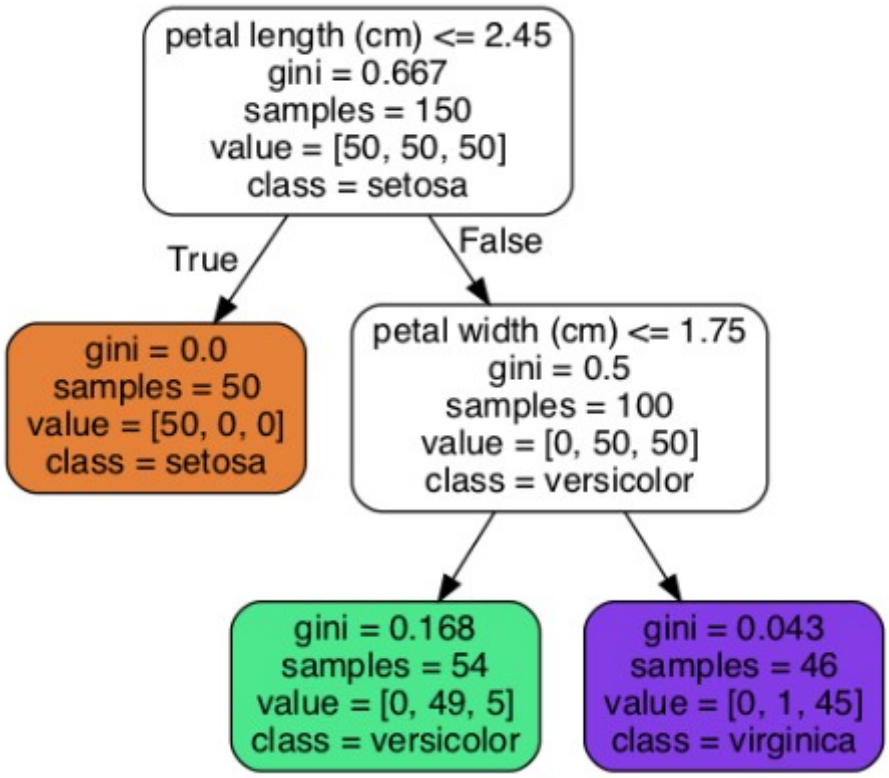


Рисунок 1.5 – Приклад дерева прийняття рішень для задачі класифікації ірисів [15]

Різні критерії вибору вузлів, такі як інформаційна ентропія чи критерій Gini, мають свої переваги та недоліки залежно від задачі та характеру даних. Наприклад, ентропія враховує ступінь невизначеності розподілу, тоді як Gini є швидшим у розрахунках і часто дає подібні результати. Вибір критерію

розщеплення може впливати на якість побудованого дерева та його здатність до узагальнення на тестових даних. Таким чином, деревоподібні моделі залишаються популярним інструментом у задачах машинного навчання завдяки своїй простоті, інтуїтивності та здатності працювати з нелінійними залежностями між ознаками та цільовою змінною.

1.3.6 Ансамблі дерев прийняття рішень

Random Forest – це ансамблевий метод машинного навчання, який будує велику кількість дерев рішень для вирішення задач класифікації та регресії. Метод використовує випадковість на двох рівнях: при виборі підвибірки даних для кожного дерева та при виборі підмножини ознак під час розбиття вузлів. Таким чином, кожне дерево навчається на унікальній частині даних і робить незалежні передбачення. У задачах класифікації результат визначається голосуванням дерев, а у задачах регресії – усередненням їхніх прогнозів.

Головною перевагою Random Forest є його стійкість до шуму та висока здатність до узагальнення. Завдяки незалежності дерев метод добре працює з великою кількістю ознак, а також на незбалансованих і шумних вибірках. Крім того, Random Forest надає можливість визначити важливість ознак, що дозволяє оцінити їхній вплив на модель, роблячи цей підхід не лише потужним, а й інтерпретованим інструментом [16].

1.4 Нейромережевий підхід

1.4.1 Багатошаровий персептрон

Штучні нейронні мережі є популярними методами машинного навчання, що імітують підхід до обробки інформації через взаємодію елементів. У таких мережах основними елементами є обчислювальні одиниці, які називаються нейронами. Ці одиниці з'єднані між собою вагами, які визначають, як дані передаються між нейронами.

Кожен вхід до нейрона масштабується за допомогою ваги, що впливає на функцію, яку обчислює нейрон. Штучна нейронна мережа виконує обчислення, передаючи дані через шари нейронів – від вхідного шару до вихідного. Процес навчання відбувається шляхом корекції ваг у мережі для досягнення кращих результатів.

Мета зміни ваг – модифікувати обчислювану функцію таким чином, щоб зробити передбачення більш точними у наступних ітераціях. Тому ваги змінюються ретельно, математично обґрунтованим способом, щоб зменшити помилку обчислення для цього прикладу. Послідовно коригуючи ваги між нейронами для багатьох пар «вхід-вихід», функція, яку обчислює нейронна мережа, вдосконалюється з часом, що дозволяє отримувати точніші передбачення. Схема мультишарового персептрону наведена на рисунку 1.6 [17].

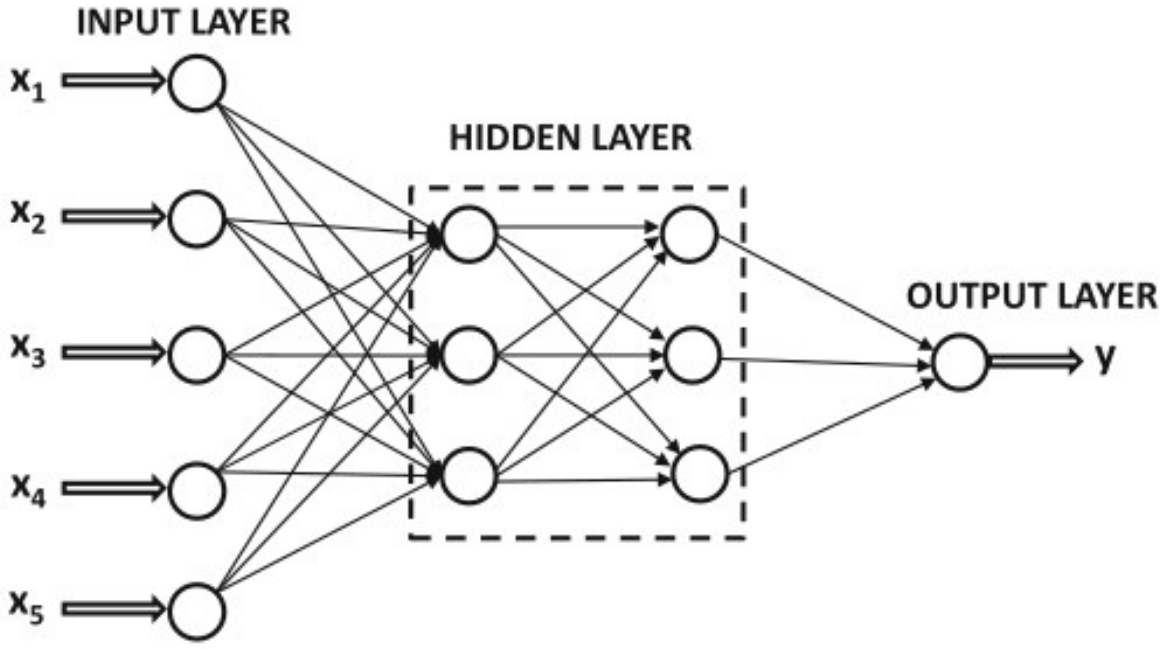


Рисунок 1.6 – Схема многошарового персептрона [17]

Попри переваги, нейромережеві моделі мають обмеження, зокрема високу потребу в обчислювальних ресурсах і складність інтерпретації результатів. Однак ці недоліки компенсуються високою точністю і здатністю

адаптуватися до різних типів даних, що робить нейронні мережі універсальним інструментом для аналізу та прогнозування в багатьох галузях [18].

1.4.2 Конволюційна нейромережа

Згорткові нейронні мережі (Convolutional Neural Network, CNN), є різновидом глибинних нейронних мереж прямого поширення, адаптованих для обробки зображень. Ці мережі натхненні організацією зорової кори тварин, де окремі нейрони реагують на стимули в межах обмежених рецептивних полів, які частково перекриваються, забезпечуючи охоплення всього зорового поля. ЗНМ включають шари входу та виходу, а також приховані шари, що зазвичай складаються зі згорткових, агрегувальних, повноз’єднаних шарів та шарів нормалізації.

Основною функцією згорткового шару є виконання операції згортки, яка імітує реакцію нейрона на візуальні стимули. Шар має набір навчальних фільтрів (або ядер) з малим рецептивним полем, які здійснюють згортку даних вхідного сигналу, утворюючи карти активації. Ці карти визначають активацію фільтрів у відповідь на специфічні ознаки у певних просторових положеннях. Усі карти активації разом формують об’єм вихідних даних шару, де кожен вихід відповідає реакції нейрона на частину вхідного зображення.

Агрегувальні шари (pooling layers) зменшують просторовий розмір подання, агрегуючи виходи нейронів попереднього шару. Це сприяє зниженню обчислювальних витрат, зменшенню кількості параметрів і контролю перенавчання. Агрегування забезпечує інваріантність до зсувів і допомагає мережі зосереджуватись на загальних ознаках замість їх точного розташування. У згорткової нейромережі ці шари зазвичай чергуються зі згортковими шарами, виконуючи обробку кожного зрізу глибини незалежно. Схема згорткової нейромережі наведена у рисунку 1.7 [19].

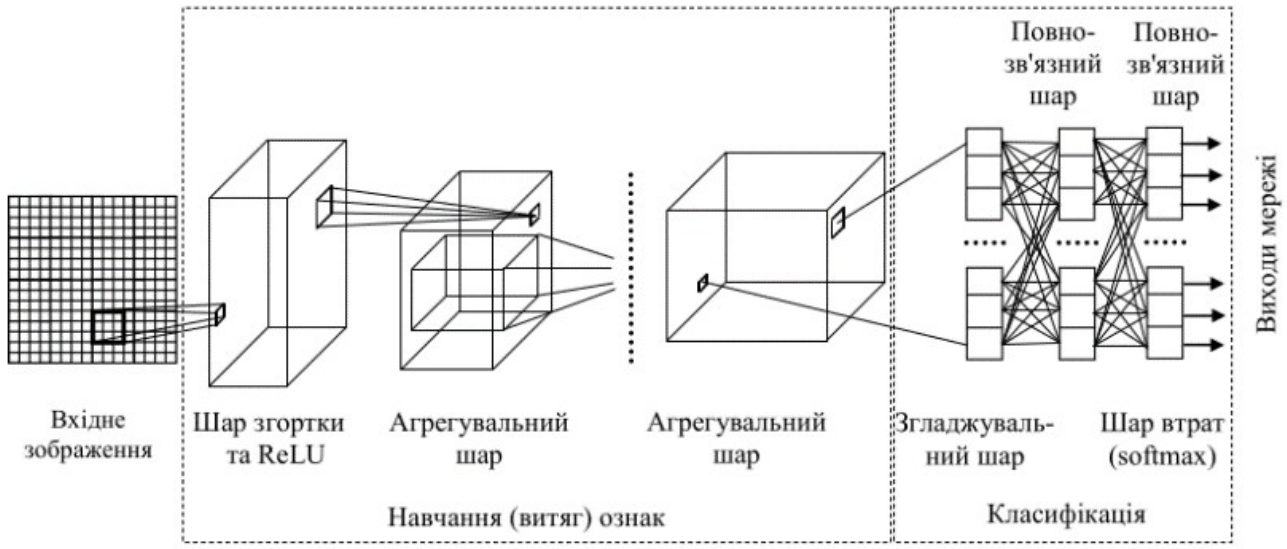


Рисунок 1.7 – Загальна схема згорткової нейромережі [19]

1.5 Висновок за розділом 1

У процесі вивчення бінарної класифікації в задачі аналізу роботи камери згоряння авіадвигуна було виявлено, що як неглибокі моделі, так і нейронні мережі мають свої переваги, які можуть бути корисними. Неглибокі моделі забезпечують простоту реалізації, швидкість навчання та інтерпретованість результатів, тоді як нейромережеві підходи пропонують більшу гнучкість і здатність виявляти складні залежності у великих наборах даних. Щоб вибрати оптимальний підхід, у даній дипломній роботі проводиться порівняльний аналіз обох методів на основі конкретних даних і критеріїв ефективності, що дозволить забезпечити найбільш точне та надійне прогнозування роботи запальника.

Оскільки дані мають обмежений обсяг, гарна інтерпретованість, наприклад, дерева рішень, може бути важливою. Тому неглибокі моделі будуть кращим вибором. Також варто зазначити, що недостатня кількість даних у наявному наборі може вплинути на точність моделювання та обмежити ефективність використання нейромережевих алгоритмів, які потребують великого обсягу даних для навчання та забезпечення надійних результатів.

2 РОЗРОБКА ПРОГРАМИ

2.1 Вимоги до програми

Розробка програми передбачає створення інструменту, що забезпечує ефективну автоматизацію задач аналізу даних та моделювання. Основними вимогами до програми є її функціональність, продуктивність, зручність у використанні, а також можливість адаптації до нових задач і розширення функціоналу.

Програма має включати модулі для аналізу та обробки даних, реалізовані у вигляді структурованих послідовностей інструкцій. Це дозволить автоматизувати процеси, пов’язані зі збором інформації з протоколів, очищенням і формуванням таблиць даних, а також визначенням змін у файлах.

Система повинна мати гнучкий інтерфейс для роботи з різними моделями машинного навчання, що дозволить легко інтегрувати нові алгоритми або замінювати існуючі.

Зручність використання досягається завдяки зрозумілому інтерфейсу, можливості налаштування параметрів обробки та чіткому виведенню результатів. Програма повинна бути багатоплатформною, підтримувати роботу на Windows і Linux, а також мати легкий програмний інтерфейс для інтеграції з середовищами розробки, такими як Jupyter Notebook, для спрощення дослідницьких задач.

2.2 Вибір інструментарію розробки програмного забеспечення

Вибір мови програмування для розробки програмного забезпечення значною мірою залежить від вимог до проєкту, доступних інструментів і можливостей інтеграції. У задачах, пов’язаних із моделюванням, машинним навчанням або аналізом даних, популярними альтернативами є MATLAB, R та

Octave. Кожна з цих мов має свої переваги. Порівняння мов програмування можно знайти у табліці 2.1.

Таблиця 2.1 – Таблиця порівняння мов програрамування

Крітерії порівняння	Мови програмування			
	Python	Mathlab	R	Octave
Відкрите ПЗ	+	-	+	+
Багатоплатформність	+++	++	++	++
Вбудовані математичні операції	+	+++	++	+++
Наявні математичні бібліотки	+	+	+	+
Наявні нейромережеві бібліотекі	+	+	+	+
Легкість інтеграції (веб/GUI/консоль)	+++	+	+	+
Підтримка модульністі	+++	+	++	+
Підтримка парадігми ООП	+++	+	+	+
Спільнота та документація	+++	+	++	+

MATLAB добре підходить для чисельного моделювання, аналізу сигналів і розв’язання складних математичних задач. Його основними перевагами є високий рівень інтеграції для математичних операцій і багатий набір вбудованих функцій. Однак MATLAB є пропрієтарним програмним забезпеченням, що може створювати обмеження через високу вартість ліцензій, особливо для розгортання проєктів у масштабі.

R в основному використовується в статистичному аналізі й візуалізації даних. Завдяки великій кількості пакетів і активній спільноті, R залишається потужним інструментом для дослідників і аналітиків. Проте R має обмежену підтримку для розгортання моделей у реальних додатках або інтеграції з іншими мовами, що зменшує його універсальність у багатофункціональних проєктах.

Octave є відкритим аналогом MATLAB і забезпечує подібний функціонал. Це безкоштовне середовище корисне для навчання і створення

прототипів, але воно поступається MATLAB за продуктивністю і набором інструментів. Крім того, підтримка спільноти й екосистеми Octave значно менша.

Python обрано із кількох ключових причин. По-перше, Python - це кросплатформна мова, яка підтримується на всіх основних операційних системах і забезпечує легке розгортання програм. По-друге, Python надає можливість створювати додатки для різних сфер застосування: від консольних утиліт до вебдодатків або інтеграції у Jupyter Notebook для дослідницьких задач. По-третє, багата екосистема бібліотек (NumPy, SciPy, Pandas, Scikit-learn, TensorFlow) дозволяє швидко вирішувати задачі будь-якої складності. В-четверте, Python забеспечую повну модульність та парадигму ООП.

Таким чином, вибір Python зумовлений його гнучкістю, доступністю та широкими можливостями для реалізації як дослідницьких, так і прикладних проєктів.

У процесі роботи використовуються такі інструменти та бібліотеки:

- Jupyter Notebook – забезпечів зручне інтерактивне середовище для розробки, тестування та візуалізації;
- Scikit-learn використовувався для реалізації моделей машинного навчання та їх оцінки. Ця бібліотека забезпечує широкий вибір алгоритмів, простий інтерфейс і зручні методи для попередньої обробки даних, вибору гіперпараметрів і оцінки якості моделей;
- TensorFlow дозволив створювати та навчати нейронні мережі;
- Matplotlib та Seaborn використовувалися для візуалізації результатів і дослідження поведінки даних.

