

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему КОМП'ЮТЕРНА СИСТЕМА ДЛЯ УПРАВЛІННЯ ВИТРАТАМИ
НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Виконав: студент 2 курсу, групи КНТ-622м
спеціальності

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Спеціалізовані комп'ютерні системи

ЛЕВЧЕНКО А.М.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент САМОЙЛИК С.С.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“24” жовтня 2023 року

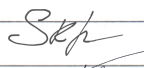
З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ЛЕВЧЕНКА Андрія Михайловича

(ПРІЗВИЩЕ, ім'я, по батькові)

- Тема проєкту (роботи) Комп'ютерна система для управління витратами на основі штучного інтелекту
керівник проєкту (роботи) к. т. н., доцент, СКРУПСЬКИЙ Степан Юрійович
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400
- Строк подання студентом проєкту (роботи) 10 грудня 2023 року
- Вихідні дані до проєкту (роботи) наявні методи та алгоритми розробки комп'ютерної системи на основі штучного інтелекту
- Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз проблеми та засобів її вирішення;
2) Розробка комп'ютерної системи;
3) Дослідження комп'ютерної системи.
- Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	СКРУПСЬКИЙ С.Ю., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз проблеми та засобів її вирішення	10.10.2023 р.	
2	Розробка структури системи	15.10.2023 р.	
3	Розробка алгоритму роботи системи	20.10.2023 р.	
4	Реалізація комп'ютерної системи	05.11.2023 р.	
5	Дослідження комп'ютерної системи	20.11.2023 р.	
6	Оформлення отриманих результатів у ПЗ	25.11.2023 р.	
7	Оформлення графічного матеріалу	01.12.2023 р.	
8	Оформлення допоміжного матеріалу	10.12.2023 р.	
9	Захист проєкту (роботи)	13.12.2023 р.	

Студент



Андрій ЛЕВЧЕНКО

(ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)



Степан СКРУПСЬКИЙ

(підпис)

(ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 73 с., 25 рис., 1 табл., 11 ліст., 56 джерел.

ANDROID, DART, FLUTTER, IOS, LSTM, PYTHON, TENSORFLOW LITE,
КОМП'ЮТЕРНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, УПРАВЛІННЯ
ВИТРАТАМИ, ШТУЧНИЙ ІНТЕЛЕКТ

Об'єкт дослідження – розробка комп'ютерної системи для управління витратами на основі штучного інтелекту.

Предметом дослідження є методи та засоби розробки мобільних застосунків з використанням фреймворку Flutter та моделей на основі штучного інтелекту за допомогою бібліотеки TensorFlow Lite.

Метою проєкту (роботи) є розробка та дослідження комп'ютерної системи для управління персональними витратами на мобільних пристроях.

Матеріали, методи та технічні засоби: мова програмування Dart, фреймворк Flutter, база даних SQLite, інтегроване середовище розробки Android Studio, мова програмування Python, бібліотека TensorFlow Lite, рекурентна нейронна мережа LSTM.

Висновки. Наукова новизна даного проєкту (роботи) є у розробці та тренуванні власної моделі на основі штучного інтелекту з використанням LSTM, що може бути розміщена на мобільних пристроях.

Результати. Практична цінність даного проєкту (роботи) полягає у реалізації комп'ютерної системи на основі моделі, яка дозволяє покращити конфіденційність, швидкодію на 8,7%, не вимагає підключення до мережі Інтернет та адаптує модель на основі штучного інтелекту до даних користувача у порівнянні з аналогами.

Галузь використання – індивідуальне управління витратами.

ABSTRACT

Explanatory note to the master's work: 73 p., 25 figures, 1 table, 11 listings, 56 sources.

ANDROID, DART, FLUTTER, IOS, LSTM, PYTHON, TENSORFLOW LITE, COMPUTER SYSTEM, MACHINE LEARNING, EXPENSE MANAGEMENT, ARTIFICIAL INTELLIGENCE

Object research – development of a computer system for expense management based on artificial intelligence.

The subject of research is methods and means of developing mobile applications using the Flutter framework and models based on artificial intelligence using the TensorFlow Lite library.

The purpose of the work is the development and research of a computer system for managing personal expenses on mobile devices.

Materials, Methods, and Techniques: Dart programming language, Flutter framework, SQLite database, Android Studio integrated development environment, Python programming language, TensorFlow Lite library, LSTM recurrent neural network.

Conclusions. The scientific novelty of this project is in the development and training of its own model based on artificial intelligence using LSTM, which can be placed on mobile devices.

The results. The practical value of this project consists in the implementation of a computer system based on the model, which allows for improved confidentiality, performance by 8.7%, does not require a connection to the Internet, and adapts the model based on artificial intelligence to user data in comparison with analogues.

The field of use is individual expense management.

ЗМІСТ

Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз проблеми та засобів її вирішення.....	10
1.1 Поняття управління витратами	10
1.2 Визначення завдання.....	11
1.3 Види розробки мобільних застосунків.....	12
1.3.1 Розробка на Android	12
1.3.2 Розробка на iOS	13
1.3.3 Кросплатформна розробка	14
1.4 Аналіз мов програмування для розробки мобільних застосунків.....	16
1.4.1 Java.....	16
1.4.2 Kotlin.....	17
1.4.3 Objective-C	18
1.4.4 Swift	19
1.4.5 Dart.....	20
1.5 Аналіз фреймворків для розробки мобільних застосунків	21
1.5.1 Xamarin	21
1.5.2 React Native	22
1.5.3 Flutter	22
1.6 Аналіз фреймворків штучного інтелекту для Flutter.....	24
1.6.1 Firebase MLKit	24
1.6.2 PyTorch Mobile.....	25
1.6.3 TensorFlow Lite	25
1.7 Аналіз мови програмування Python для ML.....	26
1.8 Методи машинного навчання	27
1.8.1 Лінійна регресія.....	27
1.8.2 Метод опорних векторів	29

1.8.3 Дерево рішень	30
1.8.4 Випадковий ліс	31
1.8.5 Рекурентна нейронна мережа.....	32
1.8.6 Довга короткочасна пам'ять	34
1.8.7 Алгоритм k-найближчих сусідів.....	35
1.9 Висновки за розділом 1.....	37
2 Розробка комп'ютерної системи.....	37
2.1 Розробка мобільного застосунку	37
2.1.1 Впровадження архітектурного шаблону.....	38
2.1.2 Розробка бази даних.....	39
2.1.3 Розробка віджетів представлення.....	45
2.2 Розробка моделі на основі штучного інтелекту	51
2.2.1 Тренування моделі	52
2.2.2 Інтеграція моделі у систему	54
2.3 Висновки за розділом 2.....	55
3 Дослідження комп'ютерної системи.....	56
3.1 Дослідження моделі на основі штучного інтелекту	56
3.2 Випробування комп'ютерної системи	63
3.3 Висновки за розділом 3.....	67
Висновки	68
Перелік джерел посилання	69

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

AI – Artificial Intelligence;
API – Application Programming Interface;
ASM – Attribute Selection Measure;
CV – Computer Vision;
IDE – Integrated Development Environment;
iOS – iPhone Operating System;
JVM – Java Virtual Machine;
KNN – K-Nearest Neighbors;
LLVM – Low Level Virtual Machine;
LSTM – Long Short Term Memory;
ML – Machine Learning;
MVC – Model-View-Controller;
NLP – Natural Language Processing;
RNN – Recurrent Neural Network;
SDK – Software Development Kit;
SQL – Structured Query Language;
SVM – Support Vector Machine;
VM – Virtual Machine;
WORA – write once, run anywhere;
XNU – X is Not Unix.

ВСТУП

Після виникнення перших спільнот людей з'явилася необхідність в обміні матеріальними речами. Людина могла щось виготовляти чи мати більше ніж їй необхідно та при цьому потребувала щось інше. Бартер став першою формою обміну. Його сутність складалась з обміну певної кількості чогось на певну кількість чогось іншого. Через це виникала проблема, що якщо у людей нема необхідних для одне одного речей, то обмін неможливий. Цю проблему можна вирішити з застосуванням універсальної міри обміну. З часом така міра обміну була зроблена й отримала назву гроші.

Сьогодні існує безліч різних національних валют, що признаються світом. Також популярність набирають альтернативні валюти такі як криптовалюта. Гроші для людей це можливість купити товар чи послугу, що зробить їх життя краще, тому люди намагаються збільшити своє багатство. Для цього існують два шляхи – оптимізація витрат та інвестиція. Перший варіант ефективний та не потребує особливих передумов, але він має обмеження. Другий варіант цих обмежень не має, але має ризики. Для розв'язання цих проблем можливе застосування комп'ютерної системи управління витратами на основі штучного інтелекту, що може з деяким успіхом прогнозувати майбутні витрати чи ризики при інвестиції.

В даному магістерському проєкті (роботі) буде розглянута розробка необхідної комп'ютерної системи для розв'язання поставлених задач. Буде обрахована практична цінність у порівнянні з аналогами по декількох критеріям, а також представлена наукова новизна, що складається з розробки моделі на основі штучного інтелекту під вимоги даної системи.

1 АНАЛІЗ ПРОБЛЕМИ ТА ЗАСОБІВ ЇЇ ВИРІШЕННЯ

1.1 Поняття управління витратами

Управління витратами – це процес систематичного контролю, планування та оптимізації грошових витрат з метою досягнення оптимального використання коштів [1].

У більш широкому сенсі управління витратами також містить в собі введення контролю доходів. Тому систематизація та аналіз необхідні для всіх видів транзакцій.

Існують наступні види витрат:

- витрати першої необхідності, які не можуть бути скорочені (витрати на їжу, воду, одяг тощо);
- дискреційні витрати, які витрачаються на власний розсуд людини (витрати на хобі та предмети розкоші);
- постійні витрати, які є запланованими та мають заздалегідь визначену фіксовану суму (витрати на інтернет та мобільні поповнення);
- змінні витрати, які важко визначити заздалегідь (витрати на пальне та медикаменти);
- поточні витрати, які повторюються з місяця в місяць (витрати на оренду та комунальні послуги);
- одноразові витрати, які трапляються нечасто та мають незапланований характер (витрати на лікування, ремонт будинку).

Загалом для того, щоб оптимізувати свої витрати необхідно вести систематичний контроль за своїми транзакціями, періодично вести аналіз над цими даними та робити з цього висновки. Наприклад, зменшити кількість непотрібних витрат чи збільшити свій дохід.

1.2 Визначення завдання

Завданням дипломного проєкту (роботи) є дослідження та розробка комп'ютерної системи для управління витратами на основі штучного інтелекту. Система повинна розміщатись на мобільних пристроях, приймати, зберігати та відображати дані про транзакції й прогнозувати витрати за допомогою моделі на основі штучного інтелекту.

З метою досягнення поставленої цілі, перед розробником виникає необхідність вирішити наступні завдання:

- аналіз видів розробки мобільних застосунків;
- аналіз наявних мов програмування для розробки мобільних застосунків;
- аналіз наявних фреймворків для розробки мобільних застосунків;
- аналіз наявних фреймворків штучного інтелекту для мобільних застосунків;
- аналіз обраної мови програмування для розробки моделі на основі штучного інтелекту;
- аналіз методів машинного навчання;
- розробка мобільного застосунку;
- впровадження архітектурного шаблону;
- підключення системи керування базами даних;
- розробка віджетів представлення;
- розробка моделі на основі штучного інтелекту;
- інтеграція моделі у мобільний застосунок;
- дослідження комп'ютерної системи;
- дослідження моделі на основі штучного інтелекту;
- випробування комп'ютерної системи.

1.3 Види розробки мобільних застосунків

Найпопулярнішими операційними системами для мобільних пристроїв є Android та iOS (iPhone Operating System). Станом на жовтень 2023 року частка Android на ринку мобільних операційних систем у всьому світі складає 69,66%, тоді як операційна система iOS складає 29,65% цього ринку. Загалом Android і iOS зараз складають 99,31% світового ринку операційних систем для мобільних пристроїв [2].

1.3.1 Розробка на Android

У листопаді 2007 року консорціум технологічних компаній під назвою Open Handset Alliance, що складався з виробників мобільних пристроїв, операторів бездротового зв'язку, виробників чипсетів, а також компанії Google [3] представили мобільну операційну систему Android. У вересні 2008 року був випущений перший комерційний смартфон на базі Android 1.0 під назвою HTC Dream. Остання актуальна версія операційної системи Android 14 була випущена 4 жовтня 2023 року Open Handset Alliance під керівництвом Google під кодовою назвою Upside Down Cake [4].

Операційна система Android заснована на модифікованій версії ядра Linux, має відкритий програмний код та переважно призначена для сенсорних мобільних пристроїв, таких як смартфони та планшети.

У грудні 2014 року Google випустила Android studio 1.0, що є офіційним інтегрованим середовищем розробки для застосунків на Android. Ця IDE (Integrated Development Environment) розповсюджується на всіх популярних платформах, такі як Windows, Linux та MacOS. Станом на липень 2023 Giraffe 2022.3.1 є останньою актуальною версією Android studio [5].

Для мобільної розробки на Android використовуються мови програмування засновані на JVM (Java Virtual Machine), такі як Java та Kotlin. Використання інших мов можливе, але потребує допомоги JVM та має обмежену підтримку API (Application Programming Interface).

Розробка здійснюється за допомогою Android SDK (Software Development Kit). До його складу входять приклади програмного коду, документація, бібліотеки, інструмент налагодження та емулятор телефону. Останньою версією SDK для Android 14 є API Level 34.

Google Play є офіційним механізмом розповсюдження застосунків на базі операційної системи Android для кінцевих користувачів. Він також дозволяє поетапне впровадження додатків та розповсюдження попередніх версій застосунків для тестування [6].

1.3.2 Розробка на iOS

У січні 2007 року компанія Apple [7] разом з телефоном iPhone представила власну операційну систему iOS. У червні 2007 року смартфон на базі iOS 1.0 був випущений для продажу. Остання актуальна версія операційної системи iOS 17 була випущена 18 вересня 2023 року [8].

Операційна система iOS базується на власному гібридному XNU (X is Not Unix) ядрі та є пропріетарним програмним забезпеченням, хоча деякий програмний код є відкритим в рамках Apple Public Source License та інших ліцензій.

Офіційним IDE для розробки на iOS є Xcode, що був випущений ще у 2003 році, але SDK для мобільної розробки був представлений пізніше, а саме 6 березня 2008 року. Станом на 18 жовтня 2023 року Xcode версії 15.0.1 є останнім актуальним та стабільним релізом [9].

SDK для мобільної розробки на iOS може бути безплатно завантажений для користувачів MacOS. Він не доступний на таких популярних платформах як Windows та Linux. SDK містить набори, які надають розробникам доступ до різних функцій та послуг пристроїв iOS, таких як апаратні та програмні атрибути. Він також містить емулятор iPhone, який моделює вигляд та стани пристрою на комп'ютері під час розробки. Нові версії SDK виходять разом з новими версіями iOS.

Разом із Xcode iOS SDK допомагає розробникам писати додатки для iOS, використовуючи офіційно підтримувані мови програмування, такі як Objective-C

та Swift. Інші компанії також створили інструменти, які дозволяють розробляти власні додатки для iOS з використанням власних мов програмування.

App Store є офіційним механізмом розповсюдження застосунків на базі операційної системи iOS для кінцевих користувачів. Але для тестування додатків, отримання технічної підтримки та розповсюдження застосунків через App Store, розробники повинні підписатися на програму розробників Apple [10].

1.3.3 Кросплатформна розробка

Кросплатформність (або багатоплатформність, мультиплатформність) – це можливість програмного забезпечення (у нашому випадку, мобільних застосунків) працювати на різних платформах. Кросплатформні додатки створюються за допомогою однієї кодової бази та працюють подібно на кожній операційній системі – iOS та Android. Розробка кросплатформних застосунків не включає написання коду мовою програмування нативної (тобто рідної, спеціально призначеної для даної операційної системи) платформи, але надає враження, подібне до нативного, завдяки візуалізаційному інтерфейсу, який використовує власні елементи управління [11].

На таблиці 1.1 зображено порівняння нативної та кросплатформної розробки.

Таблиця 1.1 – Порівняння нативної та кросплатформної розробки [12]

	Нативна розробка	Кросплатформна розробка
Швидкодія і продуктивність застосунку	Висока продуктивність, більша надійність та менша схильність до помилок та збоїв	Кросплатформні застосунки повільніші ніж нативні, але на практиці різниця майже не помітна
Вартість розробки та підтримки	Висока вартість, бо треба два додатки на дві платформи	Дорожче ніж один нативний застосунок десь на 20-50%, тому у будь-якому випадку дешевший ніж два нативних

Продовження таблиці 1.1

	Нативна розробка	Кросплатформна розробка
Час розробки	Великий, бо застосунок треба писати з нуля	Низький через можливість повторного використання програмного коду
Швидкість виходу на ринок	Повільна, через час розробки	Швидка, через повторне використання програмного коду
Досвід користувача	Найкращий користувацький інтерфейс та швидкодія	Найшвидше розгортання застосунку
Функції застосунку	SDK дає всі можливості API пристрою	Має деякі обмеження до функцій пристрою
Мови програмування	Java та Kotlin для Android Objective-C та Swift для iOS	C#, JavaScript, Dart та інші
Досвід програмування	Опанувати нативне програмування складніше	Простіше, бо багато кросплатформних фреймворків засновані на вебтехнологіях

На основі викладеної вище інформації можна зробити висновок, що кросплатформна розробка краще через наступні переваги:

- дешева та швидка мобільна розробка;
- невелике падіння швидкодії порівняно з нативним підходом;
- простіші мови програмування для опанування [13].

1.4 Аналіз мов програмування для розробки мобільних застосунків

Важливим етапом мобільної розробки є вибір мови програмування. Від досвіду розробника та складності мови залежить час за який буде написано застосунок, а також функціонал додатка.

