Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська дипломна робота на тему:

ПРОГРАМНИЙ WEB-СЕРВІС ПРОГНОЗУВАННЯ ОПТИМАЛЬНИХ КРОКІВ ДЛЯ ДОСЯГНЕННЯ МЕТИ. ЧАСТИНА 2. СЕРВЕРНА ЧАСТИНА

Виконав: студент 4 курсу, групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Прудивус О.С.
(прізвище та ініціали)

Керівник: к.т.н., проф. каф. КН
Колесницький О. К.
(прізвище та ініціали)
«07» 06 2024 р.

Рецензент: к.т.н., проф., каф. КСУ
Биков М. М.
(прізвище та ініціали)
«07» 06 2024 р.

Допущено до захисту
Зав. кафедри проф., д.т.н. Яровий А.А.
«07» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А. А.
«07» 03 2024 р

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Прудивусу Олександру Сергійовичу

1. Тема роботи: Програмний WEB-сервіс прогнозування оптимальних кроків для досягнення мети. Частина 2. Серверна частина.
Керівник роботи: Колесницький Олег Костянтинович, к.т.н., професор.
Затверджені наказом вищого навчального закладу "11" 03 2024 року №80
2. Строк подання студентом роботи 27 05 2024
3. Вихідні дані до роботи:
кількість аналізованих параметрів досяжності мети – 15, обсяг навчальної вибірки – не менше 1000, обсяг тестової вибірки – не менше 200,мова програмування Python; середовище розробки PyCharm 2024.1.1
4. Зміст текстової частини (перелік питань, які потрібно розробити):
вступ, аналіз предметної web-сервісів прогнозування оптимальних кроків для досягнення мети, проектування серверної частини та модулю класифікації досяжності мети web-сервісу прогнозування оптимальних кроків для досягнення метипрограмна реалізація модулю класифікації та серверної частини, програмна реалізація серверної частини та модулю класифікації досяжності мети web-сервісу прогнозування оптимальних кроків для досягнення мети, тестування та аналіз результатів роботи серверної частини та модуля класифікації досяжності мети web-сервісу прогнозування оптимальних кроків для досягнення мети, висновки, список використаної літератури
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Структурна схема архітектури WEB-сервісу, Схема алгоритму роботи модуля класифікатора визначення досяжності мети, UML-діаграма класів серверної частини WEB-сервісу, UML-діаграма компонентів серверної частини WEB-сервісу, UML-діаграма розгортання серверної частини WEB-сервісу, UML-діаграма послідовності створення нової цілі, UML-діаграма активності серверної частини WEB-сервісу

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Колесницький О. К., к.т.н., професор		

7. Дата видачі завдання 07.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської дипломної роботи	Термін виконання		Примітка
		Початок	Закінчення	
1	Аналіз предметної області WEB-сервісів для прогнозування оптимальних кроків досягнення мети	06.05.24	10.05.24	
2	Постановка задачі	11.05.24	16.05.24	
3	Аналіз існуючих методів класифікації в машинному навчанні	17.05.24	20.05.24	
4	Вибір мови програмування та середовища розробки	21.05.24	24.05.24	
5	Розробка серверної частини WEB-сервісу для прогнозування оптимальних кроків досягнення мети.	25.05.24	29.05.24	
6	Розробка модуля класифікації досяжності мети	30.05.24	03.06.24	
7	Аналіз функціонування модуля класифікації досяжності мети	04.06.24	06.06.24	

Студента _____ **Прудивус О.С.**
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ **Колесницький О.К.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 69 сторінки формату А4, на яких є 18 рисунків, 4 таблиця, список використаної літератури містить 17 найменування.

Метою дослідження є підвищення достовірності визначення класу досяжності мети сервісу прогнозування оптимальних кроків досягнення мети за рахунок машинного навчання.