2.3 Вибір підходів та шаблонів

У процесі аналізу предметної області, даних і поставлених завдань у межах розробки програмного проєкту було обрано два поведінкові шаблони проєктування з числа описаних у класичній роботі «Design Patterns» («Шаблони

об’єктно-орієнтованого проєктування», так звана «Книга банди чотирьох»): «Ланцюжок відповідальностей» (Chain of Responsibility) [20] і «Стратегія» (Strategy) [20]. Ці шаблони були адаптовані до специфіки задачі для досягнення максимального рівня модульності, зручності підтримки та масштабованості.

Шаблон «Ланцюжок відповідальностей» у цьому проєкті реалізовано не в класичній формі передачі об'єкта між процедурами, а у вигляді набору дій над даними, де самі дані виступають як об'єкт. Такий підхід дозволяє описувати кілька схожих потоків маніпуляцій над даними за допомогою мінімальної кількості інструкцій. У програмному коді цей шаблон структурує весь процес – від завантаження даних і пошуку змін до підготовки таблиці для моделей машинного навчання. Це усуває дублювання коду, полегшує налагодження та знижує ймовірність помилок.

Другий обраний шаблон, «Стратегія», забезпечує об'єднання схожих алгоритмів у рамках одного сімейства, де кожен алгоритм реалізується у власному класі. Це дозволяє використовувати різні алгоритми під час виконання програми, змінюючи їх без втручання в основну структуру системи. У цьому проєкті шаблон «Стратегія» забезпечив уніфікований підхід до роботи з алгоритмами машинного навчання, де всі моделі, незалежно від їхньої складності чи типу, мали однаковий набір методів – .train(), .predict() і .estimate().

Кожен із обраних підходів сприяв модульності та зручності розширення системи. Наприклад, шаблон «Ланцюжок відповідальностей» дозволив розробляти кожен етап обробки як незалежний компонент, який легко адаптувати чи замінити без впливу на загальну логіку програми. Це дало змогу налаштовувати систему під нові завдання, оптимізувати обробку даних і покращувати алгоритми маніпуляції із даними, не змінюючи базової архітектури.

Шаблон «Стратегія» спростив інтеграцію нових алгоритмів машинного навчання. Для додавання нового алгоритму достатньо було реалізувати інтерфейс із відповідними методами, не змінюючи наявний код. Такий підхід

зробив систему гнучкою, масштабованою та стійкою до змін, водночас зберігаючи її простоту й узгодженість.

У підсумку, використання цих шаблонів дало змогу створити модульну систему, яка легко піддається адаптації, розширенню та підтримці. Така архітектура облегшує майбутні модифікації.

2.4 Рекомендації до системних вимог

Python є багатоплатформною мовою програмування, тому жорсткі вимоги до операційної системи відсутні. Можна використовувати будь-яку ОС, яка підтримує Python версії 3.8 і вище, наприклад, Windows 10 або 11, а також будь-який сучасний дистрибутив Linux. Під час навчання моделей було відзначено значне навантаження на центральний процесор, тому рекомендовано використовувати CPU з максимальною кількістю ядер і потоків для покращення швидості навчання моделей.

Вимоги до оперативної пам’яті залишаються помірними, оскільки обсяг даних невеликий і без проблем розміщується в пам’яті сучасного настільного комп’ютера. Також перевірено, що моделі, які використовують бібліотеку TensorFlow, можуть успішно задіяти тензорні ядра графічних процесорів серії RTX, якщо такі є доступні. Загалом, система не потребує високих специфікацій, і проєкт можна запускати на більшості сучасних комп’ютерів без додаткових вимог.

2.5 Структурна схема програми

Для спрощення роботи системи та впорядкування її окремих логічних і функціональних компонентів було запроваджено модульну структуру. Графічний вигляд структурної схеми програми можна побачити на рисунку 2.1.

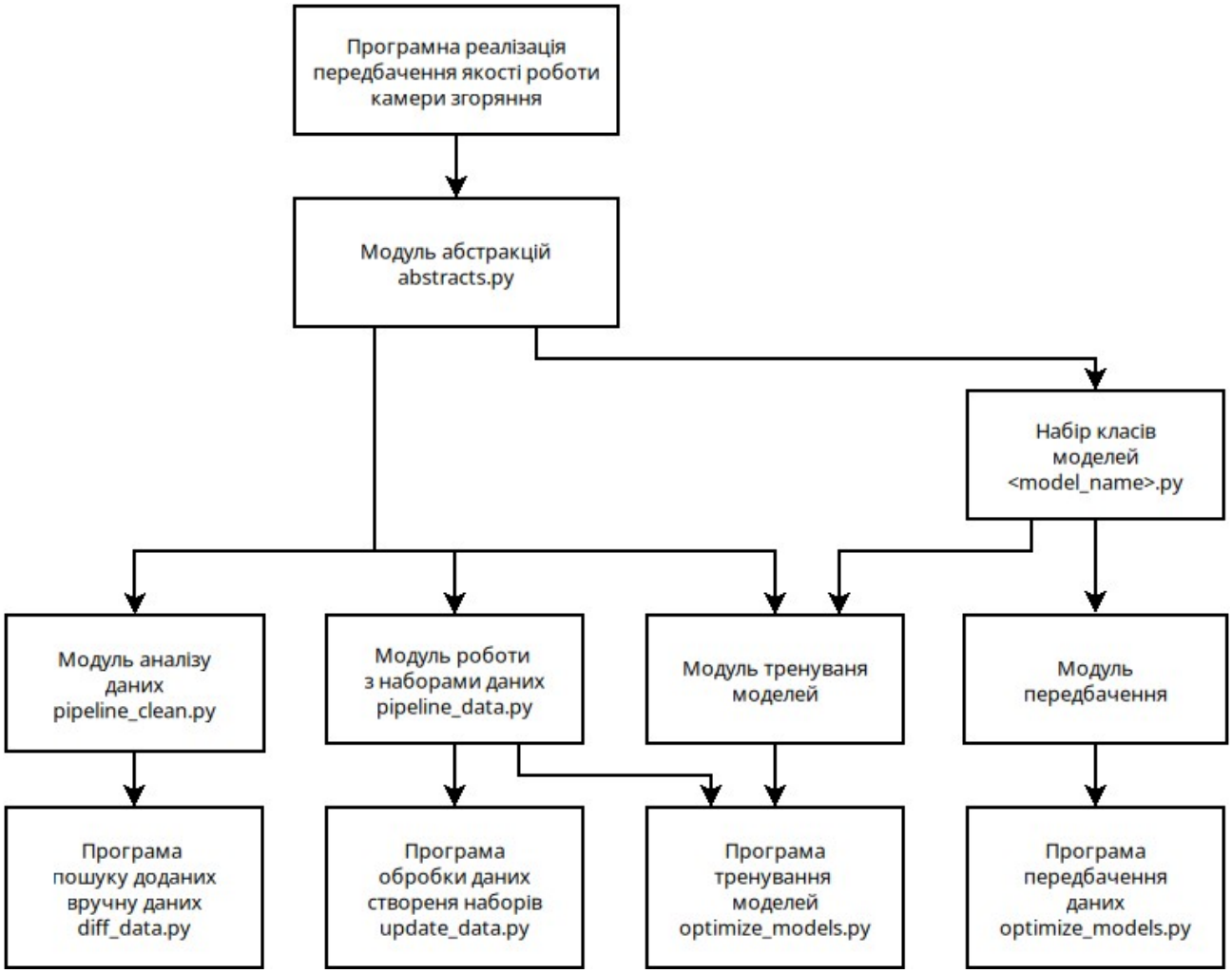


Рисунок 2.1 – Структурна схема програми

У результаті система включає такі модулі:

- abstracts.py містить абстрактні класи, які відповідальні за різні аспекти системи, а саме: читання/вивід даних, моделей та мета-інформації моделей;
- constants.py містить константи проєкту;
- utils.py містить утилітарні функції, які не можна класифікувати до якогось модулю;
- main.ipynb – Jypiter Notebook який містить додаткові функцій для вивчення даніх;
- cnn.py – містить клас моделі CNN;
- decision_tree.py – містить клас моделі дерева прйняття рішень;

- gaussian_classifier.py – містить клас моделі гаусова наївного класифікатора;
- knn_classifier.py – містить клас моделі kNN;
- logistic_regression.py – містить клас моделі логістичної регресії;
- mlp.py – містить клас моделі мультішарового персептрона;
- random_forest.py – містить клас моделі Random Forest;
- svm_classifier.py – містить клас моделі Support Vector Machine;
- pipeline_clean.py – містить клас, який виконує пошук різниці даних;
- pipeline_data.py – містить клас, який оперує наборами даних;
- diff_data.py – консольна програма, яка виконує пошук даних;
- update_data.py – консольна програма, яка оновлює набори даних;
- optimize_models.py – консольна програма, яка шукає та зберігає кращі моделі;
- estimate_data.py – консольна програма, яка виконує передбачення.

Кожна з наведених консольних програм виконує одну персональну функцію згідно із шаблоном «ланцюжок відповідальності». Кожен із ціх модулів описаний у розділах 2.6-2.9.

2.6 Модуль аналізу вхідних даних

Данні представлені у вигляді набору протоколів вимірювань у файлах типу «*.rtf». Приклад протоколу наведено у Додатку 1. Кожен файл протоколу було зібрано із використанням специфічного програмного забезпечення розробки ЗМКБ “Прогрес”, тому можливо сказати що файли мають повторювану структуру та можуть бути також розібрані на змінні. Треба зазначити, що деякі файли було редаговано вручну – додана фраза «В воздухоподводящем отверстии установлено...», що виявлено в ході попереднього аналізу даних. Присутність цієї строки у протоколи означає, що перед запуском установки, у вхідному отворі повітряного патрубка було

встановлено відповідна кількість металічних дротів описаного діаметру із метою знизити площу вхідного повітряного патрубка.

Для визначення різниці у протоколах була розроблена програма diff_data.py. Вона визначає нестандартні строки у файлах та просто виводить їх у консоль. Функціональна схема програми для аналізу файлів протоколів наведена на рисунку 2.2.

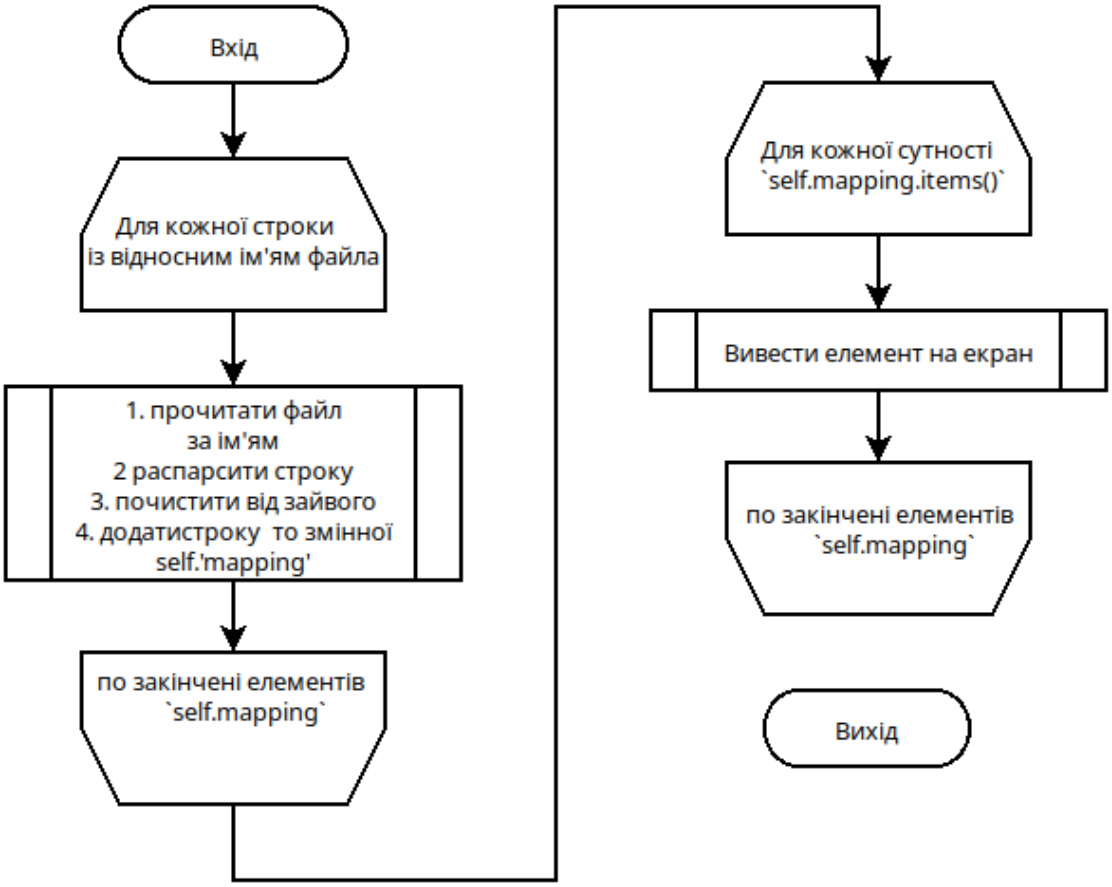


Рисунок 2.2 – Схема функціонування програми diff_data.py

2.7 Модуль підготовки вхідних даних

Для ефективної обробки та підготовки вхідних даних із файлів протоколів був розроблений модуль у вигляді консольної програми update_data.py. Ця програма автоматизує процес збору, парсингу та збереження набору даних. Функціональна схема роботи програми update_data.py наведена на рисунку 2.3.

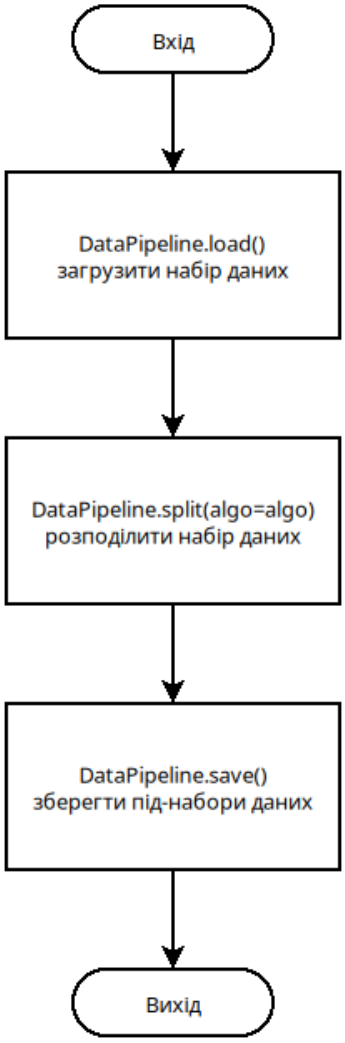


Рисунок 2.3 – Схема роботи програми update_data.py

Програма виконує рекурсивне читання всіх файлів протоколів у вказаній директорії, обробляючи кожен файл построково. Виділені дані зберігаються у два окремих файли: train.csv та val.csv. Перший файл використовується для тренування моделей, а другий – для валідації. Стандартним методом розподілу є виділення кожного п'ятого рядка з вихідного набору у файл val.csv, тоді як інші записи додаються до файлу train.csv. Таким чином, забезпечується співвідношення між тренувальними та валідаційними даними у пропорції 80% до 20%.

Такий підхід гарантує стабільність і відтворюваність результатів при кожному запуску програми за умови незмінності початкового набору протоколів. Це спрощує відстеження результатів моделювання та забезпечує узгодженість у процесі підготовки даних.

Однак у деяких випадках може виникнути потреба змінити порядок даних, щоб досягти кращої продуктивності моделей. Для цього в програмі передбачено додатковий ключ командного рядка --algo=random, який перемішує записи в початковому наборі перед їх розподілом між файлами train.csv і val.csv. У цьому випадку співвідношення між тренувальними та валідаційними даними також зберігається на рівні 80% до 20%.

2.8 Модуль тренування моделей

Для кожної математичної моделі в рамках проєкту було створено окремий клас, який відповідає за вибір оптимальних параметрів, тренування моделі та повернення найкращого варіанта моделі за визначеною метрикою. Такий підхід забезпечує гнучкість та універсальність, дозволяючи адаптуватися до різних типів моделей і завдань.

Також на цьому етапі визначається набір метрик навчання моделей та зберігається на диску.

Практично всі моделі, за винятком згорткових нейронних мереж, були створені з використанням інструмента scikit-learn, що спростило інтеграцію методів машинного навчання. Для пошуку найкращих параметрів моделей використовувався метод прямого перебору параметрів із використанням класу GridSearchCV. Цей клас реалізує вичерпний перебір вказаних значень параметрів для обраного естіматора (моделі). Схема работи програми, яка знаходить оптимальні моделі зображено на рисунку 2.4.

Процес пошуку найкращих параметрів з використанням GridSearchCV включає:

- визначення діапазонів значень або наборів можливих параметрів для моделі;
- побудову всіх можливих комбінацій параметрів;
- послідовне тренування моделі з кожною комбінацією параметрів на навчальному наборі даних;

- оцінювання кожного варіанту моделі за метрикою «accuracy»;
- вибір найкращої моделі на основі результатів оцінки.