Як було визначено раніше головними мовами програмування для платформи Android є Java та Kotlin, а для iOS є Objective-C та Swift. Для кросплатформної розробки використовують багато мов, але в даному розділі розглянемо лише Dart.

1.4.1 Java

Java була розроблена спочатку Джеймсом Гослінгом у Sun Microsystems. Вона була випущена в травні 1995 року як основна складова платформи Java в Sun Microsystems. Первинні та референтні реалізації компіляторів Java, віртуальних машин та бібліотек класів спочатку були випущені Sun під пропрієтарними ліцензіями. Oracle [14] пропонує свою власну віртуальну машину Java HotSpot, проте офіційною референтною реалізацією є JVM OpenJDK, яка є безплатним відкритим програмним забезпеченням і використовується більшістю розробників, а також є стандартною JVM для майже всіх дистрибутивів Linux [15].

Java – це високорівнева, клас-орієнтована, об'єктно-орієнтована мова програмування, призначена мати якнайменше залежностей від реалізації. Вона є загальною мовою програмування, призначеною для того, щоб дозволити програмістам писати один раз і запускати де завгодно (WORA – write once, run anywhere), що означає, що скомпільований код Java може запускатися на будь-яких платформах, які підтримують Java, без необхідності повторної компіляції. Додатки Java зазвичай компілюються в байт-код, який може працювати на будь-якій JVM, незалежно від архітектури комп'ютера. Синтаксис Java схожий на C і C++, але має менше низькорівневих можливостей, ніж будь-яка з них. Виконувач середовища Java надає динамічні можливості (такі як відображення і зміна коду в режимі виконання), які зазвичай не доступні в традиційних скомпільованих мовах.

Станом на вересень 2023 року Java 21 є остаточною версією, проте попередні версії Java 8, 11 і 17 все ще офіційно підтримуються.

Мова програмування Java є ключовим компонентом в операційній системі Android, яка є відкритою мобільною платформою. Попри те, що Android, побудований на основі ядра Linux, написаний головним чином мовою C, набір інструментів розробника Android SDK використовує мову Java як основу для створення застосунків Android, але не використовує жодних стандартів Java. Мова байт-коду, яку підтримує Android SDK, несумісна з байт-кодом Java і виконується на власній віртуальній машині, оптимізованій для пристроїв з обмеженим обсягом оперативної пам'яті, таких як смартфони та планшетні комп'ютери. Залежно від версії Android, байт-код може бути або інтерпретований віртуальною машиною Dalvik, або компільований в нативний код виконання Android. Android не надає повної стандартної бібліотеки Java Standard Edition, хоча SDK розробника Android включає незалежну реалізацію значної її частини. Він підтримує Java 6 і деякі функції Java 7, пропонуючи реалізацію, сумісну зі стандартною бібліотекою (Apache Harmony) [16].

Станом на 2023 рік Java використовується 30,55% розробників та займає сьоме місце за популярністю [17].

1.4.2 Kotlin

Kotlin – це кросплатформовна, статично типізована загальна високорівнева мова програмування зі сховищем типів. Kotlin розроблена з метою повної сумісності з Java, і JVM-версія стандартної бібліотеки Kotlin залежить від Java Class Library, але завдяки сховищу типів, синтаксис Kotlin може бути більш лаконічним. Kotlin в основному спрямована на JVM, а також компілюється в JavaScript (наприклад, для фронтенд вебдодатків, що використовують React) або в нативний код за допомогою LLVM (Low Level Virtual Machine) (наприклад, для нативних iOS-додатків, що ділять бізнес-логіку з Android-додатками). Витрати на розробку мови несе компанія JetBrains [18], а Kotlin Foundation захищає торгову марку Kotlin [19].

7 травня 2019 року компанія Google оголосила, що мова програмування Kotlin тепер є їхньою вибраною мовою для розробників застосунків Android. Починаючи з випуску Android Studio 3.0 у жовтні 2017 року, Kotlin включена як альтернативний компілятор до стандартного Java. Компілятор Kotlin для Android створює байт-код Java 8 за замовчуванням (який працює на будь-якій більш пізній віртуальній машині), але дозволяє програмісту вибрати спрямування від Java 9 до 20 для оптимізації, або надає більше можливостей, підтримує взаємодію з записами на двох сторонах для JVM, введених в Java 16, яка стабільна з моменту Kotlin 1.5.

Kotlin має підтримку для вебу за допомогою Kotlin/JS, як через класичний інтерпретаторний бекенд, який було оголошено стабільним з версії 1.3, так і через бекенд на основі проміжного представлення, який було оголошено стабільним з версії 1.8. Kotlin/Native (наприклад, для підтримки Apple silicon) розглядається як бета-версія з версії 1.3.

Станом на 2023 рік Kotlin використовується 9,06% розробників та займає п'ятнадцяте місце за популярністю [17].

1.4.3 Objective-C

Objective-C – це високорівнева загального призначення, об'єктно-орієнтована мова програмування, яка додає стиль сповіщення Smalltalk до мови програмування C. Оригінально розроблена Бредом Коксом і Томом Лавом на початку 1980-х років, вона була вибрана компанією NeXT для своєї операційної системи NeXTSTEP. Завдяки прямому походженню Apple macOS від NeXTSTEP, Objective-C була стандартною мовою програмування, що використовувалася, підтримувалася і просувалася Apple для розробки додатків для macOS і iOS (за допомогою відповідних API, Cocoa і Cocoa Touch) до введення мови програмування Swift у 2014 році [20].

Програми на Objective-C, розроблені для операційних систем, які не від Apple, або які не залежать від API Apple, також можуть бути компільовані для будь-якої платформи, яку підтримує LLVM/Clang.

Станом на 2023 рік Objective-C використовується 2,31% розробників та займає тридцяте місце за популярністю [17].

1.4.4 Swift

Мова програмування Swift – це високорівнева загального призначення, багатопарадигмова, компільована мова програмування, розроблена компанією Apple та спільнотою з відкритим кодом. Swift компілюється до машинного коду, оскільки це компілятор, заснований на LLVM. Swift була вперше випущена в червні 2014 року, і інструментальний набір Swift постачається в Xcode з версії 6, що була випущена у 2014 році [21].

Компанія Apple призначила Swift підтримувати багато основних концепцій, пов'язаних з Objective-C, зокрема динамічний виклик, широке пізнє зв'язування, програмування з можливістю розширення та подібні функції, але в "безпечній" формі, що полегшує виявлення програмних помилок. Swift має функції, які виправляють деякі загальні помилки програмування, такі як скасування вказівника на нуль та надає синтаксичний цукор для уникнення "піраміди відчайнння". Swift підтримує концепцію розширюваності протоколу, систему розширюваності, яку можна застосовувати до типів, структур і класів, яку Apple просуває як справжню зміну в програмних парадигмах, яку вони називають "протокол-орієнтованим програмуванням" (схожим на риси та класи типів).

Swift була представлена на конференції розробників Apple 2014 року та пройшла оновлення до версії 1.2 у 2014 році та значного оновлення до Swift 2 у 2015. Спочатку це була пропрієтарна мова, але версія 2.2 була випущена як відкритий вихідний код під ліцензією Apache License 2.0 3 грудня 2015 року для платформ Apple та Linux.

Swift 4.0, випущена у 2017 році, внесла зміни в деякі вбудовані класи та структури. Код, написаний за попередніми версіями Swift, може бути оновлений за допомогою функціональності міграції, вбудованої в Xcode. Swift 5, випущена в березні 2019 року, внесла стабільний бінарний інтерфейс на платформах Apple, що дозволило включити виконувальну систему Swift в операційні системи Apple. Вона є сумісною з Swift 4.

Swift 5.1 була офіційно випущена в вересні 2019 року. Swift 5.1 розширює стабільні функції мови до часу компіляції з введенням стійкості модуля. Введення стійкості модуля дозволяє створювати й спільно використовувати бінарні бібліотеки, які працюватимуть з майбутніми версіями Swift.

Swift 5.5, офіційно оголошена Apple у 2021, значно розширює підтримку мови для конкурентності та асинхронного коду, в основному, введенням унікальної версії моделі актора.

Поточна версія, Swift 5.9, була випущена в вересні 2023 року і містить в собі макросистему, загальні параметри пакетів та функції володіння, такі як новий оператор споживання.

Станом на 2023 рік Swift використовується 4,65% розробників та займає двадцяте місце за популярністю [17].

1.4.5 Dart

Dart – це об'єктно-орієнтована мова зі схемою на основі класів, яка використовує збірку сміття та має синтаксис у стилі мови програмування C. Вперше представлена у жовтні 2011 року. У 2013 році Dart 1.0 був випущений. Вона може компілюватися в машинний код, JavaScript або WebAssembly. Dart підтримує інтерфейси, змішування, абстрактні класи, узагальнення зі збереженням типів та виведення типів [22].

Початково Dart мала суміш прийняття та критики. Деякі критикували ініціативу Dart за роздроблення вебсередовища через плани включення Dart VM (Virtual Machine) у браузер Chrome. Однак ці плани були скасовані у версії Dart 1.9, і увага була перемкнута на компіляцію коду Dart у JavaScript.

Версія Dart 2.0 була випущена у серпні 2018 року та включала зміни в мові, включно із системою типів.

Dart 2.6 ввів нове розширення – dart2native. Це розширило можливість нативної компіляції на платформи Linux, macOS та Windows. Раніше розробники могли створювати нові інструменти лише з використанням пристроїв Android або iOS. Завдяки цьому розширенню розробники могли розгортати програми у самостійних виконавчих файлів без потреби у встановленні SDK Dart для їх

виконання. Засіб Flutter інтегрує Dart, так що його можна компілювати на невеликих службах, таких як підтримка серверів.

Dart 3.0 вніс зміни до системи типів, вимагаючи звуку надійності щодо нульових значень. У цьому випуску були включені нові функції, такі як records, patterns та модифікатори класів. Dart 3 також включав попередню підтримку Web Assembly.

Dart випустила п'ятий варіант специфікації своєї мови 9 квітня 2021 року. Це охоплює всі синтаксичні конструкції до версії Dart 2.10. Чернетка шостого видання включає всі синтаксичні конструкції до версії 2.13. Затверджені пропозиції для специфікації та чернетки потенційних функцій можна знайти в сховищі мови Dart на GitHub.

Станом на 2023 рік Dart використовується 6,02% розробників та займає вісімнадцяте місце за популярністю [17].

1.5 Аналіз фреймворків для розробки мобільних застосунків

Наступним кроком після обрання кросплатформного підходу до мобільної розробки буде аналіз та вибір фреймворку. В цьому розділі розглянемо Xamarin, React Native та Flutter.

1.5.1 Xamarin

Xamarin – це відкрита платформа для створення сучасних та продуктивних додатків для iOS, Android та Windows на платформі .NET. Xamarin є абстракційним шаром, який керує комунікацією спільного коду з кодом базової платформи. Xamarin працює в керованому середовищі, яке надає зручності, такі як виділення пам'яті та збір сміття [23-24].

Xamarin надає розробникам можливість спільно використовувати в середньому 90% свого додатка на різних платформах. Ця практика дозволяє розробникам писати всю бізнес-логіку однією мовою (або використовувати

наявний код додатка), однак досягати нативної продуктивності, вигляду та відчуття на кожній платформі.

Додатки на Xamarin можуть бути написані на ПК або Mac і компілюватися в нативні пакети додатків, такі як файл .apk на Android або файл .ipa на iOS.

Xamarin надає можливості для прямого виклику бібліотек Objective-C, Java, C та C++, що дозволяє використовувати різноманітний сторонній код. Ця функціональність дозволяє використовувати наявні бібліотеки для iOS та Android, написані на Objective-C, Java або C/C++. Крім того, Xamarin пропонує проекти зв'язування, які дозволяють зв'язувати нативні бібліотеки Objective-C та Java за допомогою декларативного синтаксису.

1.5.2 React Native

React Native – відкритий програмний фреймворк для створення користувацьких інтерфейсів, створеним компанією Meta [25-27].

React Native використовується для розробки додатків для платформ Android, Android TV, iOS, macOS, tvOS, Web та Windows, надаючи розробникам можливість використовувати фреймворк React разом із можливостями нативної платформи.

React Native розпочинає свою роботу у фоновому процесі, який обробляє JavaScript, створений розробниками, безпосередньо на пристрої, і взаємодіє з нативною платформою шляхом обміну серіалізованими даними через механізми асинхронної та пакетної комунікації.

Компоненти React інтегруються з наявним нативним кодом та взаємодіють з нативними API через декларативну парадигму React та мову програмування JavaScript. У сучасних додатках React Native часто використовується TypeScript замість JavaScript через збільшену типову безпеку.

1.5.3 Flutter

Flutter – це набір відкритих інструментів для розробки користувацьких інтерфейсів, створений компанією Google. Він використовується для створення кросплатформних застосунків зі спільного коду для будь-якого веббраузера, операційної системи Fuchsia, Android, iOS, Linux, macOS та Windows [28].

Перша версія Flutter була відома як "Sky" і працювала на операційній системі Android. Вона була представлена на саміті розробників Dart у 2015 році з виразною метою рендерити контент стабільно з частотою 120 кадрів на секунду. 25 січня 2023 року було оголошено про випуск Flutter 3.7.

Додатки Flutter розробляються мовою Dart та використовують багато передових можливостей цієї мови. Для досягнення кращої продуктивності версії додатків Flutter на всіх платформах використовують компіляцію заздалегідь (ahead-of-time compilation).

Flutter успадковує менеджер пакетів Pub та репозиторій програмного забезпечення мови Dart, що дозволяє користувачам публікувати та використовувати власні пакети, а також спеціальні плагіни для Flutter.

Рушій Flutter, в основному написаний мовою C++, надає підтримку низькорівневого рендерингу, використовуючи графічну бібліотеку Skia від Google або спеціальний графічний рівень "Impeller". Крім того, він взаємодіє з платформо залежними SDK, такими як ті, що надаються Android і iOS, для реалізації можливостей доступності, введення/виведення файлів та мережі, підтримки нативних плагінів та іншого.

Бібліотека Foundation, написана мовою Dart, надає базові класи та функції, які використовуються для створення додатків за допомогою Flutter, такі як API для взаємодії з рушієм.

Основним компонентом у програмі Flutter є "віджет", який своєю чергою може складатися з інших віджетів. Віджет описує логіку, взаємодію та дизайн елементу користувацького інтерфейсу з реалізацією, подібною до React. На відміну від інших кросплатформових інструментів, таких як React Native і Xamarin, які малюють віджети з використанням нативних компонентів платформи, Flutter самостійно відображає віджети на піксельному рівні.

Фреймворк Flutter містить два набори віджетів, які відповідають конкретним дизайн-мовам: віджети Material Design реалізують мову дизайну Google з такою ж назвою, а віджети Cupertino реалізують інтерфейсні правила Apple для iOS. Flutter дозволяє розробнику використовувати обидва набори

віджетів на будь-якій платформі, навіть віджети Cupertino на Android. Сторонні пакети можуть використовуватися для автоматичної адаптації дизайну додатка до поточної операційної системи.

Через вищезгадані переваги для розробки мобільного застосунку було обрано фреймворк Flutter та відповідно мову програмування Dart.

1.6 Аналіз фреймворків штучного інтелекту для Flutter

Штучний інтелект може бути впроваджений в застосунки Flutter різними способами, кожен із яких пропонує унікальні функціональні можливості [29-30].

Обробка природної мови (NLP – Natural Language Processing) може підтримувати чат-ботів, які відповідають на запити користувачів у розмовний спосіб, роблячи взаємодію більш природною та співчутливою.

Машинне навчання (ML – Machine Learning) дозволяє застосункам вчитися на поведінці користувачів та адаптуватися з часом. Незалежно від того, чи йдеться про рекомендації продуктів, чи оптимізацію продуктивності, ML додає рівень інтелекту, що робить додатки більш орієнтованими на користувачів.

Від розпізнавання облич до ідентифікації об'єктів, комп'ютерне бачення (CV – Computer Vision) відкриває світ взаємодій на основі зображень. Додатки можуть бачити та розуміти візуальний світ, пропонуючи функції, які є інноваційними та інтерактивними.

1.6.1 Firebase MLKit

ML Kit – це мобільний набір інструментів, який принесе експертизу машинного навчання від Google в Android і iOS додатки в потужному, проте легкодоступному пакеті. ML Kit також надає зручні API, які допомагають використовувати власні моделі TensorFlow Lite у мобільних додатках [31].

ML Kit поставляється з набором готових до використання API для загальних мобільних випадків: розпізнавання тексту, виявлення облич, ідентифікації

пам'ятних місць, сканування штрих-кодів, маркування зображень та визначення мови тексту.

Вибір API ML Kit працює на пристрої чи в хмарі. API на пристрої можуть обробляти дані швидко і працювати навіть при відсутності мережевого підключення. Хмарні API, з іншого боку, використовують потужність технології машинного навчання Google Cloud для надання ще вищого рівня точності.

1.6.2 PyTorch Mobile

PyTorch – це фреймворк машинного навчання, що базується на бібліотеці Torch. Він використовується для застосувань у таких галузях, як комп'ютерний зір та обробка природної мови. Спочатку розроблений компанією Meta AI (Artificial intelligence), він тепер є частиною під егідою Linux Foundation. Це вільне та відкрите програмне забезпечення [32].

Хоча інтерфейс для мови програмування Python більш розвинений і є основним напрямком розвитку, у PyTorch також є інтерфейс для мови програмування C++.

На цей час PyTorch Mobile перебуває на стадії бета-версії та вже широко використовується в промисловості. Згодом він стане доступним як стабільний реліз після закріплення API.