У роботі було обґрунтовано вибір методу машинного навчання Random Forest для модуля визначення класу досяжності мети накопичення. Розроблено алгоритм роботи модуля класифікації на основі обраного методу. Також було спроектовано серверну частину WEB-сервісу з використанням UML діаграм, що відображають структуру та поведінку системи.

Для програмної реалізації модуля класифікації було обрано мову програмування Python та бібліотеку Scikit-learn. Навчання та тестування модуля відбувалось з використанням набору даних, що містить 21153 екземплярів, серед яких 12831 є досяжними, а 8322 – недосяжними. Розроблений модуль класифікації продемонстрував достовірність класифікації на рівні 98%, перевершивши аналог на 3%. Також було програмно реалізовано серверну частину WEB-сервісу.

Ключові слова: класифікація, машинне навчання, серверна частина, Random Forest.

ABSTRACT

The bachelor's thesis consists of 69 A4 pages, with 18 figures, 4 tables, and a bibliography of 17 references.

The purpose of the study is to increase the reliability of determining the achievability class of a service goal by predicting the optimal steps to achieve the goal using machine learning.

The paper substantiates the choice of the Random Forest machine learning method for the module for determining the class of achievability of the accumulation goal. An algorithm for the classification module based on the selected method was developed. The server side of the WEB-service was also designed using UML diagrams that reflect the structure and behavior of the system.

The Python programming language and the Scikit-learn library were chosen for the software implementation of the classification module. The module was trained and tested using a dataset containing 21153 instances, of which 12831 are reachable and 8322 are unreachable. The developed classification module demonstrated 98% classification accuracy, outperforming the analogue by 3%. The server part of the WEB-service was also programmatically implemented.

Keywords: classification, machine learning, server side, Random Forest.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. У сучасному, динамічному суспільстві, кожен намагається реалізувати свої прагнення та досягти поставлених завдань. Для втілення цих амбіцій в життя, люди залучають широкий спектр засобів та можливостей, серед яких важливу роль відіграють інноваційні технології. Еволюція глобальної мережі Інтернет суттєво спростила процедуру пошуку готових рішень, на кшталт автоматизованих помічників та інтелектуальних агентів. Однак, не всі прагнення можливо здійснити за допомогою універсальних інструментів, особливо коли справа стосується накопичення фінансових ресурсів, адже існує велика кількість факторів що так чи інакше впливають на результат.

Наявність коштів є визначальним аспектом у реалізації численних намірів, починаючи від придбання сучасного гаджета і закінчуючи заснуванням власної справи. Саме тому виникає необхідність ефективного акумулювання необхідного капіталу. Для розв'язання цієї проблеми створюються спеціалізовані боти, віртуальні асистенти та онлайн-платформи для прогнозування оптимальних кроків накопичення.

Наявні спеціалізовані платформи мають свої сильні сторони, наприклад, широкий діапазон критеріїв для аналізу завдання. Проте, вони також мають певні недоліки, такі як значна вартість, обмеженість безкоштовних тарифів та використання елементарних алгоритмів, які не завжди пропонують практичні рекомендації.

WEB-сервіс, поданий у даній роботі, ґрунтується на методах машинного навчання і має на меті подолати зазначені обмеження, надаючи більш влучні результати для сприяння користувачам у досягненні їхніх фінансових цілей.

Для отримання результатів прогнозування, перш за все задане завдання має бути класифіковане як досяжне чи. Це є важливим аспектом для правильного функціонування WEB-сервісу як єдиного цілого.

Метою дослідження є підвищення достовірності визначення класу досяжності мети сервісу прогнозування оптимальних кроків для досягнення мети за рахунок машинного навчання.

Об’єктом дослідження є процес прогнозування оптимальних кроків для досягнення мети накопичення коштів.

Предметом дослідження є програмні засоби для прогнозування оптимальних кроків досягнення мети та достовірність визначення класу досяжності мети.