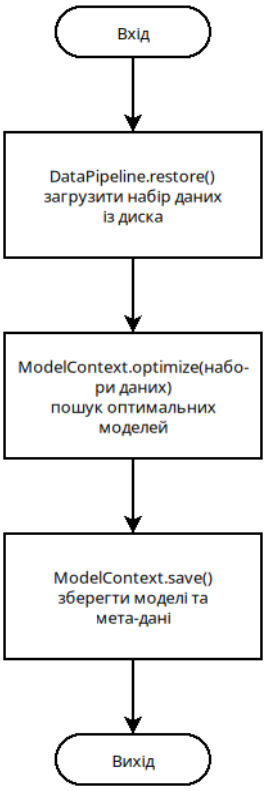


Рисунок 2.4 – Схема роботи програми optimize_models.py

Завдяки автоматизації цього процесу клас GridSearchCV значно полегшив оптимізацію моделей, забезпечуючи їхню високу якість. Крім того, використання цього підходу зменшило кількість ручних налаштувань та зробило тренування моделей більш ефективним. Для пошуку оптимальної моделі було розроблено програму optimize_models.py. Схема роботи програми наведено на рисунку 3.3:

2.9 Модуль передбачення даних

Модуль передбачення даних призначений для використання моделей, які були навчені за допомогою програми optimize_models.py. Основна функція цього модуля – забезпечити можливість застосування збережених моделей для

передбачення результатів на нових наборах даних. Для цього була розроблена консольна програма estimate_data.py, яка виконує цю задачу.

Програма estimate_data.py приймає обов'язкові аргументи, що відповідають параметрам, які використовувалися під час навчання моделей. Це гарантує коректність процесу передбачення, оскільки забезпечується узгодженість між параметрами навчання і передбачення.

У процесі виконання програма викликає метод .predict() для кожної з навченої моделі. Результати передбачення виводяться у консоль із відповідними ремарками. Зокрема, програма оцінює, чи є модель є перенавчаною (overfitting), і повідомляє метрику точності (accuracy) для кожної моделі. Це дозволяє оцінити якість роботи моделі на нових даних та зробити висновки про її придатність передбачення подальшого використання.

Програма побудована таким чином, щоб забезпечити зручність і прозорість для користувача. Інформація, що виводиться в консоль, включає:

- оцінку точності моделі на новому наборі даних;
- застереження щодо можливої перенавченості моделі;
- узагальнені рекомендації щодо ефективності моделі для конкретного набора даних.

Такий підхід забезпечує інтеграцію моделі в загальну систему та її ефективне використання для практичних задач. Модуль передбачення даних є ключовим компонентом у життєвому циклі роботи моделей, оскільки він дозволяє перевірити продуктивність та приймати рішення на основі результатів передбачення.

2.10 Висновок за розділом 2

Застосування шаблонів розробки, таких як «Ланцюжок відповідальностей» та «Стратегія», дозволило створити гнучку, модульну та масштабовану систему. Такий підхід забезпечує автоматизацію рутинних

процесів, спрощує інтеграцію нових компонентів і сприяє структурованості коду. Це робить систему легкою в підтримці та адаптації до змінних умов.

Системні вимоги для роботи програми залишаються помірними завдяки ефективному використанню ресурсів. Python, як багатоплатформна мова програмування, дозволяє запускати проєкт на будь-якій сучасній операційній системі. Оптимальним є використання процесора з високою кількістю ядер і потоків для пришвидшення обчислень, а також відеокарти з підтримкою тензорних ядер для навчання моделей. У поєднанні з невисокими вимогами до оперативної пам’яті це дозволяє запускати проєкт на більшості сучасних комп’ютерів, забезпечуючи доступність і зручність роботи з системою.

У ході роботи було створено комплексну систему для обробки, підготовки, тренування та передбачення даних, представлених у вигляді файлів протоколів. Використання програм, таких як diff_data.py для аналізу вхідних даних і update_data.py для їхньої підготовки, забезпечило стандартизовану обробку протоколів та формування тренувальних і валідаційних наборів даних.

Для оптимізації моделей було розроблено програму optimize_models.py, яка автоматизує пошук найкращих параметрів за допомогою класу GridSearchCV. Це забезпечило ефективний підбір моделей та їхню високу якість. Використання єдиної метрики оцінки (accuracy) дозволило об’єктивно порівнювати різні моделі та вибирати оптимальні варіанти для подальшої роботи.

Для передбачення даних, був реалізована відповідна програма estimate_data.py. Вона дозволяє легко застосовувати навчені моделі для нових наборів даних, забезпечуючи зручний вивід результатів у консоль із додатковими застереженнями щодо якості моделі. Завдяки цьому система є зручною у використанні, гнучкою до змін і придатною для вирішення широкого спектра практичних завдань.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

3.1 Аналіз даних

3.1.1 Попередній аналіз даних

Для отримання карти налаштувань діаметру отворів необхідно скористатися програмою diff_data.py, яка була розроблена для аналізу протоколів даних. Ця програма здійснює порівняння вхідних файлів, визначаючи нестандартні строки, що містять інформацію про встановлені діаметри отворів. Приклад роботи програми наведений на рисунку 3.1:

```
$ python diff_data.py
data/Воспл. 4500355000-01 № 108/2015_03_18 Воспл. 4500355000-01 № 108/protA3.rtf
Режимы 5, 7, 14 - запуски воспламенителя 0140305000
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_04_03 Воспл. 4500355000-01 № 095/1/protA3.rtf
В воздухоподводящем отверстии установлено 10 проволок: из,них 9 штук d=1,5 мм., и 1 штука d=1мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_04_03 Воспл. 4500355000-01 № 095/2/protA3.rtf
В воздухоподводящем отверстии установлено 9 проволок: из,них 8 штук d=1,5 мм., и 1 штука d=1мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_04_07 Воспл. 4500355000-01 № 095/protA3.rtf
В воздухоподводящем отверстии установлено 9 проволоктd=1,5,мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_03_26 Воспл. 4500355000-01 № 095/1/protA3.rtf
В воздухоподводящем отверстии установлено 6 проволоки,d=1,5 мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_03_26 Воспл. 4500355000-01 № 095/2/26_03_14/protA3.rtf
В воздухоподводящем отверстии установлено 7 проволоки,d=1,5 мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_03_27 Воспл. 4500355000-01 № 095/protA3.rtf
В воздухоподводящем отверстии установлено 8 проволоки,d=1,5 мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_03_27 Воспл. 4500355000-01 № 095/2/protA3.rtf
Ā āīēäóōīļīäāīä ̀łåģ īņāåšńņčč óńņąķīāėåķī 9 ļšīāīėīź,d=1,5 ģģ.,Čēģåš ̊åģūå č šąń÷åņķūå ļąšąģåņšū:
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_03_31 Воспл. 4500355000-01 № 095/protA3.rtf
В воздухоподводящем отверстии установлено 10 проволок,d=1,5 мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_03_25 Воспл. 4500355000-01 № 095/protA3.rtf
В воздухоподводящем отверстии установлено 5 проволоки,d=1,5 мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_03_20 Воспл. 4500355000-01 № 095/1/protA3.rtf
В воздухоподводящем отверстии установлено 3 проволоки,d=1,5 мм.
********************************************************************************
data/Воспл. 4500355000-01 № 095/2014_03_20 Воспл. 4500355000-01 № 095/2/protA3.rtf
В воздухоподводящем отверстии установлено 4 проволоки,d=1,5 мм.
********************************************************************************
```

Рисунок 3.1 – Вивід програми diff_data.py

Після завершення цього етапу оператор повинен скоректувати значення словника (dict) SQUARE_MAP у модулі constants.py таким чином, щоб у ключі словника було ім’я файлу, а у значені - вирахувана відповідна площа впускного отвору запальника. Вірно заповнений словник містить назви файлів як ключі, а відповідні площі впускних отворів – як значення.

3.1.2 Аналіз характеристик даних

У цільовому наборі даних представлені такі вхідні параметри:

- площа отвору запальника (поле: square);
- висота польоту (поле: altitude);
- перепад тиску у кільцевому каналі (поле: delta_P_kk);
- перепад тиску палива (поле: delta_P_fuel);
- температура повітря (поле: air_temp);
- температура палива (поле: fuel_temp);
- температура факела горіння (поле: torch_temp);
- час горіння факела (поле: torch_time);
- температура запалення (поле: ignition_temp).

Усі вхідні параметри подані або приведені до формату числових змінних типу numpy.float, що дозволяє безпосередньо використовувати їх для математичних розрахунків і моделювання. Цільова змінна має формат boolean і відображає результат експерименту (значення «горить» або «не горить»).

Набір даних містить 308 записів, із яких 246 використовується у тренувальному піднаборі, а 62 – у валідаційному. Набір даних у графічному вигляді наведено на рисунку 3.2.

Рисунок 3.2 – Набір даних у графічному вигляді

Теплова карта кореляції параметрів, наведена на рисунку 3.3, демонструє взаємозв’язок між числовими змінними та цільовою змінною. Як видно з графіка, серед усіх параметрів найбільш тісний зв’язок із цільовою змінною (factor) має параметр delta_P_kk (Перепад тиску у кільцевому каналі). Значення кореляції для delta_P_kk показує помірний негативний зв’язок(k=-0.46), що означає помірну залежність між цим параметром та ймовірністю позитивного результату «запалювання». Таким чином, зниження тиску у кільцевому каналі може суттєво звищити якість роботи запальника. Також є

вже слабша позитивна кореляція між температурою факелу, температурю займання та цільовою змінною.

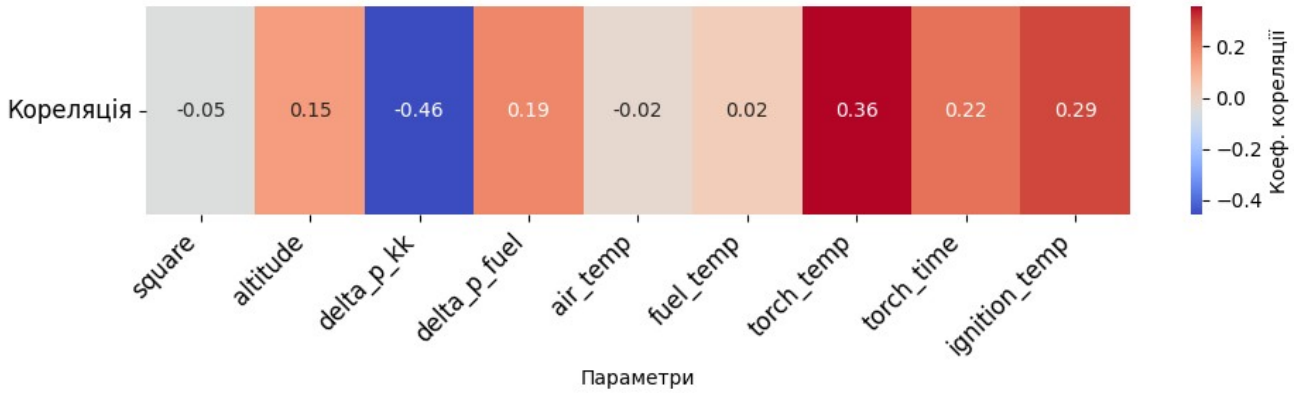


Рисунок 3.3 – Теплова карта кореляції

На сітці графіків (т.з. pairplot) на рисунку 3.4 зображена залежність між найбільш корельований значеннями параметрів та цільової змінної.

На основі pairplot можна зробити висновки про розподіл параметрів та їхній взаємозв’язок із цільовою змінною factor. Для параметра delta_P_kk видно, що менші значення частіше відповідають позитивному результату (factor = 1), тоді як більші значення характерні для випадків factor = 0. Параметри torch_temp і ignition_temp демонструють зворотну тенденцію: вищі значення цих параметрів частіше пов’язані з позитивним результатом. Розподіл даних свідчить про те, що ці параметри можуть мають важливий вплив на результат.

Взаємозв’язки між параметрами показують, що torch_temp і ignition_temp мають позитивну залежність: підвищення температури факела зазвичай супроводжується збільшенням температури запалення. Для delta_P_kk спостерігається розподіл, за якого високі значення тиску пов’язані з низькими значеннями температури факела та запалення. Це вказує на значущість цих параметрів для моделювання, зокрема torch_temp і ignition_temp як ключових позитивних чинників, а delta_P_kk – як чинника з негативним впливом.

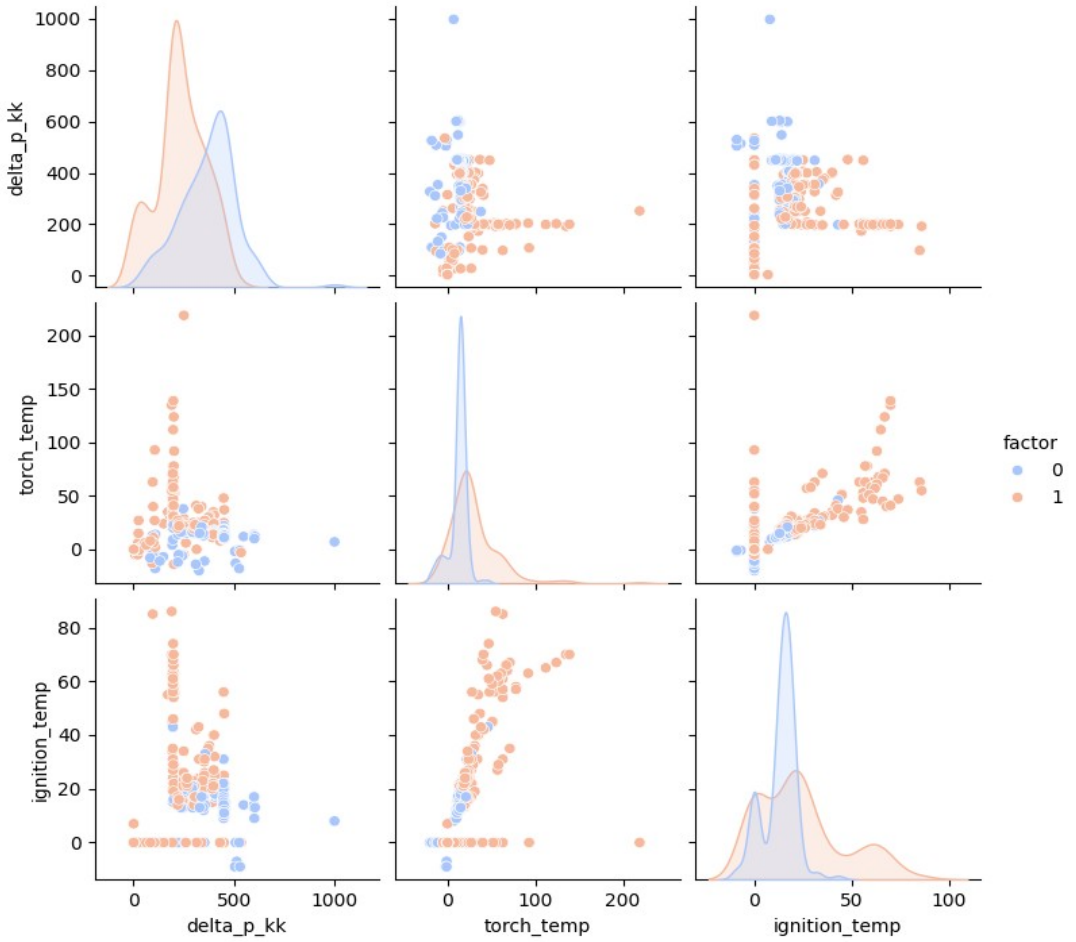


Рисунок 3.4 – Pairplot для полів delta_P_kk, torch_temp та ignition_temp

3.2 Порівняння навчених моделей

3.2.1 Логістична регресія

Найкращі параметри навчання моделі: «C»: 0.01, «l1_ratio»: 0.01, «max_iter»: 100, «penalty»: «l1», «solver»: «liblinear».

Отриманий поліном логістичної регресії наведений у формулі (3.1).

$$\begin{aligned} Z = {} & 0.01 \cdot square + 0.01 \cdot altitude - 0.01 \cdot delta\_p\_kk + \\ & + .00 \cdot delta\_p\_fuel + 0.00 \cdot air\_temp - \\ & - 0.01 \cdot fuel\_temp + 0.08 \cdot torch\_temp + \\ & + 0.00 \cdot torch\_time + 0.01 \cdot ignition\_temp. \end{aligned} \tag{3.1}$$

Як можна зазначити, із усіх вагових коефієнтів, найбільший вплив має torch_temp (температура факела). Решта – відображаються на результаті або слабо або ніяк.

3.2.2 Метод k-найближчих сусідів

Найкращі параметри навчання моделі:

- «metric»: «euclidean»;
- «n_neighbors»: 9;
- «weights»: «distance».

3.2.3 Гаусів наївний баєсівський класифікатор

Найкращі параметри навчання моделі: «var_smoothing» = 0.001.

Отримані дані середніх значень параметрів моделі зібрані у таблиці 3.1.

Таблиця 3.1 – середні значення моделі гаусова наївного баєсівського класифікатора

Параметр	Середнє (горить)	Середнє (не горить)
square	37,22	36,6
altitude	0,95	19,61
delta_p_kk	372,23	234,6
delta_p_fuel	3,6	4,41
air_temp	5,91	4,85
fuel_temp	0,5	0,63
torch_temp	12,11	30,53
torch_time	0,02	2,37
ignition_temp	13,32	24,07

Як видно із наведеної таблиці 3.1:

- altitude (висота): Значна відмінність між класами, може бути основною ознакою для їх розділення. Тобто, згідно оцінки цієї моделі висота є значущою ознакою;
- delta_p_kk (перепад тиску): Горіння пов'язане з вищими значеннями перепаду тиску;
- torch_time (час роботи запальника): Тривалий час роботи пальника пов'язаний із класом «не горить».

3.2.4 Метод опорних векторів

Найкращі параметри навчання моделі:

- «C»: 100;
- «gamma»: 0,01;
- «kernel»: «sigmoid».

3.2.5 Дерево прийняття рішень

Найкращі параметри навчання моделі:

- «criterion»: «entropy»;
- «max_depth»: 3;
- «max_features»: null;
- «min_samples_leaf»: 4;
- «min_samples_split»: 2.