1.6.3 TensorFlow Lite

TensorFlow – це вільна та відкрита бібліотека програмного забезпечення для машинного навчання та штучного інтелекту. Вона може бути використана для вирішення різноманітних завдань, проте основний акцент робиться на навчанні та інференції глибоких нейронних мереж [33].

Розробка TensorFlow відбулася командою Google Brain для внутрішнього використання в дослідженнях та виробництві Google. Перша версія була випущена під ліцензією Apache License 2.0 у 2015 році. У вересні 2019 року Google випустив оновлену версію TensorFlow, яка отримала назву TensorFlow 2.0.

TensorFlow може бути використаний в широкому спектрі мов програмування, включаючи Python, Dart, JavaScript, C++ та Java. Ця гнучкість

надає можливість використовувати TensorFlow в різних галузях та для вирішення різноманітних завдань.

У травні 2017 року компанія Google оголосила про створення програмного стека TensorFlow Lite, спеціально розробленого для розробки мобільних застосунків. У січні 2019 року команда TensorFlow випустила попередню версію розробки мобільного GPU механізму висновку, використовуючи обчислювальні шейдери OpenGL ES 3.1 на пристроях Android і обчислювальні шейдери Metal на пристроях iOS.

TensorFlow Lite має API для мобільних додатків або вбудованих пристроїв для створення та розгортання моделей TensorFlow. Ці моделі стискаються та оптимізуються з метою забезпечення більшої ефективності та вищої продуктивності на пристроях з обмеженими ресурсами.

TensorFlow Lite використовує формат серіалізації даних FlatBuffers для мережових моделей, ухиляючись від формату Protocol Buffers, який використовується в стандартних моделях TensorFlow.

Для розробки моделі на основі штучного інтелекту в комп'ютерній системі обираємо TensorFlow Lite через велику кількість документації, довгого строку в релізі та підтримки від Google.

1.7 Аналіз мови програмування Python для ML

Python – це високорівнева, відкрита мова програмування, яка використовується для розробки загального призначення. Останнім часом вона набула популярності завдяки використанню у сфері науки про дані та машинного навчання. Python універсальна мова програмування, яка має багато пакетів для аналізу даних і може значно підвищити продуктивність [34].

Перевагою є те, що це інтерпретована мова програмування, і, на відміну від скомпільованих мов, кожен рядок коду може бути виконаний зразу ж після його

введення, а результати можна бачити негайно. Це дуже зручно, особливо при використанні правильних інструментів, таких як JupyterLab.

JupyterLab – це останнє веборієнтовне інтерактивне середовище для розробки для ноутбуків, коду та даних. Його гнучкий інтерфейс дозволяє користувачам налаштовувати та організовувати робочі процеси у галузі науки про дані, наукове обчислення, обчислювальну журналістику та машинне навчання. Модульна структура сприяє можливості розширення та покращення функціональності [35].

1.8 Методи машинного навчання

Машинне навчання (ML) – це галузь штучного інтелекту, яка акцентується на розробці алгоритмів та статистичних моделей, які можуть вчитися з даних і робити прогнози. ML є сферою штучного інтелекту, яка допоможе здійснити поставленні задачі з прогнозування майбутніх витрат. У цьому розділі ми розглянемо методи ML [36-43].

1.8.1 Лінійна регресія

Лінійна регресія – це тип алгоритму машинного навчання з учителем, який обчислює лінійний зв'язок між залежною змінною та однією чи кількома незалежними ознаками. Коли кількість незалежних ознак дорівнює одному, це відомо як одновимірна лінійна регресія, а в разі більше однієї ознаки – багатовимірна лінійна регресія. Мета алгоритму – знайти найкраще лінійне рівняння, яке може прогнозувати значення залежної змінної на основі незалежних змінних. Рівняння встановлює пряму лінію, яка представляє зв'язок між залежною та незалежними змінними. Кут нахилу цієї лінії вказує, наскільки залежна змінна змінюється при одиничній зміні незалежних змінних.

Умови для моделі лінійної регресії:

- лінійність (незалежні та залежні змінні мають лінійний зв'язок між собою);
- незалежність (спостереження в наборі даних є незалежними одне від одного);
- гомоскедастичність (на всіх рівнях незалежних змінних дисперсія помилок є стала);
- нормальність (помилки в моделі мають нормальний розподіл);
- відсутність мультиколінеарності (між незалежними змінними мало або жодної кореляції).

На рисунку 1.1 зображено візуалізацію лінійної регресії.

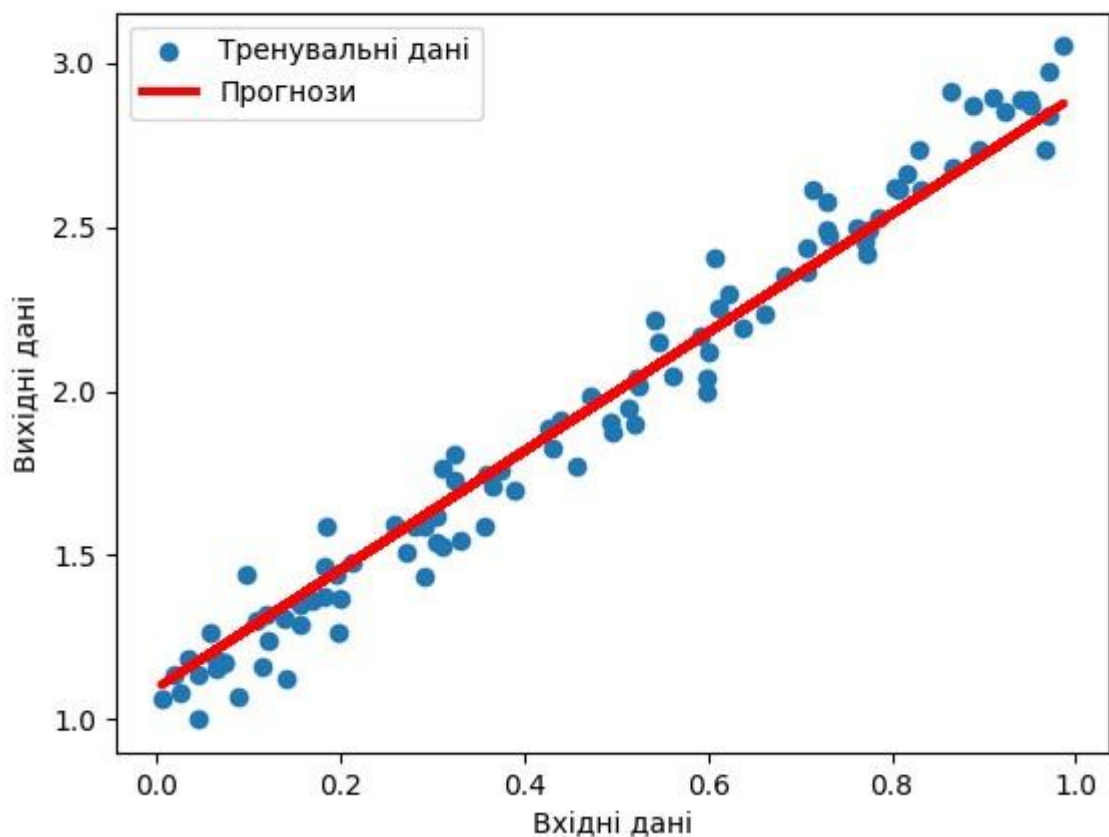


Рисунок 1.1 – Візуалізація лінійної регресії

1.8.2 Метод опорних векторів

Метод опорних векторів (SVM – Support Vector Machine) – потужний алгоритм машинного навчання, який використовується для лінійної чи нелінійної класифікації, регресії та навіть виявлення викидів. SVM може використовуватися для різноманітних завдань, таких як класифікація текстів, класифікація зображень, виявлення спаму, ідентифікація почерку, аналіз експресії генів, виявлення обличчя та виявлення аномалій. SVM адаптивні та ефективні в різних застосуваннях через їхню здатність обробляти високовимірні дані та нелінійні зв'язки.

Переваги SVM:

- ефективний у великих габаритних випадках;
- його пам'ять ефективна, оскільки він використовує підмножину тренувальних точок у функції прийняття рішень, які називаються опорними векторами;
- для функцій прийняття рішень можна вказати різні функції ядра, а також можна вказати власні ядра.

На рисунку 1.2 зображено візуалізацію методу опорних векторів.

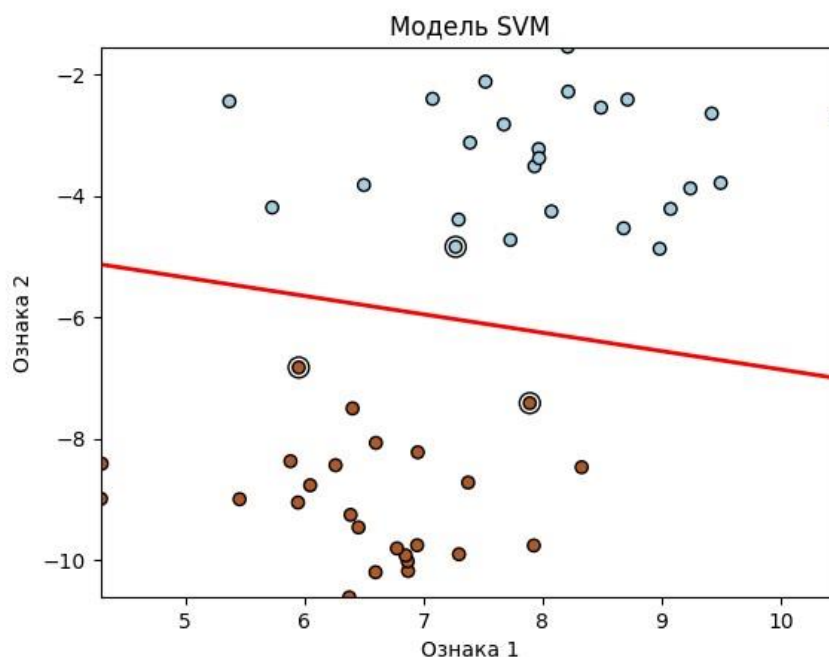


Рисунок 1.2 – Візуалізація методу опорних векторів

1.8.3 Дерево рішень

Дерево рішень (Decision Tree) є одним з найбільш потужних інструментів алгоритмів навчання з учителем, використовуваним як для задач класифікації, так і для задач регресії. Воно будує структуру дерева, схожу на блок-схему, де кожен внутрішній вузол вказує на тестування атрибута, кожна гілка представляє результат тесту, а кожен листовий вузол (термінальний вузол) містить мітку класу. Дерево конструюється шляхом рекурсивного розбиття навчальних даних на підмножини, базуючись на значеннях атрибутів, поки не буде виконано критерій зупинки, такий як максимальна глибина дерева чи мінімальна кількість зразків, необхідних для розбиття вузла.

Під час тренування алгоритм дерева рішень вибирає найкращий атрибут для розбиття даних за допомогою метрики, такої як ентропія чи імпульсивність Джині, яка вимірює ступінь нечистоти або випадковості в підмножинах. Мета полягає в знаходженні атрибута, який максимізує приріст інформації чи зменшення нечистоти після розбиття.

Дерево рішень може бути навчено, розбивши вихідний набір на підмножини на основі мір вибору атрибутів. Міра вибору атрибутів (ASM – Attribute Selection Measure) є критерієм, який використовується в алгоритмах дерева рішень для оцінки корисності різних атрибутів для розділення набору даних. Метою ASM є визначення атрибута, який створить найбільш однорідні підмножини даних після розділення, тим самим максимізуючи приріст інформації. Цей процес повторюється для кожного отриманої підмножини в рекурсивний спосіб, відомий як рекурсивне розділення. Рекурсія завершується, коли підмножина в вузлі має однакове значення цільової змінної або коли подальше розділення не додає значення передбаченням.

Ентропія – це міра ступеня випадковості чи невизначеності в наборі даних. У випадку класифікації вона вимірює випадковість на основі розподілу класових міток в наборі даних.

На рисунку 1.3 зображено візуалізацію дерева рішень.

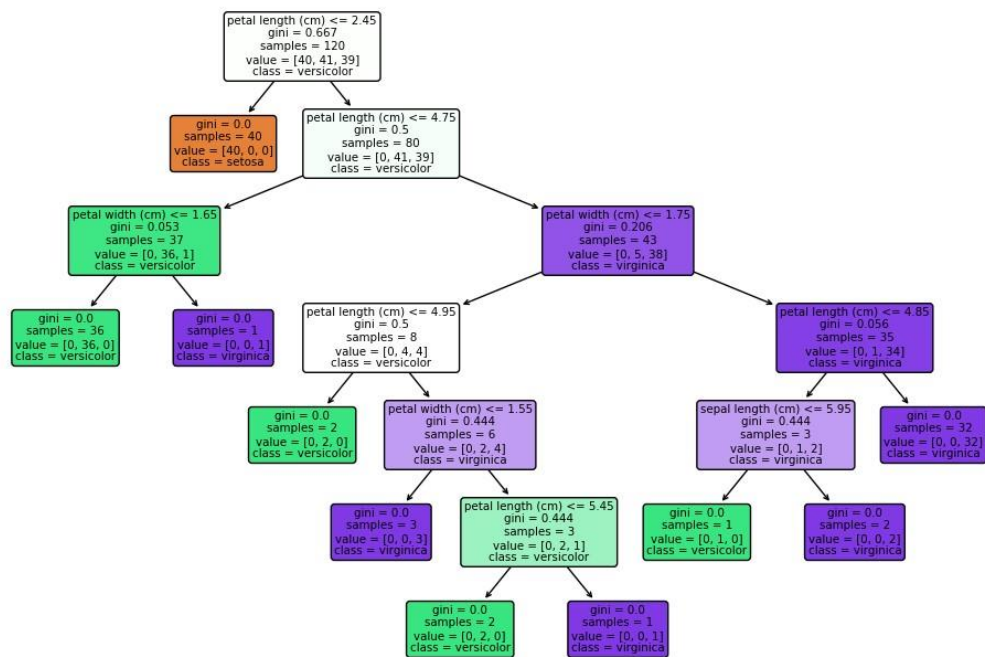


Рисунок 1.3 – Візуалізація дерева рішень

1.8.4 Випадковий ліс

Випадковий ліс – це ансамбльна техніка, здатна виконувати як завдання регресії, так і класифікації за допомогою кількох дерев прийняття рішень і техніки, відомої як Bootstrap та Aggregation, або ж узагальнення. Основна ідея полягає в тому, щоб комбінувати кілька дерев прийняття рішень для визначення кінцевого виводу, замість покладання на окремі дерева прийняття рішень.

У випадковому лісі використовується кілька дерев прийняття рішень як базових моделей навчання. З набору даних випадковим чином вибираються рядки та ознаки, формуючи вибірккові набори даних для кожної моделі. Цей етап відомий як Bootstrap.

На рисунку 1.4 зображено візуалізацію випадкового лісу.

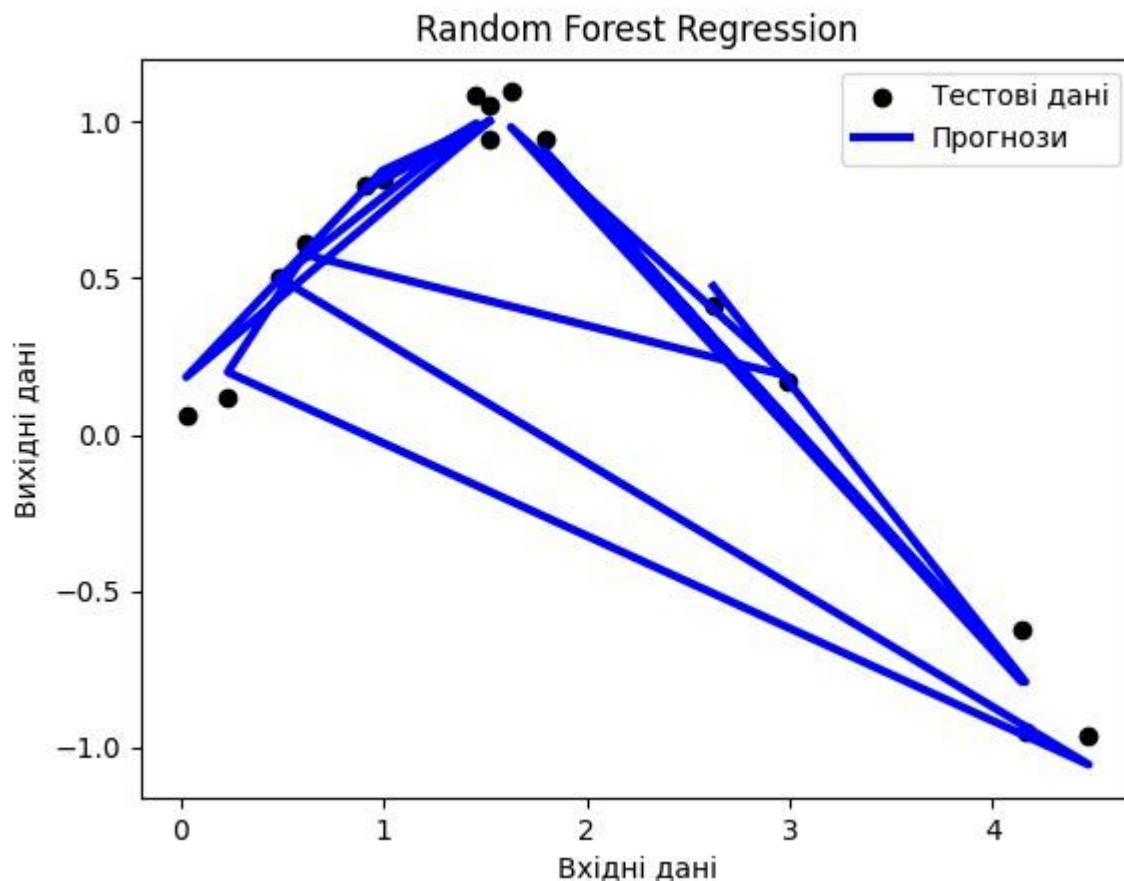


Рисунок 1.4 – Візуалізація випадкового лісу

1.8.5 Рекурентна нейронна мережа

Рекурентна нейронна мережа (RNN – Recurrent Neural Network) – це тип нейронної мережі, де вихід з попереднього кроку подається як вхід на поточний крок. У традиційних нейронних мережах всі вхідні та вихідні дані незалежні одне від одного, але в випадках, коли потрібно передбачити наступне слово в реченні, необхідні попередні слова, тому є необхідність пам'ятати попередні слова. Таким чином з'явилася RNN, яка розв'язала цю проблему за допомогою прихованого шару (Hidden Layer). Основною та найважливішою особливістю RNN є його прихований стан (Hidden State), який запам'ятовує певну інформацію про послідовність. Стан також називається станом пам'яті (Memory State), оскільки він зберігає попередній вхід в мережу. Він використовує ті самі параметри для кожного входу, оскільки він виконує ту саму задачу для всіх входів або

прихованих шарів для отримання вихідних даних. Це зменшує складність параметрів, на відміну від інших нейронних мереж.