У ході виконання роботи необхідно виконати наступні задачі:

- Провести аналіз предметної області методів визначення класу досяжності мети та обрати напрямок дослідження.
- Розробити алгоритм роботи модуля визначення класу досяжності мети та серверної частини WEB-сервісу
- Програмно реалізувати модуль визначення класу досяжності мети та серверну частину WEB-сервісу
- Провести тестування розробленого програмного модуля.

Апробації: Робота апробувалася на конференції «Молодь в науці: дослідження, проблеми, перспективи» 2024 року, доповідь «Перспективи використання алгоритмів прогнозування оптимальних кроків для досягнення цілі» [1].

Публікації: За результатами бакалаврської дипломної роботи опубліковано тезу доповідь на конференції: «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)» (Вінниця, Україна) [1].

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ WEB-СЕРВІСІВ ПРОГНОЗУВАННЯ ОПТИМАЛЬНИХ КРОКІВ ДЛЯ ДОСЯГНЕННЯ МЕТИ

Для ефективного аналізу, розробки та забезпечення масштабованості, WEB-сервіс було розділено на дві основні частини: серверну і клієнтську. Окрім того, функціональність WEB-сервісу було розділено на два ключові модулі: модуль класифікації досяжності мети та модуль класифікації визначення доцільності витрат. Такий поділ дозволяє зосередитися на розробці та оптимізації кожного модуля окремо, що підвищує ефективність та якість роботи системи в цілому.

У рамках даної бакалаврської роботи основна увага приділяється розробці та реалізації модуля класифікації досяжності фінансових цілей, який є фундаментальною складовою для подальшого функціонування WEB-сервісу та взаємодії з модулем визначення доцільності витрат.

1.1. Постановка задачі

Отже, у даній роботі досліджується класифікатор, реалізований за допомогою методів машинного навчання, що здатен визначити клас досяжності мети для прогнозування оптимальних кроків досягнення цієї мети.

Таким чином задля отримання результатів прогнозування, перш за все задана мета має бути класифіковане як здійсненна чи нездійснена. Це є важливим аспектом для правильного функціонування WEB-сервісу як єдиного цілого. Для початку потрібно визначити вхідні данні, які будуть взаємодіяти з WEB-сервісу.

Вхідний набір даних містить 12831 прикладів для здійснених накопичень і 8322 нездійснених. Класи задаються числами від 0 до 1. Вхідні данні мають наступний вигляд:

1. Ціль накопичення (initial_goal): Грошова кількість що досягається.

2. Початкова дата (start_date): Початкова дата накопичення.
3. Кінцева дата (end_date): Кінцева дата накопичення.
4. Сума надходжень в місяць (monthly_income): Кількість надходжень щомісяця.
5. Сума, яку можна відкласти в місяць (monthly_savings): Кількість коштів що запропоновані на заощадження щомісяця.
6. Витрати на їжу (food_expenses): Кількість щомісячних витрат на харчування.
7. Витрати на подорожі (travel_expenses): Кількість щомісячні витрат на подорожів, громадський транспорт.
8. Витрати на відпочинок (entertainment_expenses): Кількість щомісячних витрат на відпочинок.
9. Витрати на інтернет (internet_expenses): Кількість щомісячних витрат на оплату інтернет трафіку.
10. Витрати на комунальні послуги (utility_expenses): Кількість щомісячних витрат на комунальні послуги (сумарно).
11. Витрати на мобільний (mobile_expenses): Кількість щомісячних витрат мобільний зв’язок.
12. Витрати на підписки на сервіси (subscription_expense): Кількість щомісячних витрат за підписки на сервіси.
13. Лікарняні витрати (healthcare_expenses): Кількість щомісячних витрат на догляд за здоров’ям (ліки, відвідування лікарів).
14. Витрати на освіту (education_expenses): Кількість щомісячних витрат на освіту.
15. Витрати на благодійність (charity_expenses): Кількість щомісячних витрат на благодійність.