Графік отриманого дерева прийняття рішень наведено у рисунку 3.5.

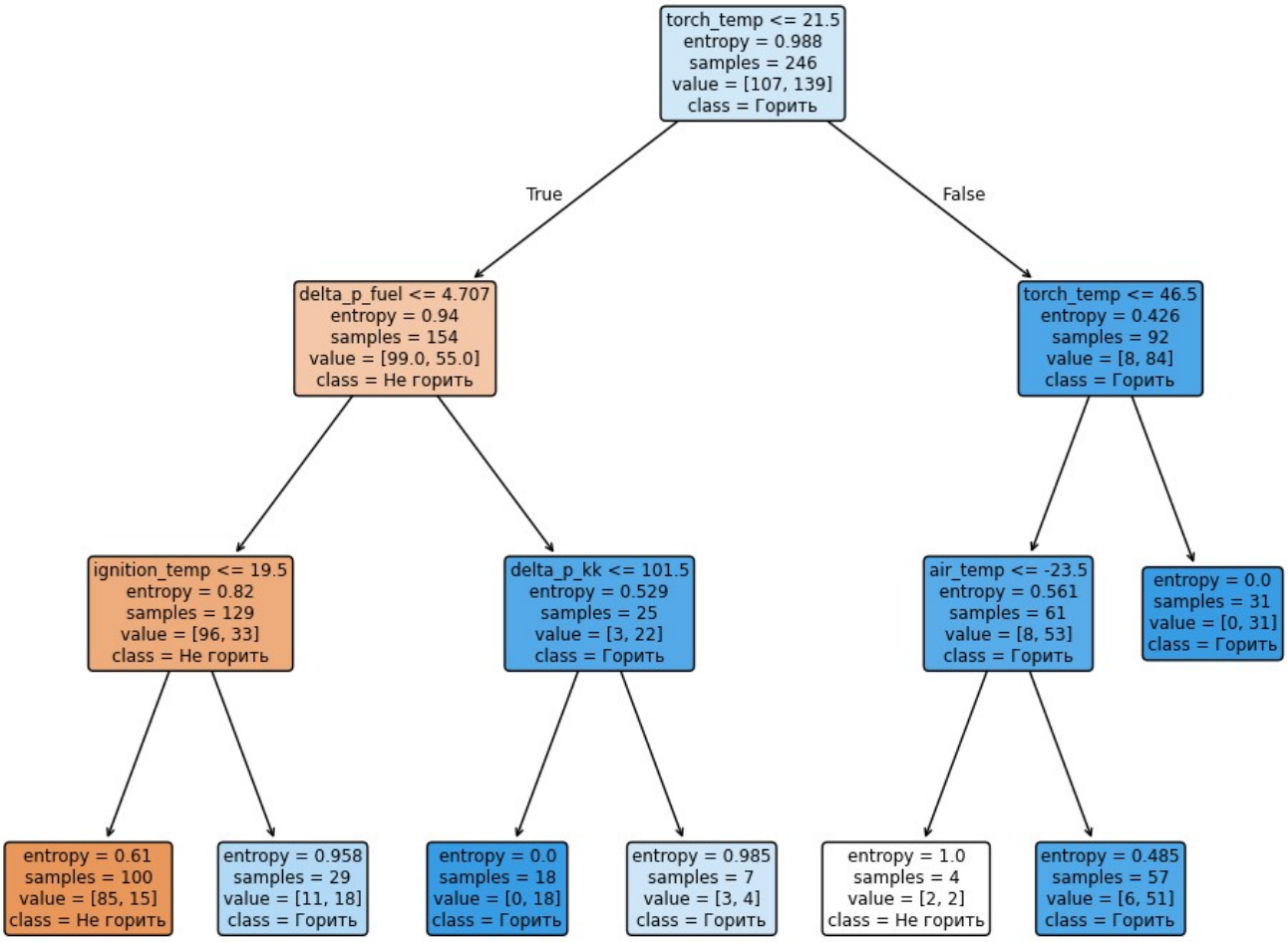


Рисунок 3.5 – Дерево прийняття рішень

3.2.6 Ансамблі дерев прийняття рішень

Найкращі параметри навчання моделі:

- «bootstrap»: true;
- «class_weight»: «balanced»;
- «max_depth»: 7;
- «max_features»: «sqrt»;
- «min_samples_leaf»: 4;
- «min_samples_split»: 2;
- «n_estimators»: 100.

Графік отриманого дерева прийняття рішень наведено у рисунку 3.6.

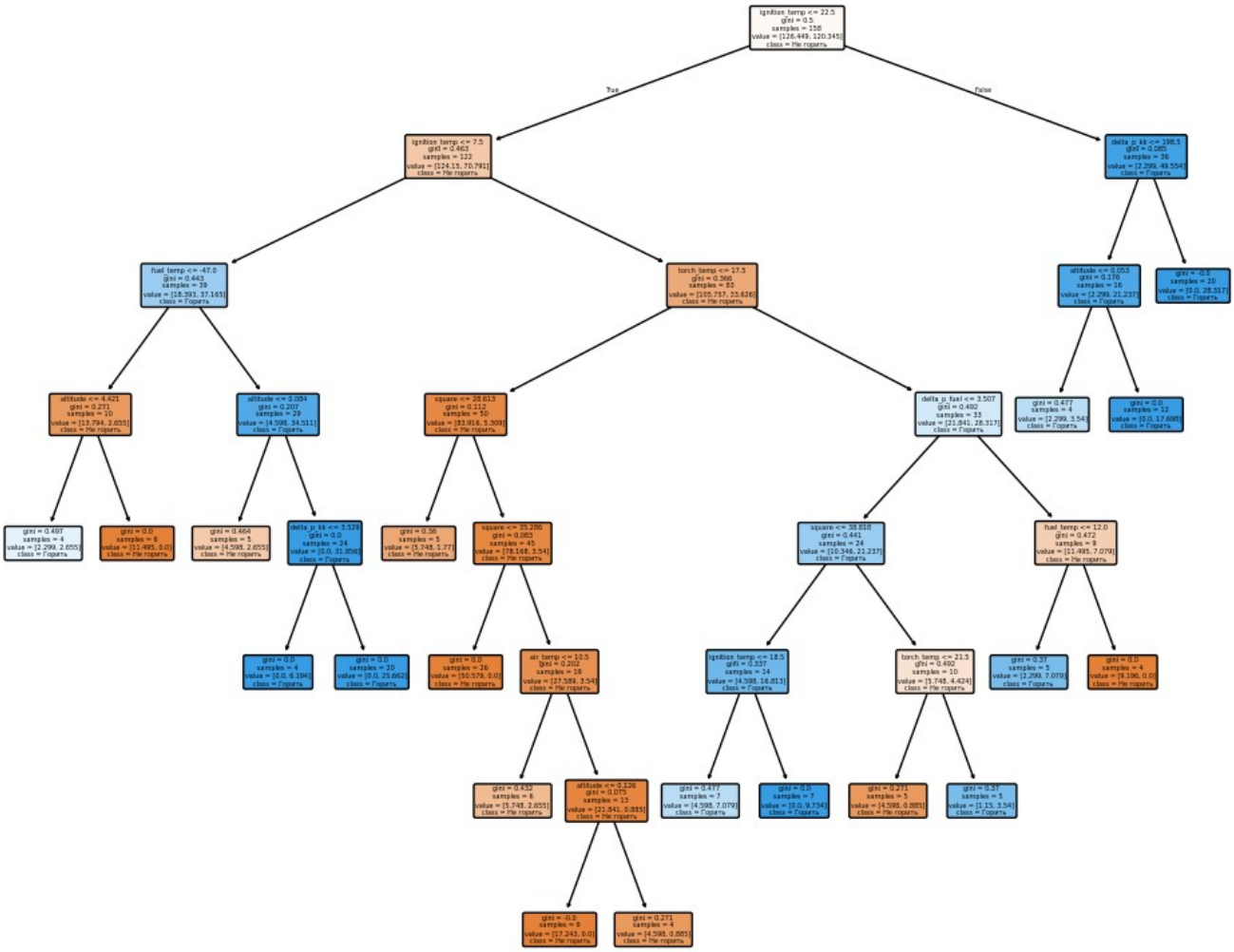


Рисунок 3.6 – Дерево прийняття рішень, знайдене алгоритмом Random Tree

3.2.7 Мультишаровий персептрон

Найкращі параметри навчання моделі:

- «activation»: «tanh»;
- «alpha»: 0,01;
- «batch_size»: 64;
- «hidden_layer_sizes»: 64;
- «learning_rate»: «constant»;
- «solver»: «adam».

3.3 Порівняння метрик моделей

На рисунку 3.7 представлені показники продуктивності різних моделей машинного навчання.

Метрики включають:

- accuracy (точність) – загальна частка правильно класифікованих прикладів;
- precision (прецизійність) – частка істинно позитивних передбачень серед усіх позитивних;
- recall (чутливість) – частка істинно позитивних передбачень серед усіх реальних позитивних прикладів;
- F1-score – гармонійне середнє між precision і recall;
- ROC-AUC – площа під ROC-кривою, що оцінює здатність моделі розрізняти класи;
- MAE (Mean Absolute Error) – середнє абсолютне відхилення, що демонструє середню похибку моделі.

На графіку видно результати для таких моделей: «Logistic Regression», «GaussianNB», «Support Vector Classifier», «Random Forest», «Decision Tree», «Multi Layer Perceptron», «Convolutional Neural Network». Модель «k-nearest Neighbors» виключена із зображення, оскільки є перенавченою и її подальше дослідження не має сенсу.

Найкраща загальна продуктивність спостерігається у моделі «Random Forest», яка демонструє найвищі значення для метрик «accuracy», «precision», «recall», «f1», «roc_auc» і найнижче значення MAE (0.08). Це вказує на те, що ця модель є ефективною та стабільною для задачі класифікації на цьому наборі даних.

Multi Layer Perceptron також показує високі результати, поступаючись Random Forest, але перевершуючи інші моделі. Це робить її хорошим вибором для задачі в умовах обмежених ресурсів.

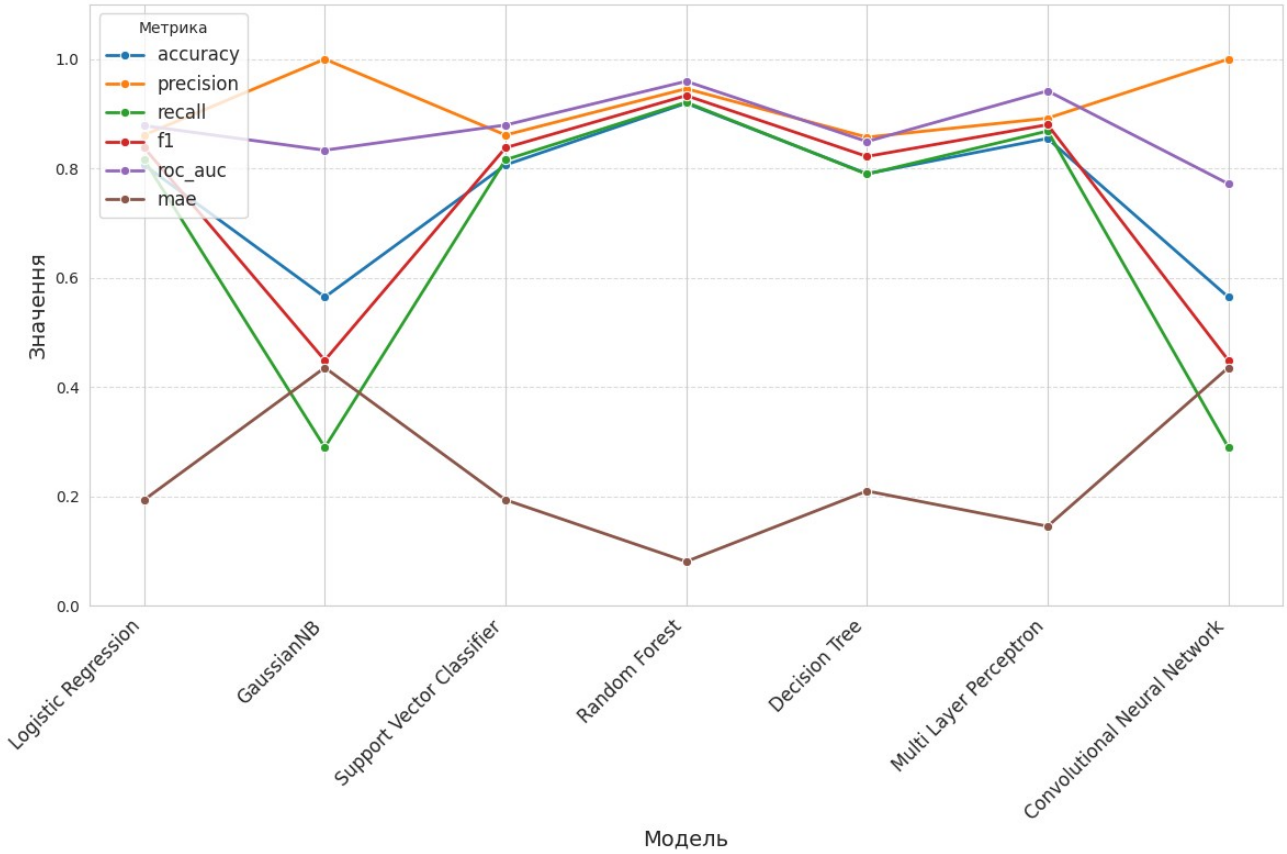


Рисунок 3.7 – Метрики не перенавчених моделей

GaussianNB і Convolutional Neural Network мають найнижчі значення метрик accuracy, f1 та recall, що вказує на низьку здатність до ефективної класифікації у цьому наборі даних.

Support Vector Classifier забезпечує добру збалансованість між точністю та іншими метриками, демонструючи стабільні результати, особливо у метриках precision і roc_auc.

Decision Tree є компромісним варіантом із середніми показниками продуктивності, що робить його менш привабливим порівняно з Random Forest.

3.4 Приклад запуску програми передбачення даних

Приклад роботи програми наведено на рисунку 3.8. Запуск команди проводився для стандартно розподілених даних, із аргументами $python

estimate_data.py --square=32 --altitude=0.0 --delta-p-kk=95 --delta-p-fuel=3.5 --air-temp=2 --fuel-temp=-34.0 --torch-temp=64 --torch-time=2 --ignition-temp=0

Модель	Передбачення	Точність моделі	Перенавчена
Logistic Regression	Горить	0.806452	Ні
GaussianNB	Горить	0.564516	Ні
K nearest neighbors	Горить	0.83871	Так
Support Vector Classifier	Горить	0.806452	Ні
Random Forest	Не горить	0.919355	Ні
Decision Tree	Горить	0.790323	Ні
Multi Layer Perceptron	Горить	0.854839	Ні
Convolutional Neural Network	Горить	0.564516	Ні

Таким чином, із 4 точних моделей 3 передбачило позитивний результат, а 1 - негативний

Рисунок 3.8 – Приклад передбачення результатів запуску запальника різними моделями

3.4 Висновок за розділом 3

Відповідно до попереднього аналізу даних, подальші експериментальні дослідження доцільно спрямувати на зменшення перепаду тиску у кільцевому каналі та підвищення температури факелу й температури займання.

На основі аналізу метрик моделей, у межах дослідження рекомендовано використовувати передбачення, отримані за допомогою моделей «Random Forest», «Multi Layer Perceptron» та «Support Vector Classifier», оскільки вони демонструють найкращі результати за обраними метриками.

ВИСНОВКИ

У роботі вирішено актуальну задачу розробки математичних моделей для оптимізації параметрів камери згоряння та запальника авіадвигуна. Отримані моделі дозволяють підвищити надійність запуску двигуна в широкому діапазоні умов експлуатації, враховуючи різні висоти та температури. Проведено аналіз технічної літератури та матеріалів, пов’язаних із бінарною класифікацією.

Наукова цінність роботи є в тому, що запропоновано підхід, що дозволяє оптимізувати елементи камери згоряння з використанням методів математичного моделювання.

Програма має практичну цінність, оскільки сприятиме скороченню витрат на проведення реальних експериментів та підвищенню точності визначення ключових параметрів, які впливають на стабільність запуску авіадвигуна.

Удосконалено методи аналізу ефективності роботи камери згоряння шляхом врахування значущих ознак, отриманих із розроблених математичних моделей. Це дозволило уточнити напрямки подальших експериментальних досліджень на підприємстві.

Практичне застосування розроблених моделей може бути розширено на інші задачі машинобудування та суміжних технічних галузей. Це дозволяє оптимізувати процеси розробки конструкцій, знизити фінансові витрати та скоротити час на досягнення оптимальних рішень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Торба, Ю.И. Экспериментальная установка и метод исследования работоспособности факельных воспламенителей в широких диапазонах имитируемых эксплуатационных характеристик [Текст] / Ю.И. Торба // Вестник двигателестроения. – 2006. – № 4. – С. 56-60.

2. Торба, Ю.І. Оптимізація конструкції факельного запальника ГТД чисельним методом [Текст] / Ю.І. Торба, Д.В. Павленко, Я.В. Двірник // Авиационно-космическая техника и технология. – Харьков, 2020. – №5(165). – С.83-95.

3. Что такое переобучение и как его избежать: основы [Электрон. ресурс]. – Режим доступа: https://habr.com/ru/companies/otus/articles/860958/.