Переваги рекурентної нейронної мережі:

- RNN запам'ятовує кожен фрагмент інформації з плином часу (це корисно для прогнозування часових рядів лише завдяки можливості пам'ятати попередні входи, також це відомо як довга короткочасна пам'ять);
- рекурентні нейронні мережі використовуються разом зі згортковими шарами для розширення ефективного піксельного оточення.

Недоліки рекурентної нейронної мережі:

- проблеми зникнення та вибуху градієнта;
- навчання RNN є дуже важким завданням;
- вона не може обробляти дуже довгі послідовності при використанні $\tanh$ або relu як функції активації.

Застосування рекурентної нейронної мережі:

- моделювання мови та генерація тексту;
- розпізнавання мови;
- машинний переклад;
- розпізнавання зображень, виявлення обличчя;
- прогнозування часових рядів.

На рисунку 1.5 зображено візуалізацію рекурентної нейронної мережі.

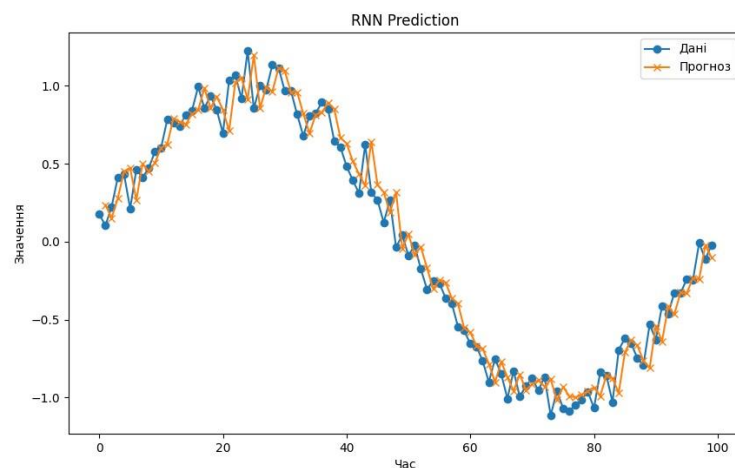


Рисунок 1.5 – Візуалізація рекурентної нейронної мережі

1.8.6 Довга короткочасна пам'ять

Довга короткочасна пам'ять (LSTM – Long Short Term Memory) – це тип рекурентної нейронної мережі (RNN), спеціально розроблений для роботи з послідовними даними, такими як часові ряди, мовлення та текст. Мережі LSTM здатні вивчати довгострокові залежності в послідовних даних, що робить їх відповідними для завдань, таких як переклад мов, розпізнавання мовлення та прогнозування часових рядів.

Переваги LSTM:

- LSTM-мережі можуть захоплювати довгострокові залежності, оскільки вони мають пам'ять, здатні до тривалого зберігання інформації;
- у традиційних RNN існує проблема зникнення та розширення градієнтів при тренуванні моделей на довгих послідовностях, LSTM-мережі розв'язують цю проблему за допомогою механізму воріт, який вибірково відновлює чи забуває інформацію;
- LSTM дозволяє моделі захоплювати та пам'ятати важливий контекст, навіть коли між важливими подіями в послідовності є значний проміжок часу, таким чином, де розуміння контексту важливе, використовуються LSTM.

Недоліки LSTM:

- порівняно з простішими архітектурами, такими як нейронні мережі з передачею вперед, LSTM-мережі обчислювально витратні, це може обмежити їх масштабованість для великомасштабних наборів даних або обмежених середовищ;
- тренування LSTM-мереж може бути більш часомістким порівняно з простішими моделями через їхню обчислювальну складність, таким чином, для досягнення високої ефективності тренування LSTM часто вимагає більше даних і триваліших часів навчання;
- оскільки обробка речень відбувається послідовно по словах, важко паралелізувати обробку речень.

На рисунку 1.6 зображено візуалізацію LSTM.

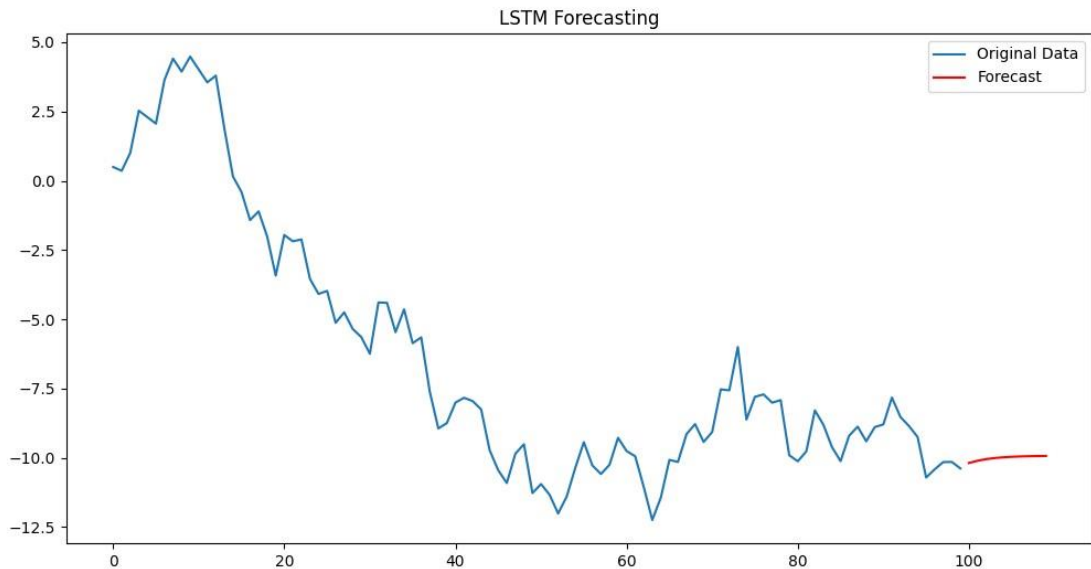


Рисунок 1.6 – Візуалізація LSTM

1.8.7 Алгоритм k-найближчих сусідів

Алгоритм k-найближчих сусідів (KNN – K-Nearest Neighbors) – це надійний і інтуїтивно зрозумілий метод машинного навчання, який використовується для вирішення завдань класифікації та регресії. Використовуючи концепцію схожості, KNN передбачає мітку або значення нової точки даних, враховуючи її k найближчих сусідів у навчальному наборі. Цей алгоритм широко застосовується в реальних сценаріях, оскільки він є непараметричним, що означає, що він не робить жодних внутрішніх припущень щодо розподілу даних.

Переваги алгоритму KNN:

- легкість втілення, оскільки складність алгоритму не настільки велика;
- легко адаптується, під час роботи алгоритму KNN весь набір даних зберігається в пам'яті, тому кожного разу, коли додається новий приклад або точка даних, алгоритм вирівнюється згідно з цим новим прикладом і має свій внесок у майбутні передбачення;
- кілька гіперпараметрів, єдині параметри, які потрібні при навчанні алгоритму KNN це значення k і вибір метрики відстані, яку ми хочемо вибрати з нашої метрики оцінки.

Недоліки алгоритму KNN:

- не масштабується, алгоритм KNN також вважається важким алгоритмом, основне значення цього терміну полягає в тому, що для його виконання потрібно багато обчислювальної потужності, а також зберігання даних, це робить цей алгоритм як часовою, так і ресурсозатратною задачею;
- прокляття розмірності, існує термін, відомий як явище піку, згідно з яким алгоритм KNN стикається з проблемою прокляття розмірності, що означає, що алгоритм має важко класифікувати точки даних належним чином, коли розмірність дуже велика;
- схильний до перенавчання, оскільки алгоритм стикається з прокляттям розмірності, він також схильний до проблеми перенавчання, таким чином, зазвичай застосовуються техніки відбору ознак та зменшення розмірності для розв’язання цієї проблеми.

На рисунку 1.7 зображено візуалізацію алгоритму KNN.

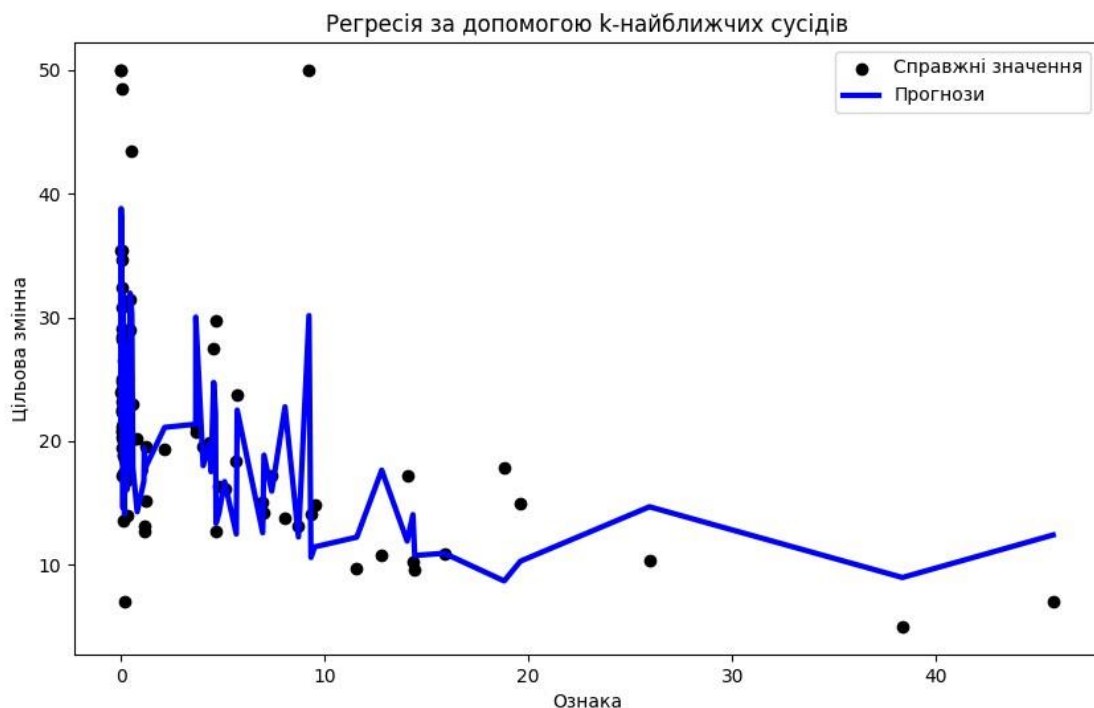


Рисунок 1.7 – Візуалізація алгоритму KNN

1.9 Висновки за розділом 1

У першому розділі було визначено поняття управління витратами, сформульовано завдання та обрано кросплатформений підхід для мобільної розробки. Мова програмування Dart та фреймворк Flutter були обрані для створення мобільного застосунку, тоді як для розробки моделі на основі штучного інтелекту у мобільному додатку був обраний фреймворк TensorFlow Lite. Також проведено аналіз мови програмування Python у контексті машинного навчання та розглянуті різні методи машинного навчання.

2 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ

Розробка комп'ютерної системи складається з розробки мобільного застосунку в IDE Android Studio за допомогою фреймворку Flutter та мови програмування Dart та розробки моделі на основі штучного інтелекту інструментами мови Python та TensorFlow Lite.

2.1 Розробка мобільного застосунку

Створення мобільного додатку складається з трьох етапів:

- вибір та впровадження патерну проєктування;
- розробка та налаштування бази даних для зберігання та обробки інформації;
- створення віджетів представлення для користувацького інтерфейсу.

У цьому розділі будуть розглянуті всі ці етапи.

2.1.1 Впровадження архітектурного шаблону

Архітектурним шаблоном було обрано MVC (Model-View-Controller). Патерн проєктування MVC визначає, що застосунок складається з моделі даних (Model), інформації для відображення (View) та інформації для управління (Controller). Цей шаблон вимагає розділення кожного з цих компонентів на окремі об'єкти. MVC основним чином стосується інтерфейсного/взаємодійного рівня застосунку [44].

Модель містить чисті дані застосунку та логіку, яка описує, як представити дані користувачеві. Це просто дані, які передаються по системі, наприклад, від сервера до представлення та від представлення до бази даних.

Представлення відображає дані моделі користувачу та знає, як отримати доступ до даних моделі, але воно не знає, що означають ці дані або як користувач може ними маніпулювати. Представлення просто відображає дані застосунку на екрані.

Контролер існує між представленням та моделлю. Він слухає події, спричинені представленням або іншим зовнішнім джерелом та виконує відповідну реакцію на ці події. У більшості випадків реакцією є виклик методу моделі. Оскільки представлення та модель пов'язані механізмом сповіщення, результат цієї дії автоматично відображається користувачу.

Переваги MVC:

- декілька розробників можуть одночасно працювати над моделлю, контролером та представленнями;
- MVC дозволяє логічно групувати пов'язані дії контролера, представлення для конкретної моделі також групуються;
- у моделей може бути декілька представлень;
- компоненти застосунку легко керовані та менш залежні один від одного для належного функціонування системи.

Недоліки MVC:

- знання кількох технологій стає нормою, розробники, які використовують MVC, повинні володіти великою кількістю технологій;

– навігація у фреймворку може бути складною через введення нових рівнів абстракції та вимагає від користувачів пристосуватися до критеріїв розкладання MVC.

На рисунку 2.1 зображено скриншот IDE Android Studio на якому показано склад теки `lib`, яка містить в собі теки проєкту розбиті за принципами патерну проєктування MVC.

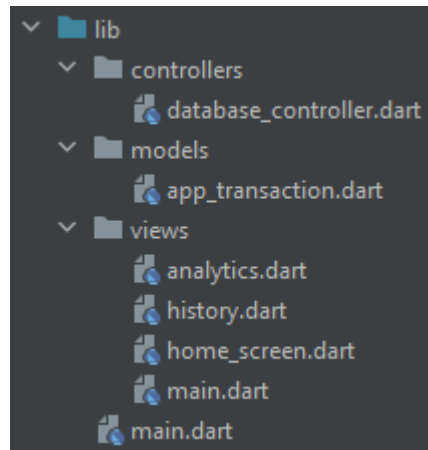


Рисунок 2.1 – Теки проєкту розбиті за принципами патерну проєктування MVC

У теці моделей є файл `app_transaction.dart`, що відповідає за дані транзакцій у застосунку. У теці контролеру є файл `database_controller`, що відповідно до назви відповідає за зв'язок з базою даних. У теці представлення є файли `analytics.dart`, `history.dart`, `home_screen.dart` та `main.dart`, що відповідають за відповідні екрани застосунку. Файл `main.dart`, що знаходиться у теці `lib` має в собі точку входу застосунку.

2.1.2 Розробка бази даних

База даних – це організована колекція даних, збережених у комп'ютерній системі та контрольована системою керування базами даних. Дані в загальних базах даних моделюються у вигляді таблиць, що робить ефективними операції запитування та обробки. Для операцій з даними часто використовується мова структурованих запитів (SQL – Structured Query Language) [45].

Для мобільного застосунку була обрана полегшена реляційна система керування базами даних SQLite, а саме бібліотека sqflite версії 2.3.0, що є пакетом оптимізованим для використання у фреймворці Flutter [46].

Через призначення SQLite для мобільних пристроїв, вона має обмежену кількість типів даних.

Підтримувані типи даних у SQLite:

- INTEGER (відповідний тип у мові Dart `int`, діапазон значень від -2^{63} до $2^{63}-1$);
- REAL (відповідний тип у мові Dart `num`);
- TEXT (відповідний тип у мові Dart `String`);
- BLOB (відповідний тип у мові Dart `Uint8List`).

Тип даних `bool` не підтримується, необхідно використовувати INTEGER зі значеннями 0 або 1. Також не підтримується тип даних `DateTime` для зберігання дати та часу, які можна зберігати в INTEGER (формат Unix time) або у TEXT (стандарт ISO8601 [47]).

Дата та час у мобільному застосунку будуть зберігатись за стандартом ISO8601 та відповідно переводитись з типу SQLite TEXT у `DateTime` мови Dart, а зворотним способом через приведення `DateTime` до типу `String` методом `toIso8601String()`.

Перед створенням бази даних необхідно визначити клас моделі `AppTransaction`, що наведено у лістингу 2.1.