Цільовим параметром є бінарна змінна, що набуває значення 0 або 1, де 0 відповідає негативному (нездійсненному) результату накопичення, а 1 – позитивному (здійсненному) результату.

Необхідно розробити класифікатор на основі методів машинного навчання, який дозволяє за набором вхідних ознак визначити клас здійсненності мети накопичення. Класифікатор повинен прогнозувати, чи є задана мета накопичення здійсненною або нездійсненною з урахуванням наданих вхідних параметрів.

1.2. Огляд відомих методів класифікацій

Задача класифікації полягає у віднесенні об'єкта до одного з попередньо визначених класів на основі його характеристик, ознак. Нехай $X = x_1, x_2, \dots, x_n$ – множина вхідних ознак, а $Y = y_1, y_2, \dots, y_m$ – множина класів. Завдання класифікації полягає в побудові моделі $f: X \rightarrow Y$, що для кожного об’єкта $x \in X$ визначає його клас $y \in Y$.

У нашому випадку відбувається бінарна класифікація, тобто множина класів Y складається з двох елементів $Y = 0,1$, де 0 відповідає негативному класу, 1 позитивному.

Для побудови моделі класифікації використовується навчальна вибірка, формула (1.1):

$$D = \left(x^i, y^i\right)_{i=1}^{N}, \tag{1.1}$$

де $x^i \in X$ – вхідні ознаки і-го прикладу, $y^i \in Y$ – відповідний клас, N – кількість прикладів у навчальній вибірці.

Після навчання модель може бути використана для прогнозування класу нових об'єктів. Існує багато різних методів класифікації, які використовуються в машинному навчанні для вирішення задач прогнозування. Розглянемо деякі з найбільш популярних і ефективних методів.

1.2.1 Дерева рішень (Decision Trees) [2]

Це ієрархічні моделі, які розбивають простір ознак на прямокутні регіони, використовуючи послідовність простих правил. Кожен вузол дерева представляє умову, яка перевіряється для вхідних даних, а листя представляють класи (рис. 1.1).

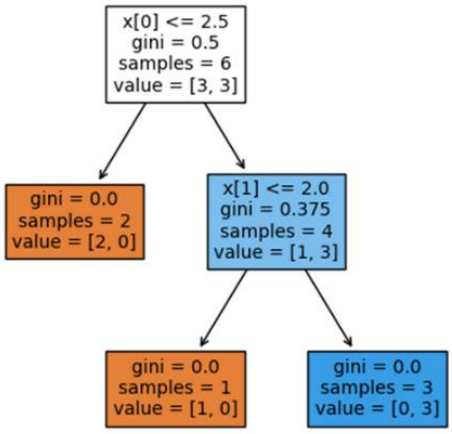


Рисунок 1.1 – Загальний вигляд дерева рішень.

Дерева рішень прості для розуміння та візуалізації, але часту піддаються перенавчанню. Метод дерев рішень доцільно використовувати в випадках:

- Необхідності отримати легко інтерпретовану модель.
- Якщо дані містять категоріальні або порядкові змінні.
- Коли, існує потреба в автоматичному відборі ознак.

1.2.2 Логістична регресія (Logistic Regression) [3]

Це статистичний метод, який використовується для бінарної класифікації. Він моделює ймовірність приналежності об'єкта до певного класу на основі лінійної комбінації вхідних ознак. Логістична регресія проста в реалізації та інтерпретації, але може не враховувати складні нелінійні залежності між ознаками. Метод Логістичної регресії доцільно використовувати в випадках:

- Необхідності отримати ймовірнісну оцінку приналежності об'єкта до класу.
- Існування лінійної залежності між ознаками та цільовою змінною.

– Необхідності проаналізувати вплив окремих ознак на результат класифікації.

1.2.3 Випадковий ліс (Random Forest) [4]

Це ансамблевий метод, який поєднує множину дерев рішень. Ансамблевий означає, такий який поєднує прогнози декількох базових моделей для отримання більш точного та надійного результату. Кожне дерево навчається на випадковій підмножині вхідних ознак і прикладів. Фінальне рішення приймається шляхом голосування серед усіх дерев (рис 1.1).