4. Gern A. Hands On Machine Learning with Scikit Learn Keras and TensorFlow, Imprint: O'Reilly Media, 2019, 864 p.

5. Zou H., Hastie T. Regularization and Variable Selection Via the Elastic Net // Journal of the Royal Statistical Society Series B:Statistical Methodology, Volume 67, Issue 2, April 2005, P. 301–320 [Electronic resource]. – Access mode: https://academic.oup.com/jrsssb/article/67/2/301/7109482?login=false#400951354.

6. Müller A., Guido S., Introduction-to-Machine-Learning-with-Python [Electronic resource]. – Access mode: https://cs.hi.is/adhvarf/intro-to-machine-learning.pdf.

7. Lavasani A. Classic Machine Learning in Python: K-Nearest Neighbors (KNN) [Electronic resource]. – Access mode: https://medium.com/@amirm.lavasani/classic-machine-learning-in-python-k-nearest-neighbors-knn-a06fbfaaf80a.

8. Gaussian Naive Bayes [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/gaussian-naive-bayes/.

9. Support Vector Machine (SVM) Algorithm [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/support-vector-machine-algorithm/.

10. Kevin P. Murphy, Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series), Imprint: The MIT Press, 2012, 1068 p.

11. Hastie T., Tibshirani R., Friedman J., The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics), 2016.

12. Theobald O., Machine Learning For Absolute Beginners. [Electronic resource]. – Access mode: https://bmansoori.ir/book/Machine%20Learning%20For%20Absolute%20Beginners.pdf.

13. Ентропія та дерева прийняття рішень [Електрон. ресурс]. – Режим доступу: https://habr.com/ru/articles/171759/.

14. Методи дерев рішень, класифікації та прогнозування [Електрон. ресурс]. – Режим доступу: https://moodle.znu.edu.ua/pluginfile.php/486136/mod_resource/content/1/Лекція 9.pdf.

15. Geron A., Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, Imprint: O'Reilly Media, 2023, 850 p.

16. Random Forest Algorithm in Machine Learning [Electronic resource]. – Access mode: https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/.

17. Charu C. Aggarwal. Neural Networks And Deep Learning. [Electronic resource]. – Access mode: http://103.203.175.90:81/fdScript/RootOfEBooks/E%20Book%20collection%20-%202024%20-%20F/CSE%20%20IT%20AIDS%20ML/Neural%20Networks%20and%20Deep%20Learning_A%20Textbook-Springer%20(2023).pdf.

18. Інтелектуальні системи : навч. посіб. / С. О. Субботін, А. О. Олійник ; за ред. С. О. Субботіна. – Запоріжжя : ЗНТУ, 2014. – 219 с.

19. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с.

20. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес Паттерны объектно-ориентированного проєктирования / Пер. с англ.: А. Слинкин. – СПб.: Питер, 2021. – 448 с.

Додаток А

Приклад файлу протоколу вимірювань (згідно із мовою оригінала)

Дата испытания: 11.02.2012
Распорядительный документ: 450МС-1/2345
Шифр затрат: 123456
Конструктивные особенности:

Приспособление: АИ-1-1234-56-78 СБ
Воспламенитель: 1223346507-02 №011
Корпус :011
Пусковая форсунка: №4
Свеча зажигания: СЗ-12-34567

Измеряемые и расчетные параметры:

№реж	Время	H км	ΔPккж мм.вод.ст	ΔPп.т. кгс/см²	tвозд °C	tп.т. °C	tфак. °C	t x фак сек	tф2 °C	t x ф2 сек	tвосп °C	t x вос сек	E Дж	τ Гц	Горение	Циклограмма							Задерж. воспл, сек	Примечание
																a	b	c	d	tay	N	N1		
1	12:12:11	0.016	220	3.494	12	10	28	0	25	0	21	0	0	0	+	0	6	14	14	0,0	0,0	0,0	1,0	Горит
2	12:23:03	0.026	236	3.486	12	10	25	0	26	0	22	0	0	0	+	0	6	14	14	0,0	0,0	0,0	1,0	Горит
3	12:24:03	0.036	200	3.506	12	10	25	0	26	0	23	0	0	0	+	0	6	14	14	0,0	0,0	0,0	1,0	Горит
4	12:25:24	0.046	149	3.505	13	10	25	0	25	0	24	0	0	0	+	0	6	14	14	2,0	1,5	0,0	0,5	Горит
5	12:26:45	0.056	232	3.504	13	10	27	0	25	0	25	0	0	0	+	0	6	14	14	2,0	1,5	0,0	0,0	Горит
6	12:37:06	0.066	426	3.494	13	10	12	0	23	0	16	0	0	0	-	0	6	14	14	2,0	1,5	0,0	0,0	Не горит
7	12:38:27	0.076	411	3.478	13	11	12	0	25	0	17	0	0	0	-	0	6	14	14	2,0	1,5	0,0	0,0	Горит
8	12:39:48	0.086	455	3.500	13	11	14	0	26	0	18	0	0	0	-	0	6	14	14	0,0	0,0	0,0	0,0	Горит
9	12:30:06	0.096	499	3.496	13	11	12	0	24	0	19	0	0	0	-	0	6	14	14	0,0	0,0	0,0	0,0	Горит
10	12:41:23	0.006	322	3.489	13	11	13	0	27	0	10	0	0	0	-	0	6	14	14	0,0	0,0	0,0	0,0	Горит
11	12:42:46	0.016	342	3.489	13	11	14	0	27	0	11	0	0	0	-	0	6	14	14	0,0	0,0	0,0	0,0	Не горит
12	12:43:57	0.026	326	3.480	13	11	14	0	26	0	12	0	0	0	-	0	6	14	14	2,0	1,5	0,0	0,0	Не горит
13	12:44:13	0.036	386	3.472	13	11	14	0	26	0	13	0	0	0	-	0	6	14	14	2,0	1,5	0,0	0,0	Не горит
14	12:55:38	0.046	228	3.478	13	11	17	0	26	0	14	0	0	0	-	0	6	14	14	2,0	1,5	0,0	0,0	Не горит
15	12:56:49	0.056	212	3.477	13	11	15	0	27	0	15	0	0	0	-	0	6	14	14	2,0	1,5	0,0	0,0	Не горит
16	12:57:12	0.066	200	3.474	13	11	21	0	26	0	16	0	0	0	+/-	0	6	14	14	0,0	0,0	0,0	0,0	Разгорается в конце режима
17	12:58:32	0.076	226	3.504	13	11	18	0	25	0	17	0	0	0	+/-	0	6	14	14	0,0	0,0	0,0	1,0	Факел внутри воспламенителя
18	12:59:43	0.086	227	3.498	13	12	21	0	25	0	18	0	0	0	+	0	6	14	14	0,0	0,0	0,0	0,5	Горит
19	13:00:05	0.096	260	3.497	13	12	19	0	25	0	19	0	0	0	+/-	0	6	14	14	0,0	0,0	0,0	0,5	Факел внутри воспламенителя
20	13:01:24	0.096	242	3.497	13	12	17	0	25	0	10	0	0	0	+/-	0	6	14	14	2,0	1,5	0,0	0,5	Не горит
21	13:02:43	0.016	222	3.497	13	12	14	0	26	0	15	0	0	0	-	0	6	14	14	2,0	1,5	0,0	0,0	Не горит

a-время включение пусковой свечи, с; b-время включение отсечного клапана, с; c-время отключение пусковой свечи, с; d-время отключение отсечного клапана, с; tay- период срабатывания отсечного клапана при осуществлении пульсации пускового топлива, с; N- продолжительность, в период срабатывания отсечного клапана пускового топлива, во время которой клапан пускового топлива находится во включенном состоянии;E- энергия разряда на свече зажигания, Дж; τ- частота искрообразования на свече зажигания, Гц, N1 – первый импульс.

Додаток Б

Набіри тренувальних даних

Б.1 Набір тренувальних даних

#	square	altitude	delta_p_kk	delta_p_fuel	air_temp	fuel_temp	torch_temp	torch_time	ignition_temp	factor
1	45	0,06	98	4,565	-5	-34	63	2	85	1
...										

Б.2 Набір валідаційних даних

#	square	alttitude	delta_p_kk	delta_p_f	air_temp	fuel_temp	torch_temp	torch_time	ignition_temp	factor
1	45	0,06	195	4,576	-4	-28	4	0	0	0
...										

Додаток В

Текст програми

В.1 Текст файлу main.ipynb

```
# render dataset
from matplotlib import pyplot
import pandas
data_frame = pandas.DataFrame(ppl.raw_data)
plot_data_frame = data_frame[data_frame.columns]
_ = plot_data_frame.plot(subplots=True, figsize=(12, 12), legend=True)
pyplot.savefig('dataset.png')


# render heatmap
import seaborn
correlation_with_factor = data_frame.corr()['factor'].drop('factor')
correlation_df = correlation_with_factor.to_frame(name='Кореляція').reset_index()
correlation_df.rename(columns={'index': 'Параметри'}, inplace=True)
pyplot.figure(figsize=(10, 3))
ax = seaborn.heatmap(correlation_df.set_index('Параметри').T, annot=True,
                     cmap='coolwarm', fmt=".2f", cbar=True)
cbar = ax.collections[0].colorbar
cbar.set_label('Коеф. кореляції', rotation=90)
ax.set_yticklabels(ax.get_yticklabels(), rotation=0, fontsize=12)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right', fontsize=12)
pyplot.tight_layout()
pyplot.savefig('corelation-heat-map.png')


# render pairplot
data_frame['factor'] = data_frame['factor'].astype('category')
```

```
seaborn.pairplot(data_frame, vars=['delta_p_kk', 'torch_temp', 'ignition_temp'], hue='factor', diag_kind='kde', palette='coolwarm')
pyplot.savefig('pair-plot.png')


# print linear regression polinome
features = ppl.raw_data[0].keys()
linear_regression = ctx.strategies[0].model
print (linear_regression)
intercept = linear_regression.intercept_[0]  # Интерсепт
coefficients = linear_regression.coef_[0]
print("Полиномиальная модель:")
print(f"z = {intercept:.2f} " + " + ".join([f"{coeff:.2f}*{feature}" for coeff, feature in zip(coefficients, features)]))


# print Gaussian parameters
gaussian_naive = ctx.strategies[1].model
priors = gaussian_naive.class_prior_
theta = gaussian_naive.theta_
print (f'Априорні вероігідності класів: {priors[0]:.2f} - горить, {priors[1]:.2f} - не горить')
print ('Середні значення ознак класу - горить')
for coeff, feature in zip(gaussian_naive.theta_[0], features):
    print (f'{feature}: {coeff:.2f}')
print ('Средние значения ознак класу - не горить')
for coeff, feature in zip(gaussian_naive.theta_[1], features):
    print (f'{feature}: {coeff:.2f}')
```

```
# print metrics
result = dict()
for strategy in ctx.strategies:
    x = strategy.metrics
    row = {
        'accuracy': x['accuracy'],
        'precision': x['precision'],
        'recall': x['recall'],
        'f1': x['f1'],
        'roc_auc': x['roc_auc'],
        'mae': x['mae'],
    }
    if not x['overfit']:
        result[strategy.model_name] = (row)

import seaborn
import matplotlib.pyplot
import pandas
data_frame =pandas.DataFrame(result).T
df_melted = data_frame.reset_index().melt(id_vars='index', var_name='Metric', value_name='Value')
df_melted.rename(columns={'index': 'Model'}, inplace=True)

pyplot.figure(figsize=(12, 8))
seaborn.lineplot(data=df_melted, x='Model', y='Value', hue='Metric', marker='o', linewidth=2)

pyplot.ylabel('Значення', fontsize=14)
pyplot.xlabel('Модель', fontsize=14)
pyplot.xticks(rotation=45, ha='right', fontsize=12)
```

```
pyplot.ylim(0, 1.1)
pyplot.grid(axis='y', linestyle='--', alpha=0.7)
pyplot.legend(title='Метрика', fontsize=12, loc='upper left')
pyplot.tight_layout()
pyplot.savefig('mtrics-plot.png')
```

В.2 Текст файлу abstracts.py

```
import os
import re
import pandas
import pickle
import csv
import json
import numpy
from utils import NumpyEncoder

from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    roc_auc_score,
    confusion_matrix,
    mean_absolute_error,
)
from striprtf.striprtf import rtf_to_text
from sklearn.model_selection import GridSearchCV

TABLE_PATTERN = re.compile(r'\d{1,2}\|.*\|')


class Pipeline:
    """Base class for all process pipelines."""
    data_path = 'data'  # path to data folder
    train_data = []
    val_data = []

    @property
    def path(self):
        """
```

```
    Generator function recursively passes OS folder and yields path to
    data file.
    """
    file_name = 'protA3.rtf'
    for (dir_path, _, filenames) in os.walk(self.data_path):
        if file_name in filenames:
            yield os.path.join(dir_path, file_name)

@staticmethod
def process_rtf(rtf_content):
    content = rtf_to_text(rtf_content)
    # remove "\r" windows-specific char
    content = content.replace("\r", "")
    # remove whitespaces
    content = re.sub(r"[ \t\r]+", " ", content)
    # split string with newline char
    data = content.split("\n")
    data = [x.strip() for x in data]
    return list(filter(bool, data))

def read_rtf(self, path):
    with open(path, "r", encoding="cp1257") as fh:
        rtf_content = fh.read()
        return self.process_rtf(rtf_content)

@staticmethod
def safe_converter(value, function=float, default=0.0):
    try:
        return function(value)
    except ValueError:
        return default

def data_file_name(self, prefix):
    return os.path.join(self.data_path, f'{prefix}.csv')

def save_data(self, prefix, data):
    with open(self.data_file_name(prefix), 'w', newline='') as fh:
        writer = csv.DictWriter(fh, fieldnames=data[0].keys())
        writer.writeheader()
        for row in data:
            writer.writerow(row)

def restore_data(self, prefix):
    attr_name = f'{prefix}_data'
    attr = getattr(self, attr_name)
```

```
        with open(self.data_file_name(prefix), newline='') as fh:
            reader = csv.DictReader(fh)
            for row in reader:
                for key, value in row.items():
                    row[key] = numpy.float64(value)
                attr.append(row)


class BaseStrategy:
    model_class = None
    model_params = None
    optimal_params = None
    raw_model = None
    optimiser = None

    def __init__(self, train_data=None, val_data=None, model=None):
        super().__init__()
        if train_data and val_data:
            self.train_data = pandas.DataFrame(train_data)
            self.train_labels = self.train_data['factor']
            self.train_data = self.train_data.drop(columns=['factor'])
            self.val_data = pandas.DataFrame(val_data)
            self.val_labels = self.val_data['factor']
            self.val_data = self.val_data.drop(columns=['factor'])
        self.model = model

    def get_model(self, random_state):
        return self.model_class(random_state=random_state)

    def optimize(self, random_state=42, scoring='accuracy'):
        self.raw_model = self.get_model(random_state=random_state)
        self.optimiser = GridSearchCV(estimator=self.raw_model,
                                      param_grid=self.model_params,
                                      scoring=scoring,
                                      cv=5, verbose=1, n_jobs=-1)
        self.optimiser.fit(self.train_data, self.train_labels)
        self.optimal_params = self.optimiser.best_params_
        self.model = self.optimal_model
        self.estimate()

    @property
    def optimal_model(self):
        return self.optimiser.best_estimator_

    def predict(self, data):
```

```
        return self.model.predict(data)[0]

    def estimate(self):
        self.estimated = self.model.predict(self.val_data)
        self.estimated_probability = self.model.predict_proba(self.val_data)[:, 1]

    _metrics = None
    @property
    def metrics(self):
        if not self._metrics:
            train_estimated = self.model.predict(self.train_data)
            train_accuracy = accuracy_score(self.train_labels, train_estimated)
            val_estimated = self.model.predict(self.val_data)
            val_accuracy = accuracy_score(self.val_labels, val_estimated)
            self._metrics = {
                'accuracy': accuracy_score(self.val_labels, self.estimated),
                'precision': precision_score(self.val_labels, self.estimated),
                'recall': recall_score(self.val_labels, self.estimated),
                'f1': f1_score(self.val_labels, self.estimated),
                'roc_auc': roc_auc_score(self.val_labels, self.estimated_probability),
                'conf_matrix': confusion_matrix(self.val_labels, self.estimated),
                'mae': mean_absolute_error(self.val_labels, self.estimated),
                'train_accuracy': train_accuracy,
                'val_accuracy': val_accuracy,
                'overfit': (train_accuracy - val_accuracy) > 0.1,
            }
        return self._metrics

    def repr(self):
        metrics = self.metrics


        print(f'Accuracy: {metrics['accuracy']}')
        print(f'Precision: {metrics['precision']}')
        print(f'Recall: {metrics['recall']}')
        print(f'F1 Score: {metrics['f1']}')
        print(f'ROC AUC: {metrics['roc_auc']}')
        print(f'Mean Absolute Error (MAE): {metrics['mae']}')
        print(f'Confusion Matrix:\n{metrics['conf_matrix']}')
        if self.optimal_params:
            print(f'Best model params: {self.optimal_params}')


class StrategyIO:
    model_name = None
```

```
model_path = 'models'

@classmethod
def model_file_name(cls):
    return os.path.join(cls.model_path, f'{cls.model_name}.model.pickle')

@classmethod
def model_scaler_file_name(cls):
    return os.path.join(cls.model_path, f'{cls.model_name}.scaler.pickle')

@classmethod
def restore_model(cls):
    """Factory method returning model instance"""
    with open(cls.model_file_name(), 'rb') as fh:
        model = pickle.load(fh)
        instance = cls(model=model)
    # upload scaler if possible
    if os.path.isfile(cls.model_scaler_file_name()):
        with open(cls.model_scaler_file_name(), 'rb') as fh:
            instance.scaler = pickle.load(fh)

    if os.path.isfile(cls.model_metrics_file_name()):
        with open(cls.model_metrics_file_name(), 'rb') as fh:
            instance._metrics = json.load(fh)

    return instance

@classmethod
def model_param_file_name(cls):
    return os.path.join(cls.model_path, f'{cls.model_name}.param.json')

@classmethod
def model_metrics_file_name(cls):
    return os.path.join(cls.model_path, f'{cls.model_name}.metric.json')

def save_model(self):
    # save model itself
    with open(self.model_file_name(), 'wb') as fh:
        pickle.dump(self.model, fh)
    # save scaler
    if hasattr(self, 'scaler'):
        with open(self.model_scaler_file_name(), 'wb') as fh:
            pickle.dump(self.scaler, fh)
    # save model params
    with open(self.model_param_file_name(), 'w') as fh:
```

```
            json.dump(self.optimal_params, fh)
        # save model metrics
        with open(self.model_metrics_file_name(), 'w') as fh:
            json.dump(self.metrics, fh, cls=NumpyEncoder)


class Strategy(BaseStrategy, StrategyIO):
    pass


class ModelContext:
    strategies = []
    predicted = {}

    def __init__(self, classes, train_data, val_data):
        self.classes = classes
        self.train_data = train_data
        self.val_data = val_data

    def optimize(self):
        for cls in self.classes:
            strategy = cls(train_data=self.train_data, val_data=self.val_data)
            strategy.optimize()
            self.strategies.append(strategy)

    def save(self):
        for strategy in self.strategies:
            strategy.save_model()

    def restore(self):
        for cls in self.classes:
            instance = cls.restore_model()
            self.strategies.append(instance)

    def predict(self, data):
        for strategy in self.strategies:
            predicted = strategy.predict(data)
            self.predicted[strategy.model_name] = {
                'predicted': bool(predicted),
                'accuracy': strategy.metrics['accuracy'],
                'overfit': strategy.metrics['overfit']
            }
```