Лістинг 2.1 – Клас `AppTransaction`

```
class AppTransaction {
  int? id;
  final String name;
  final double amount;
  final DateTime date;

  AppTransaction({
    this.id,
    required this.name,
    required this.amount,
    required this.date,
  });
}
```

```

factory AppTransaction.fromJson(Map<String, dynamic> json) {
  return AppTransaction(
    id: json['id'],
    name: json['name'],
    amount: json['amount'].toDouble(),
    date: DateTime.parse(json['date']),
  );
}

Map<String, dynamic> toJson() {
  return {
    'id': id,
    'name': name,
    'amount': amount,
    'date': date.toIso8601String(),
  };
}
}

```

Клас `AppTransaction` складається з чотирьох полів: необов'язкового (через те, що система керування базами даних сама видає ідентифікатор при додаванні інформації) `id` типу `int`, `name` типу `String`, `amount` типу `double` та `date` типу `DateTime`.

Відповідно до того, що це транзакції до змінних вказано параметр `final`, що забороняє зміну значень після ініціалізації. Також у класі визначено два методи `fromJson()` та `toJson()` для отримання інформації з бази даних у вигляді `Map<String, dynamic>` та відповідного заповнення полів класу та зворотного процесу для заповнення бази даних.

Після створення класу `AppTransaction` можна реалізувати клас контролеру бази даних `DatabaseController`, де буде ініціалізація бази даних та якщо таблиці `transactions` нема вона буде створена з вказанням відповідних типів, що підтримує `SQLite`.

У лістингу 2.2 наведено фрагмент класу `DatabaseController`, де показана ініціалізація бази даних.

Лістинг 2.2 – Клас `DatabaseController`, ініціалізація бази даних

```

import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
import '../models/app_transaction.dart';

```

```

class DatabaseController {
  late Future<Database> _database;
  static const tableName = 'transactions';

  DatabaseController() {
    _initializeDatabase();
  }

  Future<void> _initializeDatabase() async {
    _database = openDatabase(
      join(await getDatabasesPath(), '$tableName.db'),
      onCreate: (db, version) {
        return db.execute(
          'CREATE TABLE $tableName(id INTEGER PRIMARY KEY, name
TEXT, amount REAL, date TEXT)',
        );
      },
      version: 1,
    );
  }
}

```

Якщо таблиця transactions існує то після ініціалізації бази даних вона буде відкрита, у зворотному випадку вона буде створена з полями: id типу INTEGER PRIMARY KEY (автоматичне присвоєння ідентифікатора), name типу TEXT, amount типу REAL та date типу TEXT.

Змінна _database оголошена з параметром late, що означає обов'язкову ініціалізацію пізніше (це необхідно через те що при використанні методів для роботи з базою даних, вона не може мати значення NULL). Також знак підкреслення в мові Dart говорить про приватність змінної, а використання класу Future з типом Database бібліотеки sqflite необхідно для написання асинхронного програмного коду.

Метод _initializeDatabase() є асинхронним тому використовується параметр async та клас Future.

Для повного контролю над базою даних у файловій системі пристрою було використано бібліотеку path версії 1.8.3 [48].

Метод join() бібліотеки path приймає два параметри: стандартний шлях до бази даних на пристрої асинхронним методом getDatabasesPath() та назву таблиці з розширенням .db.

У лістингу 2.3 наведені інші методи класу DatabaseController.

Лістинг 2.3 – Клас DatabaseController, інші методи

```
Future<void> insertTransaction(AppTransaction transaction)
async {
    final db = await _database;

    await db.insert(
        tableName,
        transaction.toJson(),
        conflictAlgorithm: ConflictAlgorithm.replace,
    );
}

Future<List<AppTransaction>> fetchTransactions() async {
    final db = await _database;

    final List<Map<String, dynamic>> maps = await
db.query(tableName);

    return maps.map((map) =>
AppTransaction.fromJson(map)).toList();
}

Future<void> updateTransaction(AppTransaction transaction)
async {
    final db = await _database;

    await db.update(
        tableName,
        transaction.toJson(),
        where: 'id = ?',
        whereArgs: [transaction.id],
    );
}

Future<void> deleteTransaction(int id) async {
    final db = await _database;

    await db.delete(
        tableName,
        where: 'id = ?',
        whereArgs: [id],
    );
}
```

Представлені методи є стандартними при роботі з системою керування базами даних та дозволяють робити: додавання (insertTransaction), отримання всіх

транзакцій (fetchTransactions), зміну (updateTransaction) та видалення (deleteTransaction).

Після реалізації контролеру бази даних та моделі транзакцій можна виконати ініціалізацію бази даних у файлі main.dart, що має точку входу в застосунок.

У лістингу 2.4 наведено програмний код файлу main.dart.

Лістинг 2.4 – Вміст файлу main.dart

```
import 'package:flutter/material.dart';
import 'controllers/database_controller.dart';
import 'views/home_screen.dart';
import 'package:tflite_flutter/tflite_flutter.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  DatabaseController databaseController = DatabaseController();

  runApp(FlutterApp(databaseController: databaseController));
}

class FlutterApp extends StatelessWidget {
  final DatabaseController databaseController;

  const FlutterApp({Key? key, required
    this.databaseController})
    : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.purple,
      ),
      home: HomeScreen(databaseController: databaseController),
    );
  }
}
```

Метод WidgetsFlutterBinding.ensureInitialized() відповідає за ініціалізацію платформних служб та гарантує коректну роботу Flutter до виклику асинхронного методу, а саме конструктору DatabaseController(), що викликає асинхронну ініціалізацію бази даних, при створенні об'єкту databaseController класу DatabaseController.

Далі цей об'єкт передається в клас `FlutterApp`, а після нього в клас `HomeScreen`, що відповідальний за відображення всіх екранів застосунку.

2.1.3 Розробка віджетів представлення

Віджети є центральною ієрархією класів у фреймворку Flutter та незмінним описом частини користувацького інтерфейсу. Віджети можуть бути розгорнуті в елементи, які управляють базовим деревом відображення [49].

Самі віджети не мають змінної стану (усі їхні поля повинні бути остаточними, тобто такими, що не зміняться після ініціалізації). Якщо необхідно підв'язати змінний стан з віджетом, можна використати `StatefulWidget`, що створює об'єкт `State` (за допомогою `StatefulWidget.createState`) кожного разу, коли його розгортають в елемент і включають у дерево.

Розглянемо клас представлення `HomeScreenState`, що відповідальний за відображення та зміну поточних віджетів за допомогою нижнього навігаційного меню. Програмний код цього класу представлено у лістингу 2.5.

Лістинг 2.5 – Клас `HomeScreenState`

```
class HomeScreenState extends State<HomeScreen> {
  final List<String> items = ['Головна', 'Історія', 'Аналітика'];
  late final List<Widget> _pages;

  @override
  void initState() {
    super.initState();
    _pages = [
      Main(databaseController: widget.databaseController),
      History(databaseController: widget.databaseController),
      Analytics(databaseController: widget.databaseController),
    ];
  }

  int _currentIndex = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: _pages[_currentIndex],
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _currentIndex,
        onTap: (index) {
          setState(() {
            _currentIndex = index;
          });
        },
      ),
    );
  }
}
```

```

    });
  },
  items: [
    BottomNavigationBarItem(
      icon: const Icon(Icons.home),
      label: items[0],
    ),
    BottomNavigationBarItem(
      icon: const Icon(Icons.list),
      label: items[1],
    ),
    BottomNavigationBarItem(
      icon: const Icon(Icons.show_chart),
      label: items[2],
    ),
  ],
),
);
}
}

```

Коли користувач натискає на пункт нижнього навігаційного меню віджет застосунку реагує на це та відповідно змінює `_currentIndex`, що в залежності від значення (0 – Головна, 1 – Історія, 2 – Аналітика) відображає відповідний віджет представлення. Також при здійсненні виклику віджета в нього передається об'єкт бази даних, що був об'явлений в `main.dart`, де знаходиться точка входу в застосунок. Це можливо через те, що клас `HomeScreen` був оголошений як `StatefulWidget`.

Розглянемо клас `History`, що відповідальний за відображення користувачеві інформації про минулі транзакції, а також їх додавання. Фрагмент програмного коду, що демонструє завантаження транзакцій через контролер бази даних наведено у лістингу 2.6.

Лістинг 2.6 – Клас `History`, завантаження транзакцій

```

class HistoryState extends State<History> {
  List<AppTransaction> transactions = [];

  @override
  void initState() {
    super.initState();
    _loadTransactions();
  }
}

```

```

Future<void> _loadTransactions() async {
    final transactions = await
widget.databaseController.fetchTransactions();

    transactions.sort((a, b) => b.date.compareTo(a.date));

    setState(() {
        this.transactions = transactions;
    });
}

String getFormattedPrice(double price) {
    String sign = price >= 0 ? '+' : '-';
    return '$sign${price.abs().toStringAsFixed(2)} ₾';
}

Color getColor(double price) {
    return price < 0 ? Colors.red : Colors.green;
}
}

```

У даному фрагменті програмного коду продемонстровано, що при створенні віджета History (метод initState()) викликається метод _loadTransactions(), який робить запит до бази даних та отримує всі транзакції, які сортує в порядку убивання по даті. Також представлені два методи getFormattedPrice() та getColor(), що виводять суму з двома знаками після коми, додають знак гривні ₾ та в залежності від типу транзакції помічають її червоним чи зеленим кольором відповідно.

Наступною можливістю класу History є додавання транзакції. Фрагмент програмного коду що демонструє метод _addTransaction() представлено у лістингу 2.7.

Лістинг 2.7 – Клас History, додавання транзакції

```

Future<void> _addTransaction() async {
    final String name = nameController.text;
    final double amount =
double.tryParse(amountController.text) ?? 0.0;

    if (name.isNotEmpty) {
        DateTime? selectedDate = await _selectDate();
        TimeOfDay? selectedTime = await _selectTime();
    }
}

```

```

        if (selectedDate != null && selectedTime != null) {
            final newTransaction = AppTransaction(
                name: name,
                amount: amount,
                date: DateTime(
                    selectedDate.year,
                    selectedDate.month,
                    selectedDate.day,
                    selectedTime.hour,
                    selectedTime.minute,
                ),
            );

            await
            widget.databaseController.insertTransaction(newTransaction);

            nameController.clear();
            amountController.clear();

            _loadTransactions();

            ScaffoldMessenger.of(context).showSnackBar(
                const SnackBar(
                    content: Text('Транзакцію успішно додано'),
                ),
            );
        }
    } else {
        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(
                content: Text('Введіть дійсну назву та суму'),
                backgroundColor: Colors.red,
            ),
        );
    }
}

```

У наведеному програмному коді додається транзакція в базу даних, якщо в полях вводу не було виявлено помилок.

Для форматування дати та додання української локалізації при відображенні назв місяців була використана бібліотека intl версії 0.18.1 [50].

На рисунках 2.2-2.3 наведені скріншоти з емулятора Android Studio, що демонструють роботу представлення мобільного застосунку, а саме перегляд історії та ввід нової транзакції відповідно.

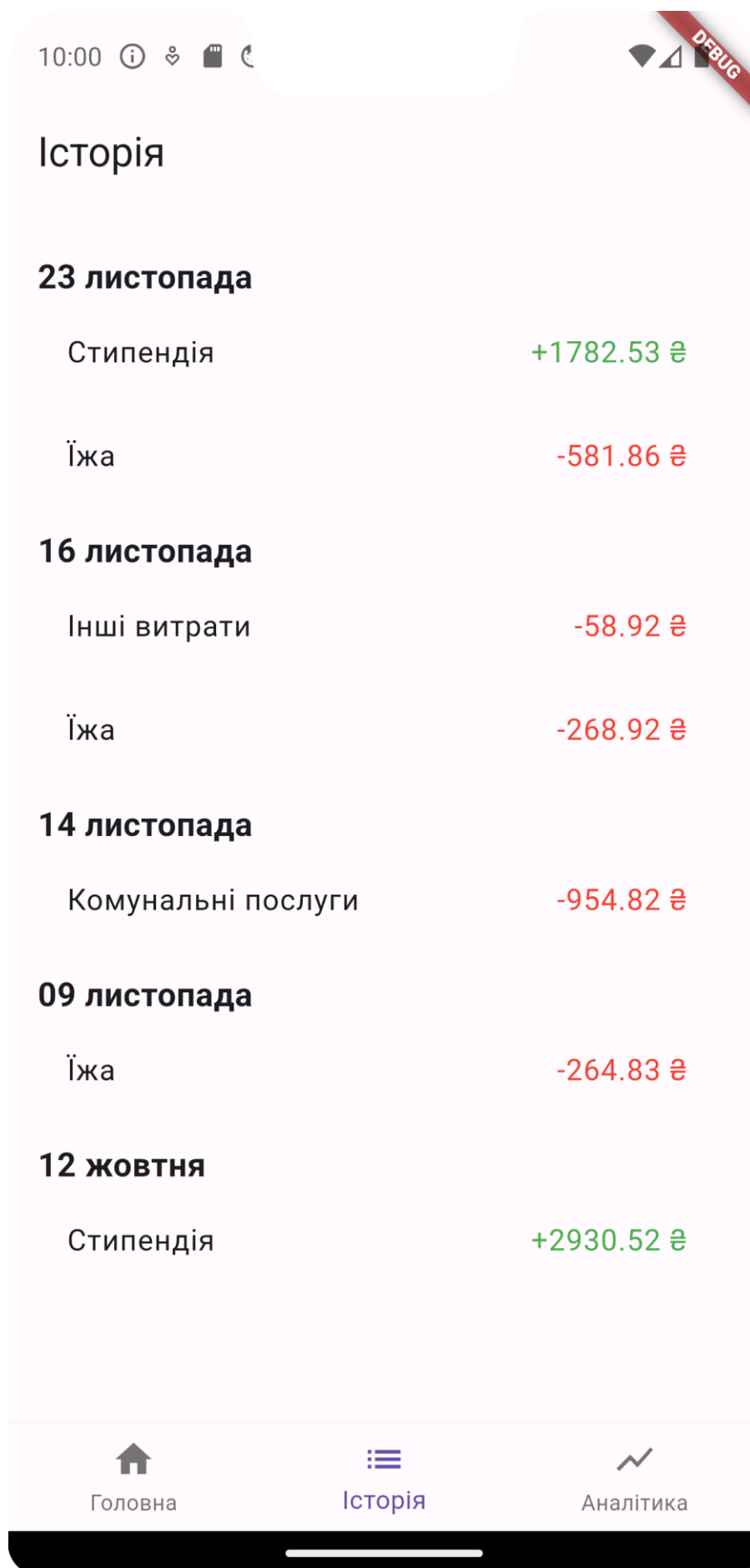


Рисунок 2.2 – Перегляд історії

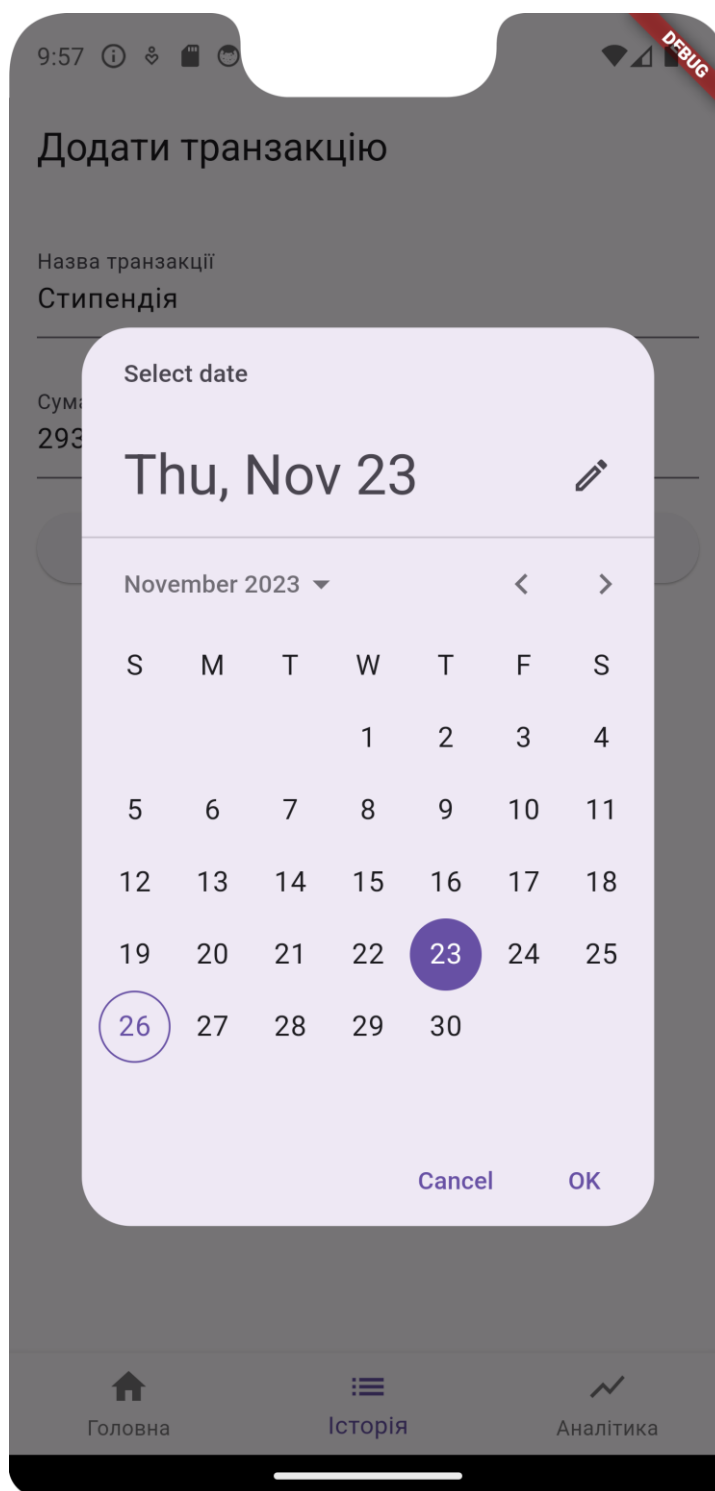


Рисунок 2.3 – Додавання транзакції

Для візуалізації результатів прогнозування у вигляді графіків у мобільному застосунку представлений клас *Analytics*. Для цього була використана бібліотека *fl_chart* версії 0.64.0 [51].