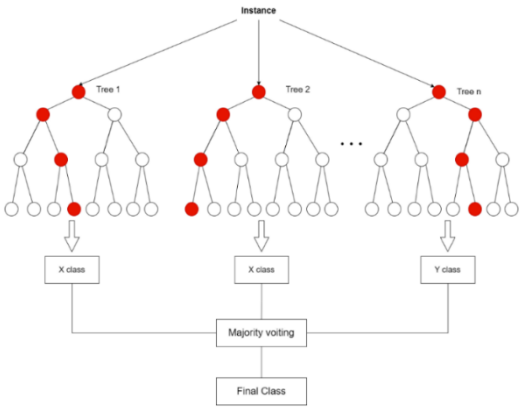


Рисунок 1.2 – Загальний вигляд методу Random Forest.

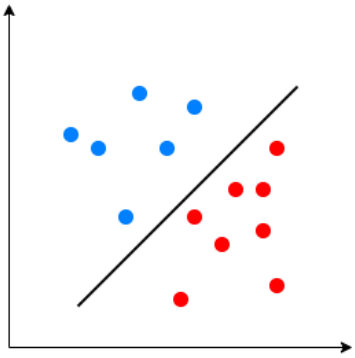
Даний метод здатний моделювати складні нелінійні залежності та добре працює з великою кількістю ознак. Метод випадкового лісу доцільно використовувати в випадках, якщо:

– Дані містять складні нелінійні залежності.
– Присутня висока розмірність вхідного простору ознак.
– Потрібно зменшити ризик перенавчання.

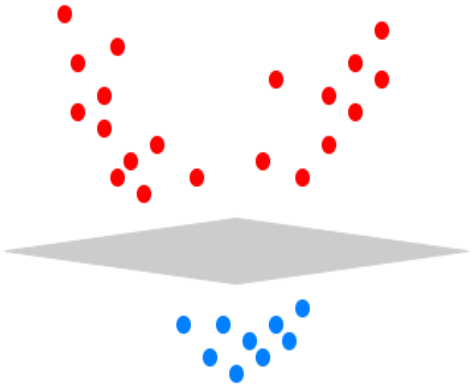
1.2.4 Метод опорних векторів (Support Vector Machines) [5]

Метод опорних векторів знаходить гіперплощину, яка максимально розділяє класи в просторі ознак. У машинному навчанні гіперплощиною

називають лінійну границю рішень, розмірність якої на одиницю менша за розмірність вхідних даних. У двовимірному просторі гіперплощина це лінія (рис. 1.1), а у тривимірному це площина (рис. 1.2), в усіх подальших вимірах візуалізація гіперплощини є важким завданням, але вона все одно існує та дорівнює N-1.



Рисунки 1.3 – Загальний вигляд гіперплощини в двовимірному просторі.



Рисунки 1.4 – Загальний вигляд гіперплощини в тривимірному просторі.

Даний підхід може використовувати ядрові функції для перетворення простору ознак у простір вищої розмірності, де класи лінійно роздільні. Також метод добре працює з малою кількістю прикладів і здатен знаходити складні границі рішень. Метод опорних векторів доцільно використовувати в випадках, якщо:

- Дані лінійно нероздільні у вхідному просторі ознак.
- Наявна мала кількість навчальних прикладів.

- Необхідно отримати чіткі границі рішень.

1.2.5 Наївний баєсівський класифікатор (Naive Bayes) [6]

Це ймовірнісний метод, який базується на теоремі Баєса. Він припускає умовну незалежність між ознаками за умови заданого класу.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}, \tag{1.2}$$

Де $P(A \mid B)$ – умовна ймовірність події A за умови, що подія B відбулася, $P(B \mid A)$ – умовна ймовірність події B, за умови, що подія A відбулася, $P(A)$ – апріорна ймовірність події A. $P(B)$ – ймовірність події B.