В.3 Текст файлу utils.py

```
import json
import numpy


class NumpyEncoder(json.JSONEncoder):
    def default(self, obj):
        if isinstance(obj, numpy.ndarray):
            return obj.tolist()
        return super(NumpyEncoder, self).default(obj)
```

В.4 Текст файлу constants.py

```
import re
from logistic_regression import LogisticRegression
from gaussian_classifier import GaussianNB
from knn_classifier import KNeighborsClassifier
from svm_classifier import SVMClassifier
from random_forest import RandomForestClassifier
from decision_tree import DecisionTreeClassifier
from mlp import MLPClassifier
from cnn import CNNClassifier


MODEL_CLASSES = [
    LogisticRegression,
    GaussianNB,
    KNeighborsClassifier,
    SVMClassifier,
    RandomForestClassifier,
    DecisionTreeClassifier,
    MLPClassifier,
    CNNClassifier,
]


DEFAULT_SQUARE = 45.0
SQUARE_MAP = {
    # ignore these 3 measurments as non-relevant
    # ['Режимы 5, 7, 14 – запуски воспламенителя 0140305000']
```

```
    # data/Воспл. 4500355000-01 № 108/2015_03_18 Воспл. 4500355000-01 № 108/protA3.rtf

    # В воздухоподводящем отверстии установлено 10 проволок: из', 'них 9 штук d=1,5 мм., и 1 штука d=1мм.
    'data/Воспл. 4500355000-01 № 095/2014_04_03 Воспл. 4500355000-01 № 095/1/protA3.rtf': DEFAULT_SQUARE - 9 * (3.14 * 1.5**2) / 4 + 1 * (3.14 * 1**2) / 4,
    # В воздухоподводящем отверстии установлено 9 проволок: из', 'них 8 штук d=1,5 мм., и 1 штука d=1мм.
    'data/Воспл. 4500355000-01 № 095/2014_04_03 Воспл. 4500355000-01 № 095/2/protA3.rtf': DEFAULT_SQUARE - 8 * (3.14 * 1.5**2) / 4 + 1 * (3.14 * 1**2) / 4,
    # В воздухоподводящем отверстии установлено 9 проволоктd=1,5', 'мм.
    'data/Воспл. 4500355000-01 № 095/2014_04_07 Воспл. 4500355000-01 № 095/protA3.rtf': DEFAULT_SQUARE - 9 * (3.14 * 1.5**2) / 4,
    # В воздухоподводящем отверстии установлено 6 проволоки', 'd=1,5 мм.
    'data/Воспл. 4500355000-01 № 095/2014_03_26 Воспл. 4500355000-01 № 095/1/protA3.rtf': DEFAULT_SQUARE - 6 * (3.14 * 1.5**2) / 4,
    # В воздухоподводящем отверстии установлено 7 проволоки', 'd=1,5 мм.
    'data/Воспл. 4500355000-01 № 095/2014_03_26 Воспл. 4500355000-01 № 095/2/26_03_14/protA3.rtf': DEFAULT_SQUARE - 7 * (3.14 * 1.5**2) / 4,
    # В воздухоподводящем отверстии установлено 8 проволоки', 'd=1,5 мм.
    'data/Воспл. 4500355000-01 № 095/2014_03_27 Воспл. 4500355000-01 № 095/protA3.rtf': DEFAULT_SQUARE - 8 * (3.14 * 1.5**2) / 4,
    # Ā āīēäóõīļīāāīä˙łåģ īņāåšńņčč óńņąķīāėåķī 9 ļšīāīėīź', 'd=1,5 ģģ.
    'data/Воспл. 4500355000-01 № 095/2014_03_27 Воспл. 4500355000-01 № 095/2/protA3.rtf': DEFAULT_SQUARE - 9 * (3.14 * 1.5**2) / 4,
    # В воздухоподводящем отверстии установлено 10 проволок', 'd=1,5 мм.
    'data/Воспл. 4500355000-01 № 095/2014_03_31 Воспл. 4500355000-01 № 095/protA3.rtf': DEFAULT_SQUARE - 10 * (3.14 * 1.5**2) / 4,
    # В воздухоподводящем отверстии установлено 5 проволоки', 'd=1,5 мм.
    'data/Воспл. 4500355000-01 № 095/2014_03_25 Воспл. 4500355000-01 № 095/protA3.rtf': DEFAULT_SQUARE - 5 * (3.14 * 1.5**2) / 4,
    # В воздухоподводящем отверстии установлено 3 проволоки', 'd=1,5 мм.
    'data/Воспл. 4500355000-01 № 095/2014_03_20 Воспл. 4500355000-01 № 095/1/protA3.rtf': DEFAULT_SQUARE - 3 * (3.14 * 1.5**2) / 4,
    # В воздухоподводящем отверстии установлено 4 проволоки', 'd=1,5 мм.
    'data/Воспл. 4500355000-01 № 095/2014_03_20 Воспл. 4500355000-01 № 095/2/protA3.rtf': DEFAULT_SQUARE - 4 * (3.14 * 1.5**2) / 4,
}
OK_FACTORS = [
    'Горит',
    'Маленький факел',
    'АИ-450М горит',
```

```
    'горит',
    'Ćīščņ',
]

TABLE_PATTERN = re.compile(r'\d{1,2}\|.*\|')

DROPLIST = [
    'Дата',
    'Распорядительный',
    'Шифр',
    'Конструктивные',
    'Воспламенитель',
    'Приспособление',
    'а-время',
    'Агрегат',
    'Корпус',
    'Пусковая',
    'Свеча',
    'Термопара tвоспл установлена',
    'Термопара tф установлена:',
    'Измеряемые и расчетные параметры:',
    '№реж|Время|Н',
    'км|DPккж',
    'мм.вод.ст|DPп.т.',
    'кгс/см2|tвозд',
    'С|tп.т.',
    'С|tфак.',
    'С|t х фак',
    'сек|tф2',
    'С|t х ф2',
    'сек|tвосп',
    'С|t х вос',
    'сек| Е',
    'Дж| t',
    'Гц|',
    'Горение|Циклограмма|Задерж.',
    'воспл, сек|Примечание|',
    '|||||||||||||| a| b| c| d|tay| N|N1|||',
    'сек|DP',
    'ккж1|tф2',
    '|||||||||||||| a| b| c| d|tay| N|N1|||',
    '№реж|Время|DPккж',
    '|||||||||||||| a| b| c| d|tay| N|N1|||',
]
```

В.5 Текст файлу logistic_regression.py

```
from abstracts import Strategy
from sklearn.linear_model import LogisticRegression as _LogisticRegression


class LogisticRegression(Strategy):
    model_class = _LogisticRegression
    model_name = 'Logistic Regression'
    model_params = {
        'penalty': ['l1', 'l2', 'elasticnet', None],  # regularisation types
        'C': [0.01, 0.1, 1, 10, 100],  # regularisation coefficient
        'solver': ['liblinear', 'saga', 'ridge', 'lbfgs'],  # optimisers
        'max_iter': [100, 500, 1000],  # max iterations
        'l1_ratio': [0.01, 0.5, 0.75, 1],
    }

    def repr(self):
        super().repr()
        print("Полиномиальна модель:")
        print(f"z = {intercept:.2f} " + " + ".join(
            [f"{coeff:.2f}*{feature}" for coeff, feature in zip(coefficients, features)])
        )
```

В.6 Текст файлу gaussian_classifier.py

```
from abstracts import Strategy
from sklearn.naive_bayes import GaussianNB as _GaussianNB


class GaussianNB(Strategy):
    model_class = _GaussianNB
    model_name = 'GaussianNB'
    model_params = {
        'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2,],
    }

    def get_model(self, random_state):
        return self.model_class()

    def repr(self):
        super().repr()
```

```
        priors = self.model.class_prior_
        theta = self.model.theta_
        print (f'Априорные вероятности классов: {priors[0]:.2f} - горить,
{priors[1]:.2f} - не горить')
        print ('Средние значения признаков для класса - горить')
        for coeff, feature in zip(self.model.theta_[0], features):
            print (coeff, feature)
        print ('Средние значения признаков для класса - не горить')
        for coeff, feature in zip(self.model.theta_[1], features):
            print (coeff, feature)
```

B.7 Текст файлу svm_classifier.py

```
from abstracts import Strategy
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC as _SVC


class SVMClassifier(Strategy):
    model_class = _SVC
    model_name = 'Support Vector Classifier'
    model_params = {
        'C': [0.1, 1, 10, 100],
        'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
        'gamma': ['scale', 'auto', 0.01, 0.1],
    }

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        if hasattr(self, 'train_data') and hasattr(self, 'val_data'):
            self.scaler = StandardScaler()
            self.train_data = self.scaler.fit_transform(self.train_data)
            self.val_data = self.scaler.transform(self.val_data)

    def get_model(self, random_state):
        return self.model_class(probability=True)

    def predict(self, data):
        """
        Method overrides parent`s class method. As long as KNN is
        need to be scaled, let`s do it here
        """
        scaled = self.scaler.transform(data)
```

```
        return super().predict(data=scaled)
```

В.8 Текст файлу random_forest.py

```
from abstracts import Strategy
from matplotlib import pyplot
from sklearn.ensemble import RandomForestClassifier as _RandomForestClassifier
from sklearn.tree import plot_tree
import numpy
import os


class RandomForestClassifier(Strategy):
    model_class = _RandomForestClassifier
    model_name = 'Random Forest'
    model_params = {
        'n_estimators': [100, 200, 500],
        'max_depth': [3, 5, 7],
        'min_samples_split': [2, 3, 4],
        'min_samples_leaf': [1, 2, 4],
        'max_features': [None, 'sqrt', 'log2'],
        'bootstrap': [True, False],
        'class_weight': ['balanced', 'balanced_subsample']
    }

    @property
    def optimal_model(self):
        """Pick the best tree from forest"""
        forest = super().optimal_model
        best_tree = None
        best_score = -numpy.inf
        for tree in forest:
            tree_score = tree.score(self.val_data, self.val_labels)
            if tree_score > best_score:
                best_score = tree_score
                best_tree = tree
        return best_tree

    @classmethod
    def model_image_file_name(cls):
        return os.path.join(cls.model_path, f'{cls.model_name}.tree.png')

    def save_model(self):
```

```
        super().save_model()
        feature_names = list(self.train_data.columns)
        pyplot.figure(figsize=(12, 10))
        plot_tree(self.model, filled=True, feature_names=feature_names,
class_names=['Не горить', 'Горить'], rounded=True)
        pyplot.savefig(self.model_image_file_name())

    def repr(self):
        super().repr()
        feature_names = list(self.train_data.columns)
        pyplot.figure(figsize=(12, 10))
        plot_tree(self.model, filled=True,
feature_names=feature_names,class_names=['Не горить', 'Горить'], rounded=True)
        pyplot.show()
```

B.9 Текст файлу decision_tree.py

```
import os
from abstracts import Strategy
from matplotlib import pyplot
from sklearn.tree import DecisionTreeClassifier as _DecisionTreeClassifier, plot_tree


class DecisionTreeClassifier(Strategy):
    model_class = _DecisionTreeClassifier
    model_name = 'Decision Tree'
    model_params = {
        'criterion': ['gini', 'entropy'],
        'max_depth': [3, 5, 7],
        'min_samples_split': [2, 3, 4],
        'min_samples_leaf': [1, 2, 4],
        'max_features': [None, 'sqrt', 'log2'],
    }
    @classmethod
    def model_image_file_name(cls):
        return os.path.join(cls.model_path, f'{cls.model_name}.tree.png')

    def save_model(self):
        super().save_model()
        feature_names = list(self.train_data.columns)
        pyplot.figure(figsize=(12, 10))
        plot_tree(self.model, filled=True, feature_names=feature_names,
class_names=['Не горить', 'Горить'], rounded=True)
```

```
        pyplot.savefig(self.model_image_file_name())

    def repr(self):
        super().repr()
        feature_names = list(self.train_data.columns)
        pyplot.figure(figsize=(12, 20))
        plot_tree(self.model, filled=True, feature_names=feature_names,
class_names=['Не горить', 'Горить'], rounded=True)
        pyplot.show()
```

В.10 Текст файлу mlp.py

```
from abstracts import Strategy
from sklearn.neural_network import MLPClassifier as _MLPClassifier


class MLPClassifier(Strategy):
    model_class = _MLPClassifier
    model_name = 'Multi Layer Perceptron'
    model_params = {
        'hidden_layer_sizes': [
            (8, 8),
            (8, 16),
            (8, 16),
            (16, 32),
            (16),
            (16, 16),
            (16, 32),
            (64),
            (32, 64),
        ],
        'activation': ['relu', 'tanh'],
        'solver': ['adam', 'sgd'],
        'alpha': [0.0001, 0.001, 0.01],
        'learning_rate': ['constant', 'adaptive'],
        'batch_size': [32, 64],
    }
```

В.11 Текст файлу cnn.py

```
from abstracts import Strategy
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout, Input
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    roc_auc_score,
    confusion_matrix,
    mean_absolute_error,
)
import numpy

class CNNClassifier(Strategy):
    model_name = 'Convolutional Neural Network'
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        if hasattr(self, 'train_data') and hasattr(self, 'val_data'):
            self.scaler = StandardScaler()
            self.train_data = self.scaler.fit_transform(self.train_data)
            self.train_data = self.train_data.reshape(self.train_data.shape[0], 1, 9, 1)
            self.val_data = self.scaler.transform(self.val_data)
            self.val_data = self.val_data.reshape(self.val_data.shape[0], 1, 9, 1)

        self.model = Sequential([
            Input([1, 9, 1]),
            Conv2D(32, (1, 3), activation='relu'),
            MaxPooling2D(pool_size=(1, 2), padding='same'),
            Dropout(0.25),
            Conv2D(64, (1, 3), activation='relu'),
            MaxPooling2D(pool_size=(1, 2), padding='same'),
            Dropout(0.25),
            Flatten(),
            Dense(16, activation='relu'),
            Dropout(0.5),
            Dense(1, activation='sigmoid')
        ])
        self.model.compile(optimizer='adam',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])

    def optimize(self, random_state=42, scoring='accuracy'):
        self.estimate()
```

```
    def fit(self):
        return self.model.fit(self.train_data, self.train_labels)

    def estimate(self):
        self.estimated_probability = self.model.predict(self.val_data)
        self.estimated = (self.estimated_probability > 0.5).astype("int32")

    def predict(self, data):
        scaled = self.scaler.transform(data)
        scaled = numpy.array(scaled)
        scaled = scaled.reshape(scaled.shape[0], 1, 9, 1)
        return super().predict(data=scaled)

    _metrics = None
    @property
    def metrics(self):
        if not self._metrics:
            train_estimated = (self.model.predict(self.train_data) > 0.5).astype("int32")
            train_accuracy = accuracy_score(self.train_labels, train_estimated)
            val_estimated = (self.model.predict(self.val_data) > 0.5).astype("int32")
            val_accuracy = accuracy_score(self.val_labels, val_estimated)
            self._metrics = {
                'accuracy': accuracy_score(self.val_labels, self.estimated),
                'precision': precision_score(self.val_labels, self.estimated),
                'recall': recall_score(self.val_labels, self.estimated),
                'f1': f1_score(self.val_labels, self.estimated),
                'roc_auc': roc_auc_score(self.val_labels, self.estimated_probability),
                'conf_matrix': confusion_matrix(self.val_labels, self.estimated),
                'mae': mean_absolute_error(self.val_labels, self.estimated),
                'train_accuracy': train_accuracy,
                'val_accuracy': val_accuracy,
                'overfit': (train_accuracy - val_accuracy) > 0.1,
            }
        return self._metrics
```