У лістингу 2.8 наведено фрагмент програмного коду класу *Analytics*, що малює графік користувачеві.

Лістинг 2.8 – Клас Analytics, побудова графіку

```
@override
Widget build(BuildContext context) {
  List<FlSpot> spots = data.entries.map((entry) {
    return
    FlSpot(entry.key.millisecondsSinceEpoch.toDouble(),
    entry.value);
  }).toList();

  return Scaffold(
    appBar: AppBar(
      title: const Text('Аналітика'),
    ),
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: LineChart(
        LineChartData(
          lineBarsData: [
            LineChartBarData(
              spots: spots,
              isCurved: true,
              color: Colors.purple,
              dotData: const FlDotData(show: true),
              belowBarData: BarAreaData(show: false),
            ),
          ],
          borderData: FlBorderData(show: false),
          gridData: const FlGridData(show: true),
          titlesData: const FlTitlesData(
            show: true,
            bottomTitles: AxisTitles(
              sideTitles: SideTitles(
                showTitles: false,
              ),
            ),
            topTitles: AxisTitles(
              sideTitles: SideTitles(
                showTitles: false,)),),),),),),),),);}
```

2.2 Розробка моделі на основі штучного інтелекту

Розробка моделі на основі штучного інтелекту складається з тренування моделі засобами мови Python, збереження у форматі TensorFlow Lite та подальшою інтеграцією у комп'ютерну систему.

2.2.1 Тренування моделі

Для машинного навчання на мові Python були використані наступні бібліотеки: NumPy, Pandas та Matplotlib. Вони допоможуть у математичних обчисленнях, поданні масивів даних та представленні результатів через візуалізацію на графіках.

NumPy є універсальним пакетом для обробки масивів. Він надає об'єкт багатовимірного масиву високої продуктивності та інструменти для ефективної роботи з такими масивами. Цей пакет є фундаментальним для виконання наукових обчислень в мові Python. Окрім очевидних наукових застосувань, NumPy може бути використаний як ефективний багатовимірний контейнер для зберігання загальних даних [52].

Бібліотека Pandas є відкритим програмним забезпеченням, яке побудоване на основі бібліотеки NumPy. Це пакет для мови Python, який пропонує різноманітні структури даних та операції для обробки числових даних і часових рядів. Він є популярним завдяки спрощенню імпорту та аналізу даних. Також Pandas швидко працює, через що має високу продуктивність та ефективність. Основною структурою даних в Pandas є DataFrame [53].

DataFrame – це двовимірна структура даних, де дані вирівнюються у табличному вигляді за допомогою рядків та стовпців. Він складається з трьох основних компонентів: даних, рядків та стовпців.

Matplotlib є бібліотекою візуалізації для створення двовимірних графіків масивів у мові Python. Однією з найбільших переваг візуалізації є те, що вона надає візуальний доступ до великої кількості даних у легко засвоюваних візуальних формах. Matplotlib містить в собі кілька типів графіків, таких як лінійні, стовпчасті, розсіювання, гістограми тощо [54].

Для навчання моделі на основі штучного інтелекту необхідно мати дані, які необхідно розділити на два набори: тренувальні та тестові. Далі треба обрати архітектуру обраної моделі. Процес тренування можна контролювати виводом значень середньоквадратичної похибки та візуалізацією через графік, де можна побачити результат.

У лістингу 2.9 представлено програмний код мовою Python, що підключає дані для навчання, підготовлює їх для тренування, створює модель на основі штучного інтелекту, навчає та зберігає цю модель.

Лістинг 2.9 – Тренування моделі лінійної регресії

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.preprocessing import MinMaxScaler

# Підключення даних для навчання
data = pd.read_csv('income_data.csv')
data['Date'] = pd.to_datetime(data['Date'])
data = data.set_index('Date')

# Нормалізація даних
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(data)

# Розділення даних на тренувальний та тестовий набори
train_size = int(len(data_scaled) * 0.8)
train_data, test_data = data_scaled[0:train_size, :],
data_scaled[train_size:len(data_scaled), :]

# Розділення на ознаки (X) та мітки (y)
X_train, y_train = np.arange(len(train_data)).reshape(-1, 1),
train_data
X_test, y_test = np.arange(len(train_data),
len(data_scaled)).reshape(-1, 1), test_data

# Визначення моделі лінійної регресії
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(1,), name='input'),
    tf.keras.layers.Dense(1, activation='linear',
name='output')
])
model.compile(optimizer='sgd', loss='mean_squared_error')

# Навчання моделі
model.fit(X_train, y_train, epochs=100, verbose=0)

# Збереження моделі
model.save('model.h5')
```

Для роботи у мобільних пристроях збережена модель потрібна конвертуватись у формат TensorFlow Lite. Програмний код цього наведено у лістингу 2.10.

Лістинг 2.10 – Конвертування моделі у формат TensorFlow Lite

```
import tensorflow as tf

# Завантаження навченої моделі TensorFlow
model = tf.keras.models.load_model('model.h5')

# Створення конвертера TensorFlow Lite
converter = tf.lite.TFLiteConverter.from_keras_model(model)

# Конвертування моделі у формат TensorFlow Lite
tflite_model = converter.convert()

# Збереження моделі у файл
with open('model.tflite', 'wb') as f:
    f.write(tflite_model)
```

Після конвертації та збереження моделі на основі штучного інтелекту у форматі .tflite можна розпочати етап її інтеграції у мобільний застосунок.

2.2.2 Інтеграція моделі у систему

Для впровадження моделі на основі штучного інтелекту в мобільний застосунок була використана бібліотека `tflite_flutter` версії 0.10.3. Цей плагін для Flutter надає гнучке та швидке рішення для доступу до інтерпретатора TensorFlow Lite та інтеграції моделі з системою. Його API схоже на TFLite Java та Swift API. Він прямо пов'язаний з TFLite C API, що робить його ефективним через низьку латентність [55].

Після навчання та збереження моделі у форматі .tflite необхідно додати файл цієї моделі у теку `assets` проєкту та додати відповідні залежності у файлі `pubspec.yaml`, щоб модель можна було використати.

У лістингу 2.11 наведено фрагмент програмного коду мовою Dart, що створює інтерпретатор, використовує модель TensorFlow Lite для необхідних обчислень та закриває її для оптимального використання ресурсів мобільного пристрою.

Лістинг 2.11 – Використання моделі TensorFlow Lite

```
DatabaseController databaseController = DatabaseController();

final interpreter = await
Interpreter.fromAsset('assets/model.tflite');

List<AppTransaction> transactions = await
databaseController.fetchTransactions();
// input: List<Object>
var inputData = transactions;

List<AppTransaction> transactionsOutput = [];

// output: Map<int, Object>
var outputData = {0: transactionsOutput};

interpreter.runForMultipleInputs(inputData, outputData);