В.12 Текст файлу pipeline_clean.py

```
import re

from abstracts import Pipeline
from constants import DROPLIST, TABLE_PATTERN
```

```
class CleanPipeline(Pipeline):
    def process(self):
        """The method iterates over the data folder and finds the differences between
files."""
        self.maping = dict()
        for path in self.path:
            data = self.clean_file(path)
            data = list(filter(lambda x: x != '', data))
            if data:
                self.maping[path] = ','.join(data)

    def clean_file(self, path):
        data = self.read_rtf(path)
        for lineno, line in enumerate(data):
            line = line.strip()
            for drop_item in DROPLIST:
                if line.startswith(drop_item):
                    data[lineno] = ''
                    break
            if re.match(TABLE_PATTERN, line):
                data[lineno] = ''
        return data
```

В.13 Текст файлу pipeline_data.py

```
from abstracts import Pipeline, TABLE_PATTERN
import re
from copy import deepcopy
import random
from constants import DEFAULT_SQUARE, SQUARE_MAP, OK_FACTORS


class DataPipeline(Pipeline):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.raw_data = []

    def load(self):
        for protocol in self.protocols:
            for line in self.clean(protocol):
                self.raw_data.append(line)

    ALGO_RANDOM = 'random'
```

```
ALGO_THRESHOLD = 'threshold'
def split(self, algo=ALGO_RANDOM, ratio=0.8, threshold=5):
    dataset = deepcopy(self.raw_data)
    if algo == self.ALGO_RANDOM:
        random.shuffle(dataset)
        limit = int(len(dataset) * ratio)
        train_data = dataset[:limit]
        val_data = dataset[limit:]
    elif algo == self.ALGO_THRESHOLD:
        train_data = []
        val_data = []
        for index, value in enumerate(dataset):
            if index % threshold != 0:
                train_data.append(value)
            else:
                val_data.append(value)
    else:
        raise NotImplementedError('Unknown data split algorithm')
    self.train_data = train_data
    self.val_data = val_data

def clean(self, protocol):
    square = DEFAULT_SQUARE
    path = protocol['path']
    if path in SQUARE_MAP:
        square = SQUARE_MAP[path]
    initial = {
        'square': square,
        # 'path': path,
        # 'device_type': protocol['device_type'],
        # 'igniter': protocol['igniter'],
    }
    for item in protocol['data']:
        data = deepcopy(initial)
        data['altitude'] = self.safe_converter(item[2])
        data['delta_p_kk'] = self.safe_converter(item[3])
        data['delta_p_fuel'] = self.safe_converter(item[4])
        data['air_temp'] = self.safe_converter(item[5])
        data['fuel_temp'] = self.safe_converter(item[6])
        data['torch_temp'] = self.safe_converter(item[7])
        data['torch_time'] = self.safe_converter(item[8])
        data['ignition_temp'] = self.safe_converter(item[11])
        factor = 0
        if item[-1] in OK_FACTORS:
            factor = 1
```

```
        # data['factor'] = factor
        data['factor'] = factor
        yield data

@property
def protocols(self):
    for path in self.path:
        data = self.read_rtf(path)
        protocol = self.parse_protocol(data)
        protocol['path'] = path
        yield protocol

def parse_protocol(self, data):
    def finder(feature, data=data):
        for line in data:
            line = line.strip()
            if feature(line):
                return line.split(':')[1].strip()

    response = {}
    response['date'] = finder(lambda x: x.startswith('Дата'))
    response['device_type'] = finder(lambda x: x.startswith('Приспособление'))
    response['igniter'] = finder(lambda x: x.startswith('Воспламенитель'))
    response['body'] = finder(lambda x: x.startswith('Корпус'))
    response['nozzle'] = finder(lambda x: x.startswith('Пусковая форсунка'))
    response['spark'] = finder(lambda x: x.startswith('Свеча зажигания'))
    response['ignitor-unit'] = finder(lambda x: x.startswith('Агрегат зажигания'))
    response['ignitor-place'] = finder(lambda x: 'твоспл' in x)
    response['torch-place'] = finder(lambda x: 'тф' in x)
    response['square'] = DEFAULT_SQUARE
    response['data'] = []
    for line in data:
        if re.match(TABLE_PATTERN, line):
            row = line.split('|')[:-1]
            response["data"].append(row)
    return response

def save(self):
    self.save_data('train', self.train_data)
    self.save_data('val', self.val_data)

def restore(self):
    self.restore_data('train')
    self.restore_data('val')
```

В.14 Текст файлу update_data.py

```
import click
from pipeline_data import DataPipeline


@click.command()
@click.option('--algo', help=("""Алгоритм розподілення даних для """
              """тренуваня моделей (за-замовченнямкожен 5ий рядок [threshold]), можна """
              """вказати ще випадковий підбір [random]"""), default=DataPipeline.ALGO_THRESHOLD)

def update(algo):
    """
    Ця програма призначена для оновлення даних. Якщо ви додали новий протокол
    вимірювань та бажаете додати його до розрахунків моделей, виконайте цю
    программу.

    Для цього вона
        * рекурсивно знаходить усі файли у діректорії "./data" та знаходить файли
          протоколів вимірювань за ім'ям "protA3.rtf"
        * аналізує кожен файл та читає таблицю виміряних значень
        * після цього вона зберігає усі отримані дані у вигляді '*.csv'
    """
    ppl = DataPipeline()
    ppl.load()
    ppl.split(algo=algo)
    ppl.save()


if __name__ == '__main__':
    update()
```

В.15 Текст файлу optimize_models.py

```
import click
from .abstracts import ModelContext
from constants import MODEL_CLASSES
from .pipeline_data import DataPipeline
```

```
@click.command()
def optimize():
    """
    Ця програма призначена для оновлення математичних моделей.
    """
    ppl = DataPipeline()
    ppl.restore()
    ctx = ModelContext(classes=MODEL_CLASSES, train_data=ppl.train_data,
val_data=ppl.val_data)
    ctx.optimize()
    ctx.save()

if __name__ == '__main__':
    optimize()
```

В.16 Текст файлу estimate_data.py

```
import click
import numpy as numpy

from abstracts import ModelContext
from pipeline_data import DataPipeline
from constants import MODEL_CLASSES
from colored import Back, Style
import tabulate

@click.command()
@click.option('--square', help='Площа отвору запальника (поле: square)',
type=numpy.float64, required=True)
@click.option('--altitude', help='Висота польоту (поле: altitude)',
type=numpy.float64, required=True)
@click.option('--delta-p-kk', help='Перепад тиску у кільцевому каналі (поле:
delta_P_kk)', type=numpy.float64, required=True)
@click.option('--delta-p-fuel', help='Перепад тиску палива (поле: delta_P_fuel)',
type=numpy.float64, required=True)
@click.option('--air-temp', help='Температура повітря (поле: air_temp)',
type=numpy.float64, required=True)
@click.option('--fuel-temp', help='Температура палива (поле: fuel_temp)',
type=numpy.float64, required=True)
@click.option('--torch-temp', help='Температура горіння полум’я (поле:
torch_temp)', type=numpy.float64, required=True)
```

```
@click.option('--torch-time', help='Час горіння полум’я (поле: torch_time)',
type=numpy.float64, required=True)
@click.option('--ignition-temp', help='Температура займання полум’я (поле: igni-
tion_temp)', type=numpy.float64, required=True)
def optimize(square, altitude, delta_p_kk, delta_p_fuel, air_temp, fuel_temp,
             torch_temp, torch_time, ignition_temp):
    """
    Ця програма призначена для передбачення даних математичних моделей.
    Приклад використання
    python estimate_data.py --square=32 --altitude=0.0 --delta-p-kk=95 --delta-p-
fuel=3.5 --air-temp=2 --fuel-temp=-34.0 --torch-temp=64 --torch-time=2 --ignition-
temp=85
    """
    ppl = DataPipeline()
    ppl.restore()
    ctx = ModelContext(classes=MODEL_CLASSES, train_data=ppl.train_data,
val_data=ppl.val_data)
    ctx.restore()
    raw_data = {
        'square': square,
        'altitude': altitude,
        'delta_p_kk': delta_p_kk,
        'delta_p_fuel': delta_p_fuel,
        'air_temp': air_temp,
        'fuel_temp': fuel_temp,
        'torch_temp': torch_temp,
        'torch_time': torch_time,
        'ignition_temp': ignition_temp,
    }
    data = [numpy.array(list(raw_data.values()))]
    ctx.predict(data)
    output = []
    factor_positive = factor_negative = 0
    for model_name, value in ctx.predicted.items():
        prediction = value['predicted']
        accuracy = value['accuracy']
        overfit = value['overfit']
        accuracy_color = Back.GREEN
        if accuracy <= 0.8 or overfit:
            prediction_color = accuracy_color = Back.GREY_37
        else:
            if prediction:
                prediction_color = Back.GREEN
                factor_positive += 1
            else:
```

```
            prediction_color = Back.red
            factor_negative += 1
        model_message = f'{accuracy_color}{model_name}{Style.reset}'
        prediction_message = f'{prediction_color}Горить{Style.reset}'
        if not prediction:
            prediction_message = f'{prediction_color}Не горить{Style.reset}'
        accuracy_message = f'{accuracy_color}{accuracy}{Style.reset}'
        overfit_message = f'Ні'
        if overfit:
            overfit_message = f'Так'
        output.append([model_message, prediction_message, accuracy_message, over-
fit_message])
    print(tabulate.tabulate(output, headers=['Модель', 'Передбачення', 'Точність
моделі', 'Перенавчена'], tablefmt="fancy_grid"))
    factor_num = factor_positive + factor_negative
    print (f'Таким чином, із {factor_num} точних моделей {factor_positive}
передбачило позитивний результат, а {factor_negative} - негативний')
    return ctx.predicted

if __name__ == '__main__':
    optimize()
```

Додаток Г

Слайди презентації

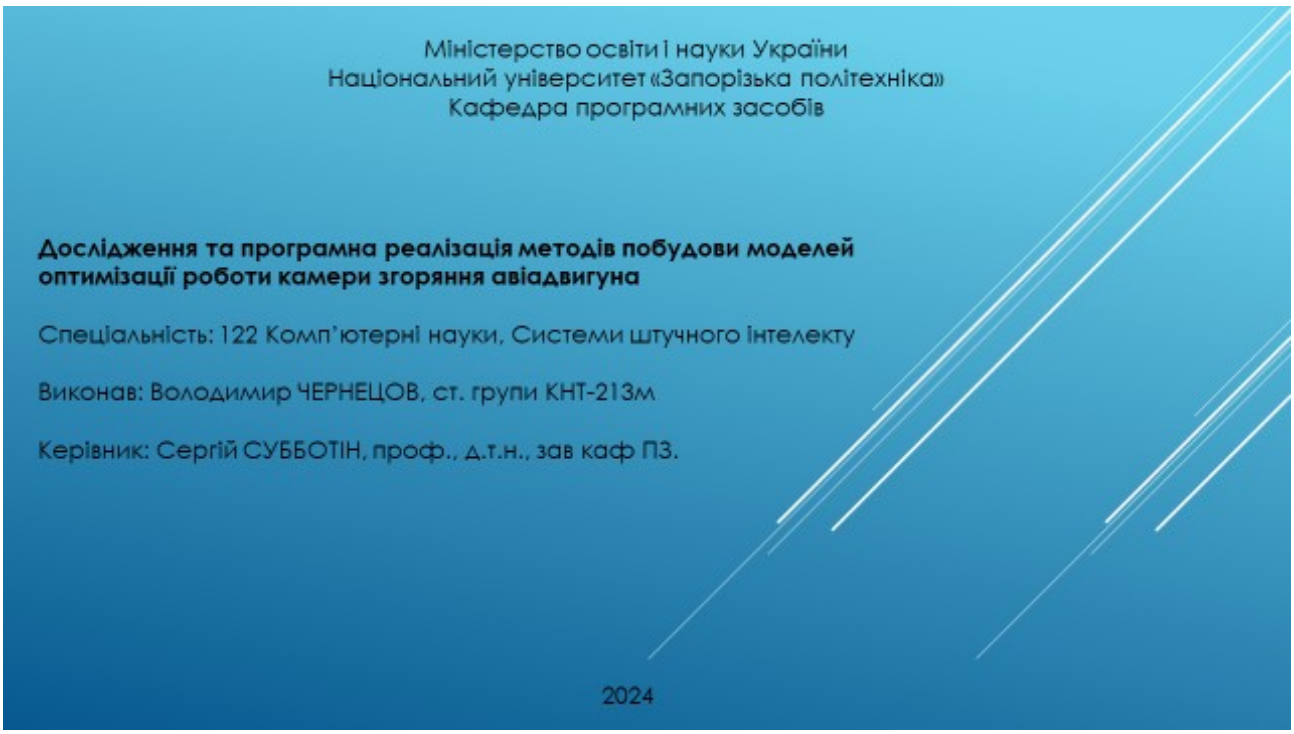


Рисунок Г.1 – Слайд №1

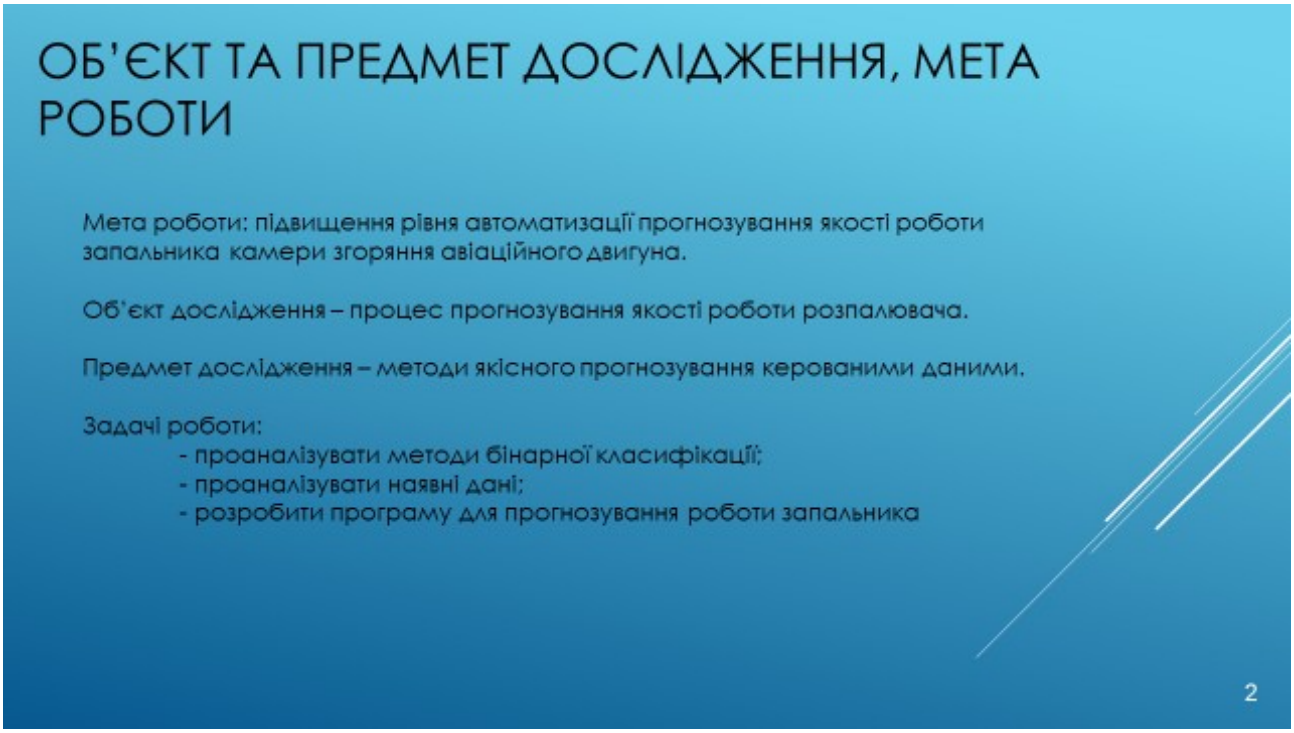


Рисунок Г.2 – Слайд №2

Рисунок Г.3 – Слайд №3

Критерії порівняння	Мови програмування			
	Python	Mathlab	R	Octave
Відкрите ПЗ	+	-	+	+
Багатоплатформність	+++	++	++	++
Вбудовані математичні операції	+	+++	++	+++
Наявні математичні бібліотки	+	+	+	+
Наявні нейромережеві бібліотекі	+	+	+	+
Легкість інтеграції (веб/GUI/консоль)	+++	+	+	+
Підтримка модульності	+++	+	++	+
Підтримка парадігми ООП	+++	+	+	+
Спільнота та документація	+++	+	++	+

Рисунок Г.4 – Слайд №4

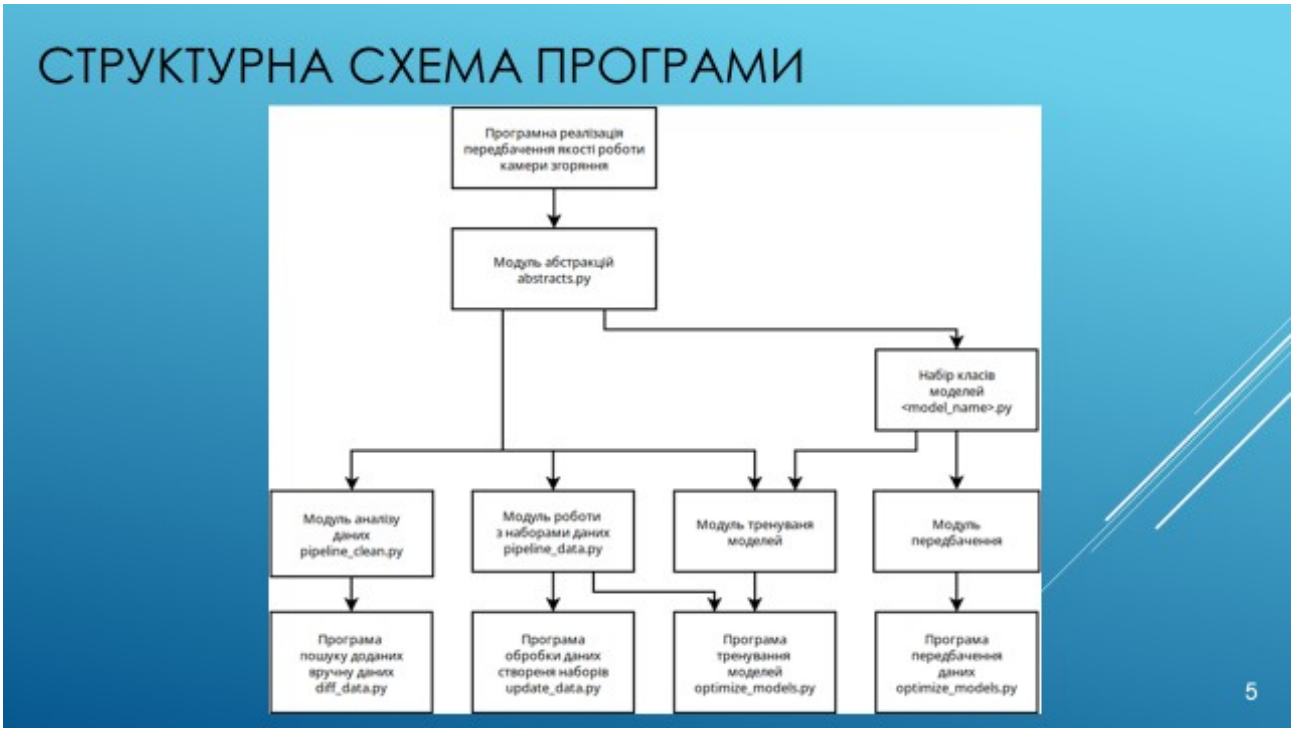


Рисунок Г.5 – Слайд №5

Рисунок Г.6 – Слайд №6

Рисунок Г.7 – Слайд №7

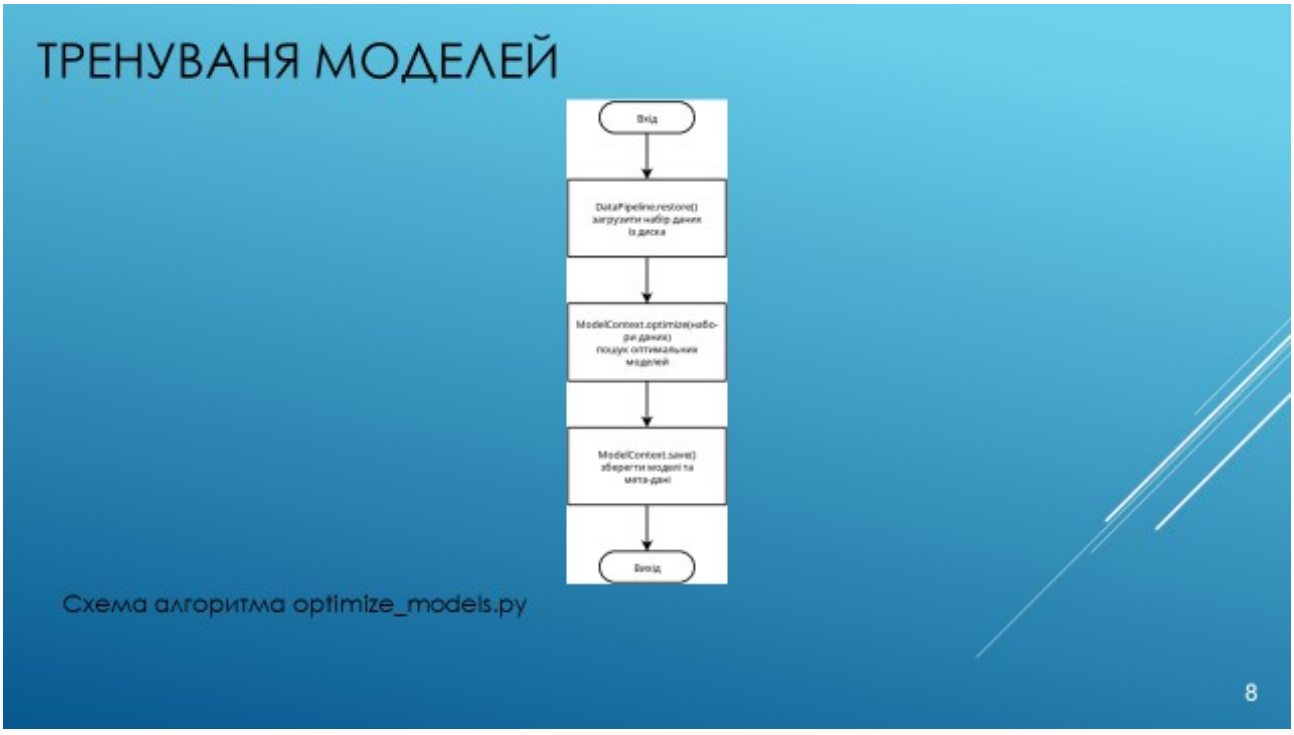


Рисунок Г.8 – Слайд №8

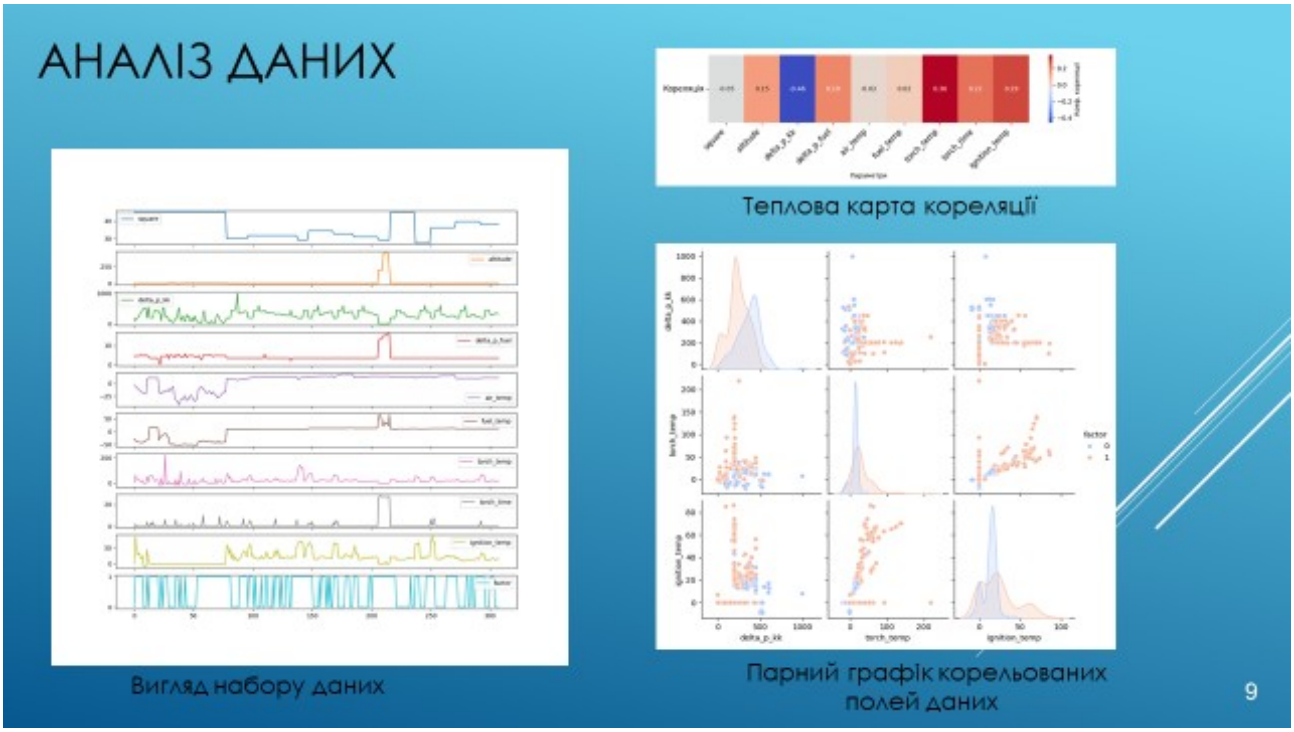


Рисунок Г.9 – Слайд №9

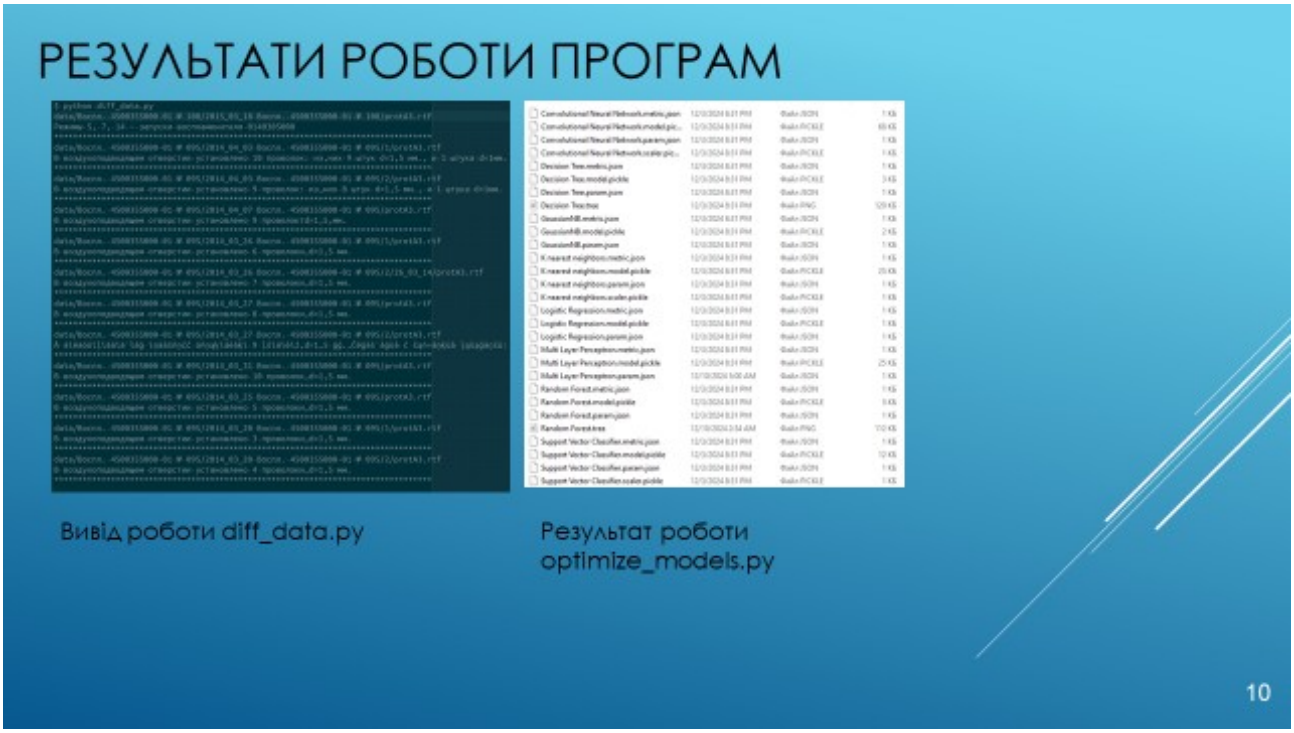


Рисунок Г.10 – Слайд №10

РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ ПЕРЕДБАЧЕННЯ ДАНИХ

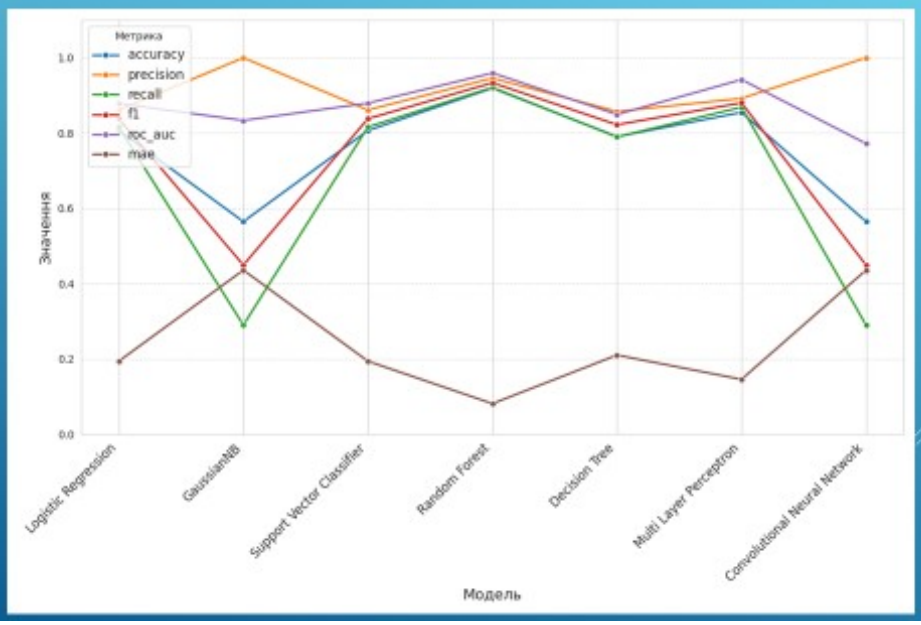
Модель	Передбачення	Точність моделі	Перенавчена
Logistic Regression	Горить	0.806452	Ні
GaussianNB	Горить	0.564516	Ні
K nearest neighbors	Горить	0.83871	Так
Support Vector Classifier	Горить	0.806452	Ні
Random Forest	Не горить	0.919355	Ні
Decision Tree	Горить	0.790323	Ні
Multi Layer Perceptron	Горить	0.854839	Ні
Convolutional Neural Network	Горить	0.564516	Ні

Таким чином, із 4 точних моделей 3 передбачило позитивний результат, а 1 - негативний

11

Рисунок Г.11 – Слайд №11

МЕТРИКИ МОДЕЛЕЙ



12

Рисунок Г.12 – Слайд №12

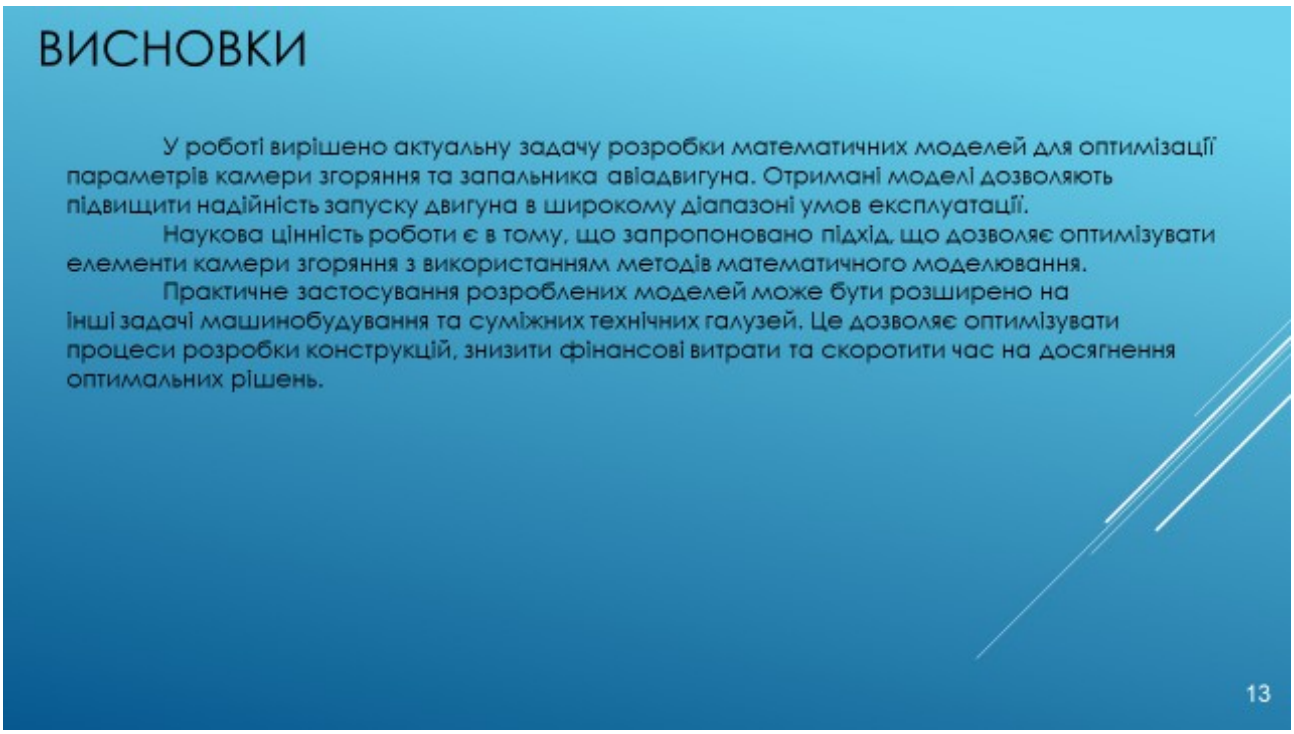


Рисунок Г.13 – Слайд №13