interpreter.close();
```

2.3 Висновки за розділом 2

У другому розділі при розробці мобільного застосунку був реалізований архітектурний шаблон MVC, підключена система керування базами даних SQLite, а саме бібліотека sqflite, що оптимізована для використання з фреймворком Flutter. Були розроблені віджети представлення для перегляду історії транзакцій, додавання нової транзакції та перегляду результатів прогнозів у вигляді графіків за допомогою бібліотеки fl_chart.

При розробці моделі на основі штучного інтелекту було представлено процес тренування на прикладі лінійної регресії. Після навчання модель була збережена у форматі TensorFlow Lite для подальшої інтеграції з мобільним застосунком, використовуючи інтерпретатор з бібліотеки tflite_flutter, що була розроблена для використання з фреймворком Flutter.

3 ДОСЛІДЖЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Дослідження моделі на основі штучного інтелекту

Для ефективного тренування моделі необхідно підготувати велику кількість даних та обчислювальних потужностей. Для задачі індивідуального управління витратами та відповідного прогнозування треба мати дані транзакцій, наприклад з банків. Але ця інформація конфіденційна, що не представляється навіть в анонімному вигляді. Тому для навчання моделі на основі штучного інтелекту було використано згенеровані реалістичні, але фальшиві дані, що відображають доходи та витрати людини.

На рисунках 3.1-3.2 зображені візуалізовані за допомогою графіків доходи та витрати протягом десяти років з 2013 по 2023, що були згенеровані для випробування роботи комп'ютерної системи.

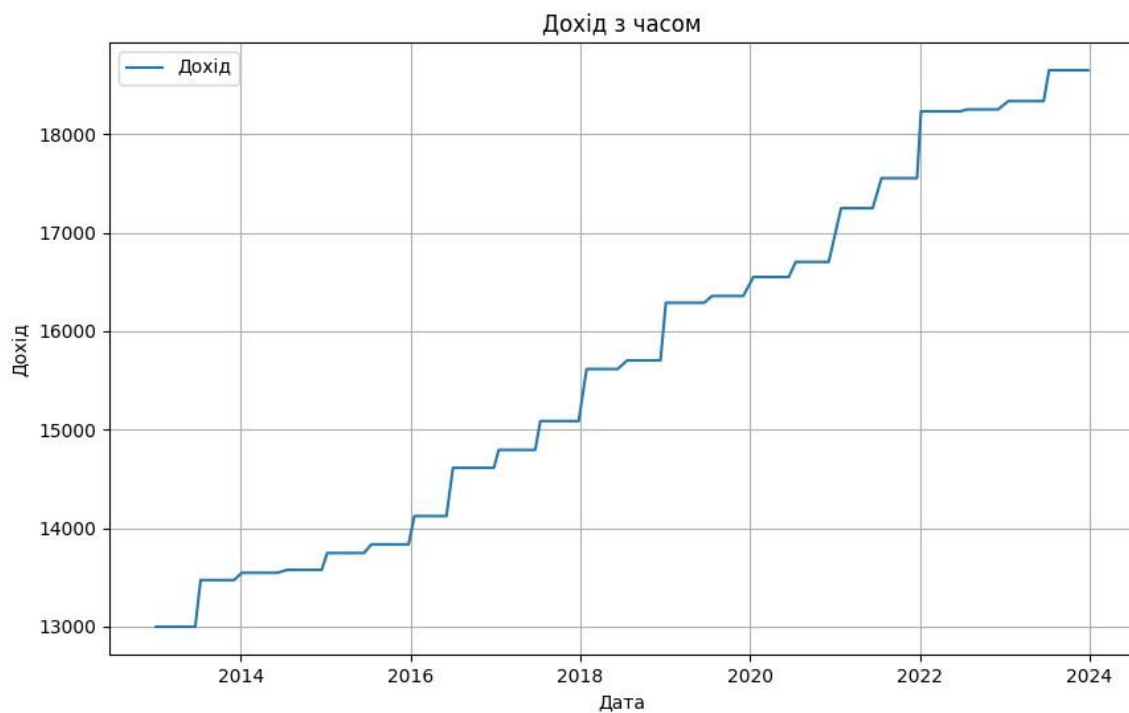


Рисунок 3.1 – Графік доходу з часом



Рисунок 3.2 – Графік витрат з часом

Створимо модель машинного навчання на основі лінійної регресії та спробуємо отримати прогнози відносно вхідних даних, що ми визначили раніше. На рисунку 3.3 зображені ці прогнози.

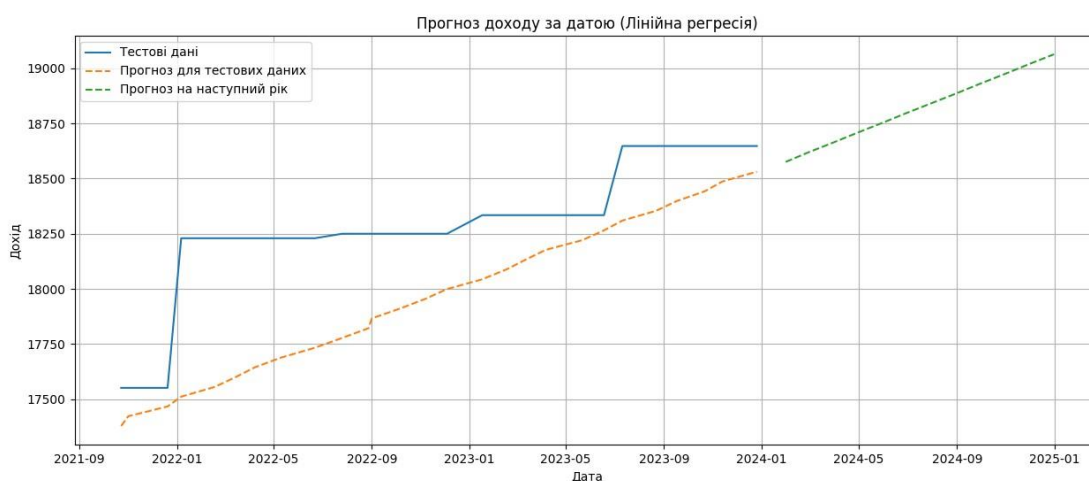


Рисунок 3.3 – Прогноз при використанні лінійної регресії

Останнім значенням доходу за місяць є 18647,50 ₴. При прогнозуванні наступного місяця отримано 18575,50 ₴.

Наступним способом для передбачення є метод опорних векторів. Він зображений на рисунку 3.4 з використанням лінійного ядра.

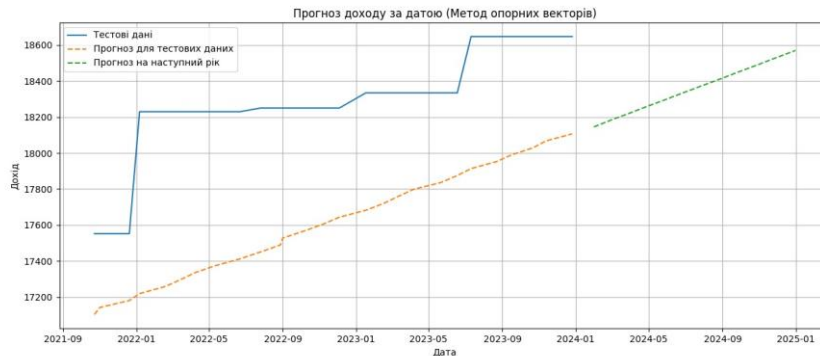


Рисунок 3.4 – Прогноз при застосуванні методу опорних векторів (лінійне ядро)

Як видно з графіку лінійної регресії прогноз будується за лінійним законом на основі тестових даних та продовжується як прогноз на наступний рік, за якими даних нема. Лінійне ядро методу опорних векторів має лінійну залежність, що і продемонстровано на рисунку, але у порівнянні з попереднім графіком дає нижчі значення прогнозу (18145,22 ₴) та відповідно має нижчу точність. Розглянемо інші параметри, а саме поліноміальне ядро та радіальне базисне ядро. Графіки з використанням цих параметрів зображені на рисунках 3.5-3.6.

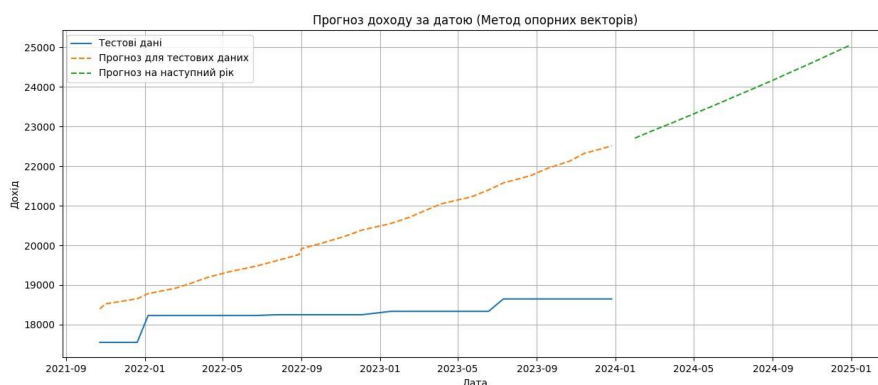


Рисунок 3.5 – Прогноз при застосуванні методу опорних векторів (поліноміальне ядро)

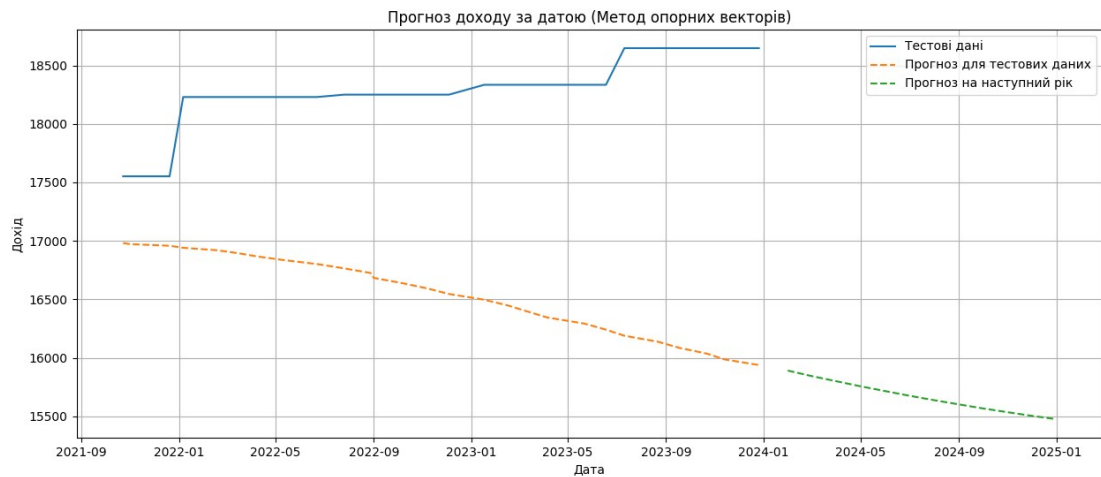


Рисунок 3.6 – Прогноз при застосуванні методу опорних векторів (радіальне базисне ядро)

При використанні поліноміального ядра наступний прогноз дорівнює 22704,69 ₴, а при використанні радіального базисного складає 15891,22 ₴.

Як видно з графіків та результатів прогнозування отримані значення дуже далекі від доходу останнього місяця (18647,50 ₴). Через що можна зробити висновок, що метод опорних векторів для прогнозування не підходить.

Наступним підходом, що можна використати є RNN. На рисунку 3.7 зображено прогнозування з використанням рекурентних нейронних мереж.

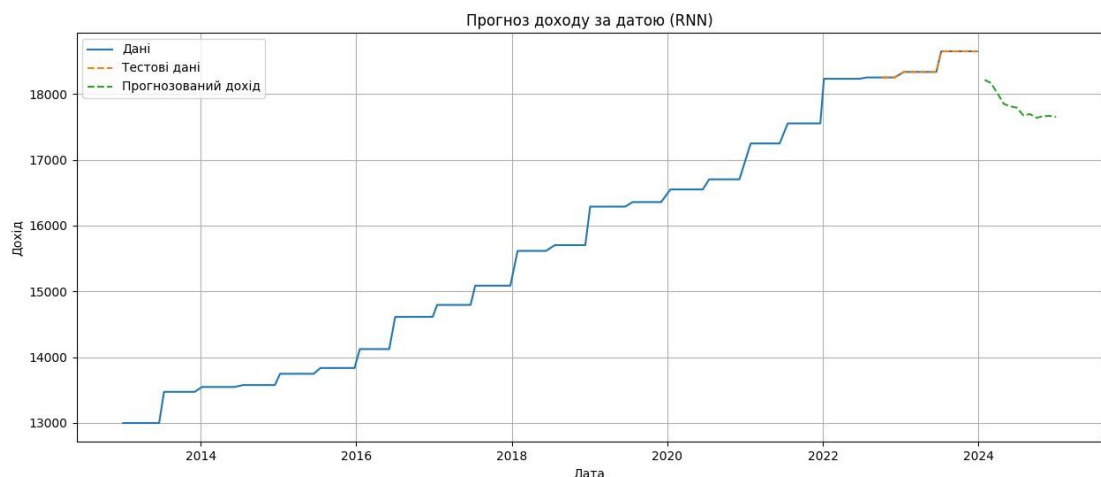


Рисунок 3.7 – Прогноз при застосуванні RNN

Передбачення на наступний місяць має значення 18212,22 ₴, що не є надто маленьким відносно доходу останнього місяця 18647,50 ₴, але прогноз на наступний рік має тенденцію на спад, що не можна вважати точним передбаченням.

Хоч RNN показала незадовільний результат, але її архітектура LSTM може показати перспективні результати. Бо ця нейронна мережа краще пристосована до задач з прогнозування часових рядів.

Прогноз LSTM на наступний місяць становить 18780,30 ₴, що не сильно відрізняється від попереднього місяця, але найголовніше прогноз на рік має тенденцію до росту. Через що цей прогноз можна вважати правдоподібним.

На рисунку 3.8 зображено прогноз доходу при використанні LSTM.

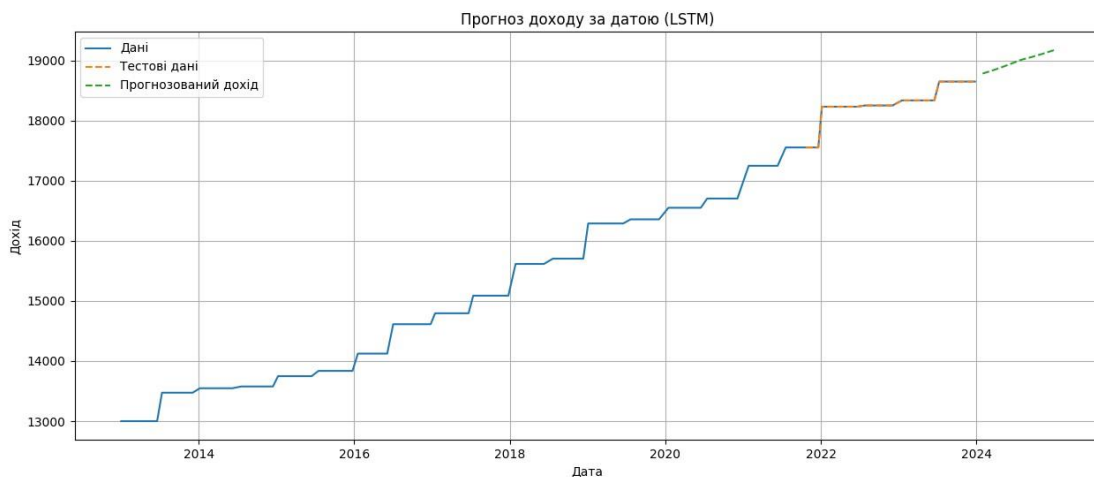


Рисунок 3.8 – Прогноз доходу при застосуванні LSTM

Для аналізу отриманих результатів необхідно подивитись на значення середньоквадратичної похибки при кожній епосі (крок навчання). Графік цієї залежності наведено на рисунку 3.9.

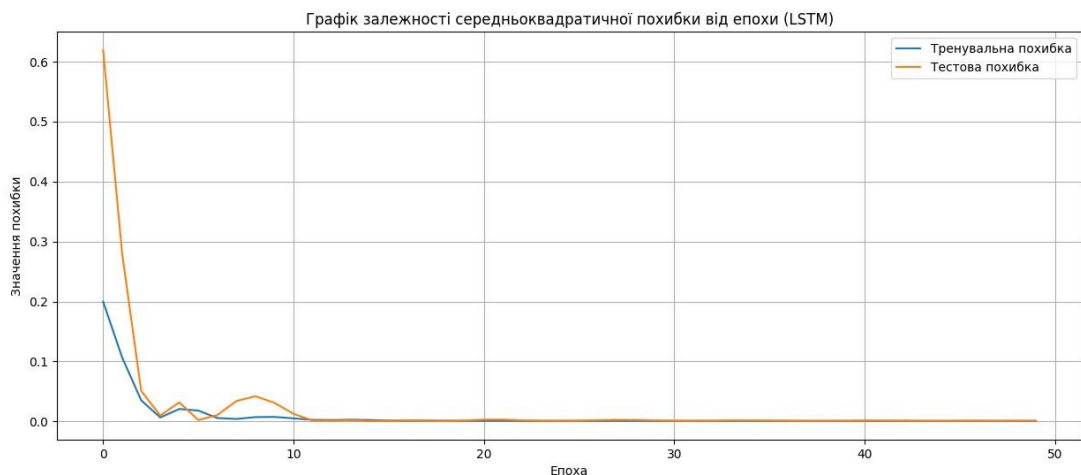


Рисунок 3.9 – Середньоквадратична похибка при прогнозуванні доходів LSTM

Як видно з графіку значення похибки досягає значення 0.01 до десяти епох, а далі наближається до нуля, але не досягає його. З цього можна зробити висновок, що після 15 епох тренування модель машинного навчання на основі LSTM вже має достатню навченість для ефективного прогнозування наданих вхідних даних.

Тепер розглянемо використання LSTM при прогнозуванні витрат, які були визначені раніше. Цей прогноз зображено на рисунку 3.10.

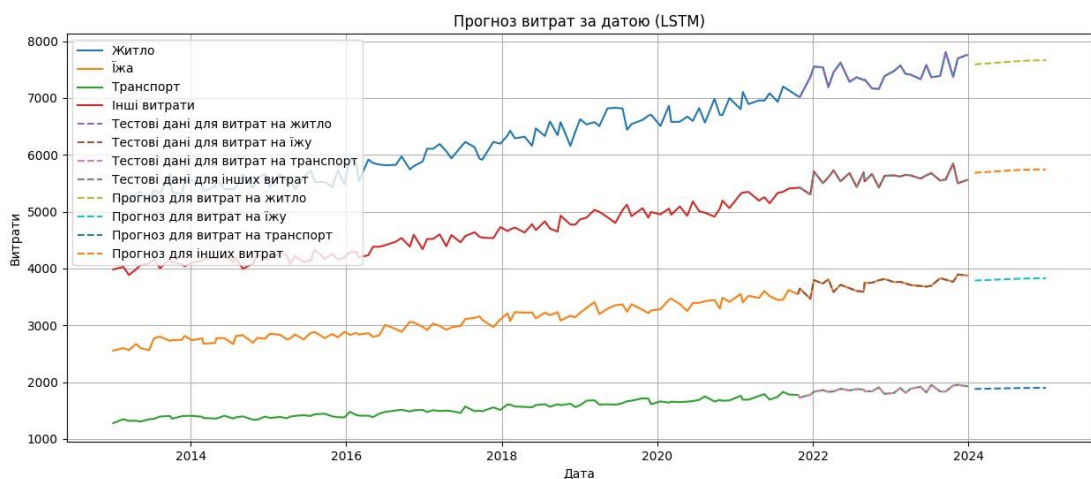


Рисунок 3.10 – Прогноз витрат при застосуванні LSTM

Прогноз на наступний місяць:

- витрати на житло 7591.17 ₴ (останній місяць 7756,82 ₴);
- витрати на їжу 3788.15 ₴ (останній місяць 3877,74 ₴);
- витрати на транспорт 1881.17 ₴ (останній місяць 1930,04 ₴);
- інші витрати 5684.70 ₴ (останній місяць 5556,21 ₴).

Як ми бачимо з графіку та результатів прогнозування результати є правдоподібними, але задача передбачення на цілий рік не дає чітких представлень, а лише обмежується майже лінійною залежністю.

Через це можна зробити висновок, що LSTM підходить для прогнозування витрат у наступному місяці, але не дає корисних результатів для довгострокового результату.

Також треба обрахувати середньоквадратичну похибку. На рисунку 3.11 наведено графік залежності похибки від епохи для витрат при використанні для прогнозування LSTM.

Як видно з графіку похибка зменшується до 15 епохи до 0.01, що схоже на результат отриманий при навчанні на даних доходу.

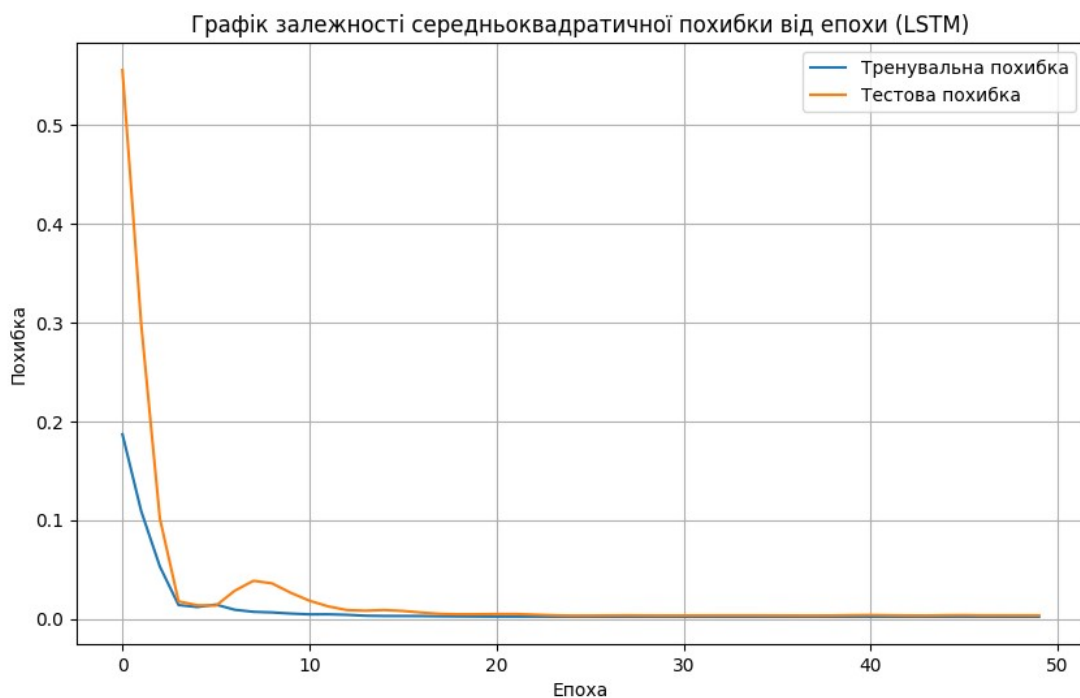


Рисунок 3.11 – Середньоквадратична похибка при прогнозуванні витрат LSTM

Зробимо ще тест для LSTM в умовах випадкових даних про витрати. Значення витрат останнього місяця дорівнюють 360,94 ₴, тоді як прогноз показує 350,59 ₴ для наступного місяця та тенденцію до росту, що є досить правдоподібним результатом та співвідноситься до попередніх передбачень. На рисунку 3.12 зображено цей прогноз для LSTM.



Рисунок 3.12 – Прогнозування витрат випадкових даних при використанні LSTM

3.2 Випробування комп'ютерної системи

За результатами дослідження моделі на основі штучного інтелекту у попередньому розділі для прогнозування витрат у комп'ютерній системі краще за все використовувати модель TensorFlow Lite, що використовує LSTM.

Для випробування використаємо дані для доходів та витрат, що були представлені раніше. На рисунку 3.13 зображено головний екран застосунку, на якому є можливість ввести транзакцію, а також показані згідно обчислень локальної моделі на основі штучного інтелекту прогнози на наступний місяць.

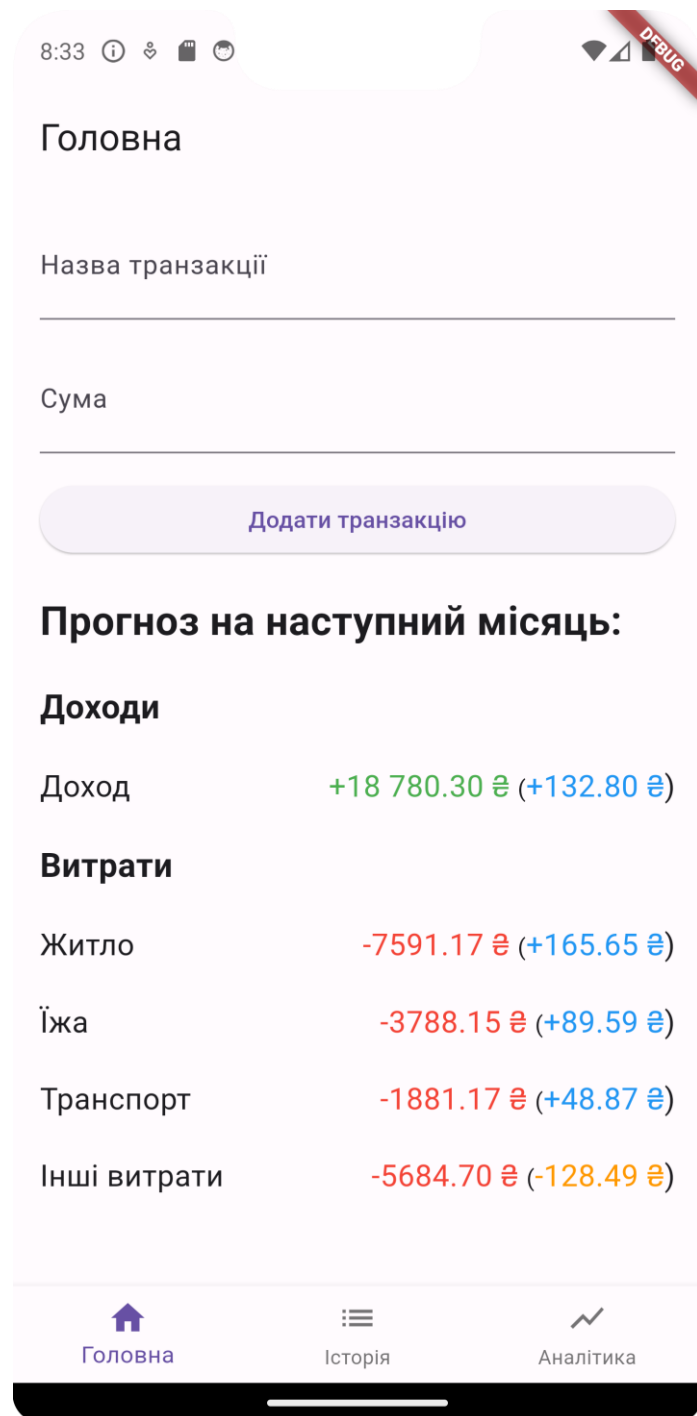


Рисунок 3.13 – Головна сторінка застосунку

Значення доходу відображені зеленим кольором, у той час, як витрати червоним, при цьому якщо прогноз вигідний для користувача, то він має голубий колір, а якщо поганий, то помаранчевий. Прогнозовані результати відповідають очікуванням та вважаються вірогідними.

На рисунку 3.14 зображено екран мобільного застосунку, що представляє графік прогнозу.

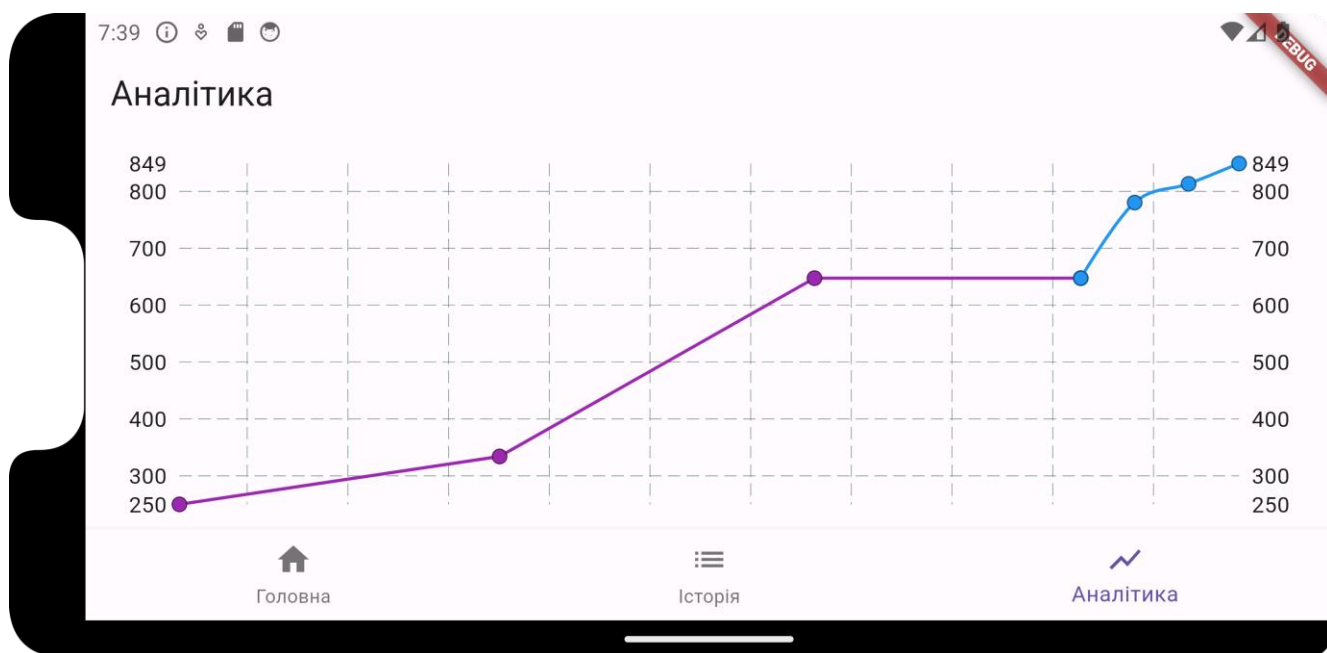


Рисунок 3.14 – Графік прогнозу

Фіолетовим кольором позначено графік, що побудований згідно з даними доходів, які зберігаються в локальній базі даних. Голубим кольором позначенні чотири значення: поточний місяць та прогнозовані значення на три наступних. Історія за минулі транзакції та передбачені результати, що представлені на графіку відповідають очікуванням.

Для вимірювання ефективності роботи комп'ютерної системи було проведено заміри для визначення необхідної кількості часу для обчислення прогнозу. Також були проведені подібні вимірювання для схожих застосунків, щоб порівняти їх між собою за критерієм часу виконання програми. Через те що не було знайдено подібного мобільного застосунку персонального управління витрат з архітектурним рішенням у вигляді розміщення моделі на основі штучного інтелекту на пристрої, то лише є змога порівняти зі схожими застосунками, але які замість обчислень локально відправляють дані на сервер [56].

На рисунку 3.15 зображено порівняння мобільного застосунку з аналогом за втраченим часом на обчислення прогнозу.

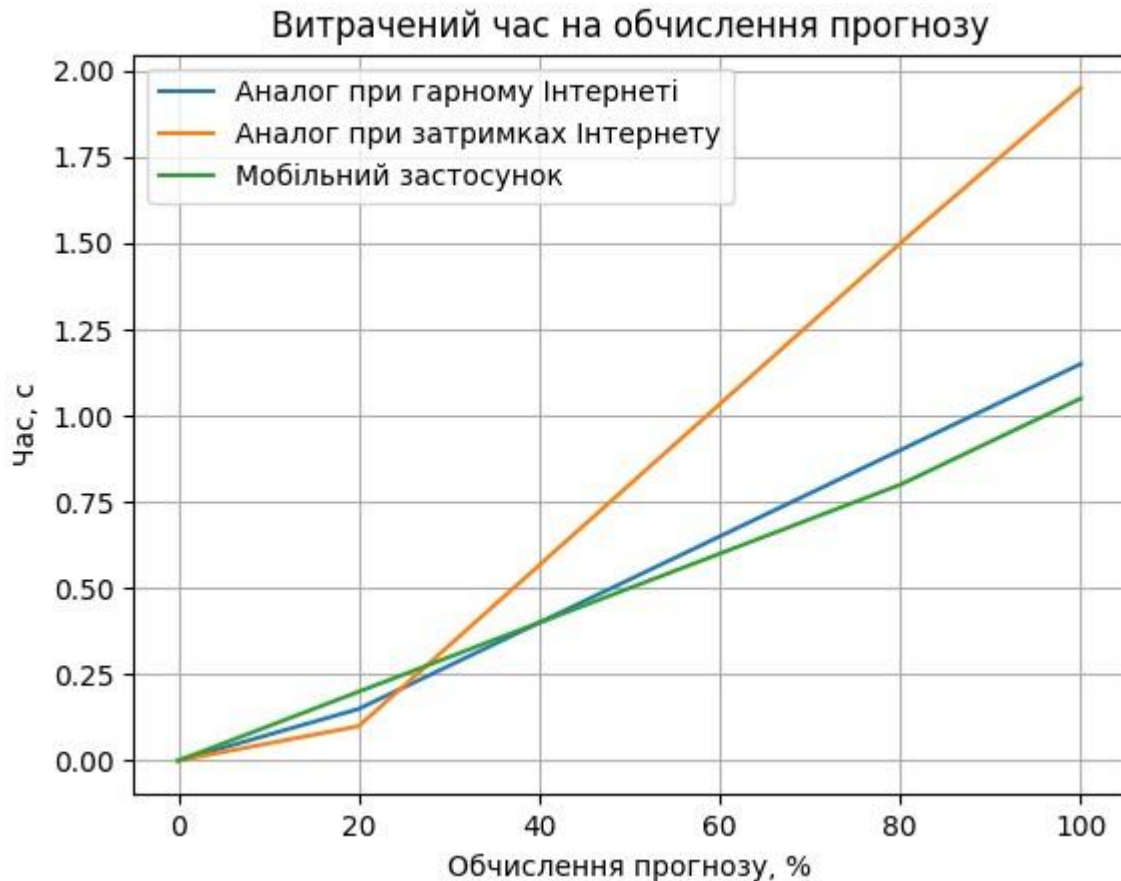


Рисунок 3.15 – Витрачений час на обчислення прогнозу

Зеленим кольором на графіку позначено розроблений мобільний застосунок, синім кольором аналог при сприятливих умовах в мережі та помаранчевим випадок коли існують затримки в мережі Інтернет. З цього графіку можна визначити те, що розроблена комп'ютерна система робить обчислення прогнозу на 0,1 с швидше ніж наведений аналог, але це не значить, що реальні обчислення виходять швидше. Через обмеженість апаратних ресурсів мобільної платформи застосунок не може проводити розрахунки швидше ніж сервер, але через те що дані доводиться пересилати мережею Інтернет, то втрачається час за який комп'ютерна система з локальною моделлю на основі штучного інтелекту стає ефективнішою.

Також очевидним є той факт, що при відсутності доступу до мережі Інтернет взагалі, то такі обчислення неможливі в принципі.

3.3 Висновки за розділом 3

У третьому розділі було проведено дослідження моделі на основі штучного інтелекту, що показало ефективні та слабкі методи для прогнозування. Так було виявлено, що метод опорних векторів при всіх параметрах робить прогнозування неточним та у випадку з поліноміальним та радіально базисним ядром протилежну тенденцію росту передбачень. Також лінійна регресія через свою простоту та RNN через помилкові очікування майбутніх значень не є гарними методами машинного навчання для прогнозування, але LSTM (архітектура рекурентної нейронної мережі) виявилася дуже ефективною мережею для прогнозування майбутніх значень часових рядів.

Після випробування комп'ютерної системи було виявлено, що значення прогнозу розраховуються згідно з очікуваними значеннями, що показує правильність та правдивість роботи застосунку. Також було обчислено час виконання програми при прогнозуванні та порівняння з аналогом, який має інше архітектурне рішення в розміщенні моделі на основі штучного інтелекту, а саме на сервері, тобто всі обчислення проходять там, а мобільний пристрій лише надсилає та отримує запити. Порівняння по критерію час обчислення прогнозу показало, що комп'ютерна система швидше на 0.1 с (8,7%) ніж аналог при найкращих умовах. А при відсутності доступу до мережі Інтернет застосунок має абсолютну перевагу над аналогами, які не мають змоги отримати прогноз у таких умовах.

ВИСНОВКИ

Дипломний проєкт (робота) був виконаний згідно з визначеним завданням. А саме була розроблена та досліджена комп'ютерна система для управління витратами на основі штучного інтелекту.

Під час виконання роботи було розглянуто поняття управління витратами, виявлені основні способи мобільної розробки та був обраний кросплатформний підхід. Було проаналізовано мови програмування, фреймворки для розробки та штучного інтелекту. Була обрана мова Dart, фреймворк Flutter та TensorFlow Lite. Був проведений аналіз використання мови Python та методів ML.

Була проведена розробка застосунку в якій було впроваджено архітектурний шаблон MVC, під'єднано систему керування базами даних SQLite та розроблені віджети мобільного додатку. Було проведено тренування та інтеграція у комп'ютерну систему моделі на основі штучного інтелекту.

Було проведено дослідження методів машинного навчання, яке показало перевагу LSTM у прогнозованні часових рядів над іншими підходами. Було проведено випробування системи, яке підтвердило відповідність всіх вимог визначеного завдання.

Науковою новизною проєкту (роботи) є розробка та тренування власної моделі на основі штучного інтелекту з використанням LSTM, що може бути розміщена на мобільних пристроях.

Практична цінність проєкту (роботи) полягає у реалізації комп'ютерної системи на основі моделі, яка дозволяє покращити конфіденційність, швидкодію на 8,7%, не вимагає підключення до мережі Інтернет та адаптує модель на основі штучного інтелекту до даних користувача у порівнянні з аналогами.

В майбутньому розроблена комп'ютерна система при достатній навченості моделі та збільшенні апаратних ресурсів мобільних пристроїв стане альтернативою для застосунків, які мають модель на основі штучного інтелекту на сервері.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 What is Expense Management, and How is it Done? | Fi.Money. *Fi.Money*. URL: <https://fi.money/blog/posts/what-is-expense-management-and-how-is-it-done> (date of access: 23.11.2023).

2 Mobile Operating System Market Share Worldwide | Statcounter Global Stats. *StatCounter Global Stats*. URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (date of access: 23.11.2023).

3 Google. URL: <https://www.google.com/> (date of access: 23.11.2023).

4 Android | Do More With Google on Android Phones & Devices. URL: <https://www.android.com/> (date of access: 23.11.2023).

5 Download Android Studio & App Tools - Android Developers. *Android Developers*. URL: <https://developer.android.com/studio> (date of access: 23.11.2023).

6 Android Apps on Google Play. URL: <https://play.google.com/> (date of access: 23.11.2023).

7 Apple. URL: <https://www.apple.com/> (date of access: 23.11.2023).

8 iOS 17. *Apple*. URL: <https://www.apple.com/ios> (date of access: 23.11.2023).

9 Xcode 15 - Apple Developer. *Apple Developer*. URL: <https://developer.apple.com/xcode/> (date of access: 23.11.2023).

10 App Store. *Apple*. URL: <https://www.apple.com/app-store/> (date of access: 23.11.2023).

11 What is cross-platform mobile development? | Kotlin. *Kotlin Help*. URL: <https://kotlinlang.org/docs/cross-platform-mobile-development.html#conclusion> (date of access: 23.11.2023).

12 Переваги та Недоліки Кросплатформної та Нативної Розробки Мобільних Додатків - Merehead. *Merehead*. URL: <https://merehead.com/ua/blog/cross-platform-native-mobile-development/> (дата звернення: 23.11.2023).

13 Benefits of Cross-Platform App Development | Linkup Studio. *Linkup Studio*. URL: <https://linkupst.com/blog/advantages-of-cross-platform-development/> (date of access: 23.11.2023).

14 Oracle | Cloud Applications and Cloud Platform. URL: <https://www.oracle.com/> (date of access: 23.11.2023).

15 Java | Oracle. URL: <https://www.java.com/> (date of access: 23.11.2023).

16 Java in App Development: Why Choose It for Mobile Applications | EPAM Startups & SMBs. *EPAM Anywhere: Work On Your Own Terms | EPAM Anywhere*. URL: <https://anywhere.epam.com/business/java-for-mobile-app-development> (date of access: 23.11.2023).

17 Most used languages among software developers globally 2023 | Statista. *Statista*. URL: <https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/> (date of access: 23.11.2023).

18 JetBrains: Essential tools for software developers and teams. *JetBrains*. URL: <https://www.jetbrains.com/> (date of access: 23.11.2023).

19 Kotlin Programming Language. *Kotlin*. URL: <https://kotlinlang.org/> (date of access: 23.11.2023).

20 Objective-C Runtime | Apple Developer Documentation. *Apple Developer Documentation*. URL: <https://developer.apple.com/documentation/objectivec> (date of access: 23.11.2023).

21 Swift - Apple Developer. *Apple Developer*. URL: <https://developer.apple.com/swift/> (date of access: 23.11.2023).

22 Dart programming language | Dart. URL: <https://dart.dev/> (date of access: 23.11.2023).

23 Get started with Xamarin - Xamarin. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/xamarin/get-started/> (date of access: 23.11.2023).

24 C# docs - get started, tutorials, reference. *Microsoft Learn: Build skills that open doors in your career*. URL: <https://learn.microsoft.com/en-us/dotnet/csharp/> (date of access: 23.11.2023).

25 Meta | Social Metaverse Company. URL: <https://about.meta.com/> (date of access: 23.11.2023).

26 React Native Learn once, write anywhere. URL: <https://reactnative.dev/> (date of access: 23.11.2023).

27 DevDocs – JavaScript documentation. *DevDocs API Documentation*. URL: <https://devdocs.io/javascript/> (date of access: 23.11.2023).

28 Flutter - Build apps for any screen. URL: <https://flutter.dev/> (date of access: 23.11.2023).

29 Ivonchik K. Flutter AI Integration: Redefining Mobile Apps. *WTF Blog*. URL: <https://blog.flutter.wtf/flutter-ai-integration/> (date of access: 23.11.2023).

30 Top Flutter Machine Learning, Deep Learning, Computer Vision (OCR, Face Detection), MLKit, Tensorflow packages | Flutter Gems. *Flutter Gems - A Curated List of Top Dart and Flutter packages*. URL: <https://fluttergems.dev/machine-learning/> (date of access: 23.11.2023).

31 ML Kit for Firebase. *Firebase*. URL: <https://firebase.google.com/docs/ml-kit> (date of access: 23.11.2023).

32 Home. *PyTorch*. URL: <https://pytorch.org/mobile/home/> (date of access: 23.11.2023).

33 TensorFlow Lite | ML for Mobile and Edge Devices. *TensorFlow*. URL: <https://www.tensorflow.org/lite> (date of access: 23.11.2023).

34 Welcome to Python.org. *Python.org*. URL: <https://www.python.org/> (date of access: 23.11.2023).

35 Project Jupyter | Home. *Project Jupyter*. URL: <https://jupyter.org/> (date of access: 23.11.2023).

36 Machine Learning Tutorial. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/machine-learning/?ref=lbp> (date of access: 23.11.2023).

37 VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data / ed. by D. Schanafelt. O'Reilly Media, 2016. 548 p.

38 Harrison M. Machine Learning: Pocket Reference. O'Reilly Media, 2019. 200 p.

39 Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, 2016.

40 Bengfort B., Bilbro R., Ojeda T. Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning / ed. by N. Tache. Beijing : O'Reilly Media, 2018. 332 p.

41 Cielen D., Meysman A., Ali M. Introducing Data Science: Big Data, Machine Learning, and more, using Python tools. Manning Publications, 2016. 320 p.

42 Géron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2017. 574 p.

43 Grus J. Data science from scratch: First principles with Python. 2015. 311 p.

44 MVC Design Pattern - GeeksforGeeks. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/mvc-design-pattern/> (date of access: 23.11.2023).

45 What is Database? - GeeksforGeeks. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/what-is-database/> (date of access: 23.11.2023).

46 sqflite | Flutter Package. *Dart packages*. URL: <https://pub.dev/packages/sqflite> (date of access: 23.11.2023).

47 ISO - ISO 8601 – Date and time format. *ISO*. URL: <https://www.iso.org/iso-8601-date-and-time-format.html> (date of access: 23.11.2023).

48 path | Dart Package. *Dart packages*. URL: <https://pub.dev/packages/path> (date of access: 23.11.2023).

49 Widget class - widgets library - Dart API. *Flutter - Dart API docs*. URL: <https://api.flutter.dev/flutter/widgets/Widget-class.html> (date of access: 23.11.2023).

50 intl | Dart Package. *Dart packages*. URL: <https://pub.dev/packages/intl> (date of access: 23.11.2023).

51 fl_chart | Flutter Package. *Dart packages*. URL: https://pub.dev/packages/fl_chart (date of access: 23.11.2023).

52 NumPy. URL: <https://numpy.org/> (date of access: 23.11.2023).

53 pandas - Python Data Analysis Library. URL: <https://pandas.pydata.org/> (date of access: 23.11.2023).

54 Matplotlib – Visualization with Python. URL: <https://matplotlib.org/> (date of access: 23.11.2023).

55 tflite_flutter | Flutter Package. *Dart packages*. URL: https://pub.dev/packages/tflite_flutter (date of access: 23.11.2023).

56 How to Develop AI-Based Expense Management App like Cleo, Fyle?. *Offshore IT Services | IT Software Solutions Company - Octal IT Solution*. URL: <https://www.octalsoftware.com/blog/ai-based-expense-management-app-development> (date of access: 23.11.2023).