Наївний баєсівський класифікатор простий в реалізації та швидко навчається, але може не враховувати складні залежності між ознаками. Метод наївної баєсівської класифікації доцільно використовувати в випадках:

- За умови що ознаки умовно незалежні.
- При необхідності швидко навчити модель.
- При потребі врахувати апріорні ймовірності класів.

1.2.6 Нейронні мережі (Neural Networks) [7]

Це моделі, що базуються на біологічних нейронних мережах. Вони складаються з взаємопов'язаних вузлів – нейронів, організованих у шари. Такі мережі здатні моделювати складні нелінійні залежності та добре працюють з великою кількістю прикладів. Найпоширеніші види нейронних мереж:

- Багатошаровий перцептрон (Multi-Layer Perceptron, MLP).
- Згорткові нейронні мережі (Convolutional Neural Networks, CNN).
- Рекурентні нейронні мережі (Recurrent Neural Networks, RNN).

Однак вони можуть бути затратними для обчислень. Нейроні мережі доцільно використовувати в випадках, якщо:

- Дані містять складні нелінійні залежності.
- Наявний великий обсяг навчальних даних.
- Існує необхідність автоматично виявляти ієрархічні представлення ознак.

Отже, вибір конкретного методу класифікації залежить від специфіки задачі, властивостей даних, вимог до інтерпретації моделі та наявних обчислювальних ресурсів.

1.3. Обґрунтування вибору програми аналога

На сьогоднішній день існує декілька популярних онлайн-сервісів та програм для фінансового планування та визначення досяжності цілей накопичення, серед яких можна виділити Sun Life Budget. Проте, незважаючи на широкий спектр функціональних можливостей, Sun Life Budget, як і інші подібні інструменти, має певні обмеження у визначенні досяжності фінансових цілей. Зокрема, ця програма може надавати недостатньо точні прогнози щодо можливості досягнення поставлених цілей накопичення в зазначені терміни, що може призвести до неоптимальних фінансових рішень та розчарувань користувачів.

Достовірність визначення досяжності фінансових цілей є критично важливою для ефективного планування. Користувачі покладаються на результати аналізу програмних засобів, щоб приймати обґрунтовані рішення щодо своїх фінансів, встановлювати реалістичні цілі та розробляти ефективні стратегії заощаджень. Неточні або ненадійні прогнози, які можуть надавати Sun Life Budget та інші подібні програми, можуть призвести до недосяжних цілей, що в кінцевому підсумку може негативно вплинути на фінансове благополуччя користувачів.

Крім того, підвищення достовірності визначення досяжності цілей накопичення дозволить користувачам краще розуміти свій фінансовий потенціал та обмеження. Це допоможе їм встановлювати більш реалістичні та досяжні цілі, а також коригувати свої фінансові стратегії відповідно до. Точніші прогнози

також дозволять користувачам приймати більш обґрунтовані рішення щодо розподілу своїх ресурсів та оптимізації своїх фінансів для досягнення довгострокових цілей.

З огляду на це, розробка нового програмного засобу з підвищеною достовірністю визначення досяжності цілей накопичення є актуальним завданням. Такий інструмент повинен використовувати передові алгоритми та методи аналізу даних для забезпечення точних та надійних прогнозів. Він також повинен враховувати різноманітні фактори, такі як доходи, витрати, інфляція, інвестиційні можливості та інші фінансові параметри, щоб надавати користувачам комплексну оцінку їхніх фінансових перспектив.

Підвищення достовірності роботи програмного засобу для визначення досяжності цілей накопичення матиме значні переваги для користувачів Sun Life Budget та інших подібних програм. Це дозволить їм приймати більш обґрунтовані фінансові рішення, встановлювати реалістичні цілі та розробляти ефективні стратегії заощаджень. У кінцевому підсумку, це сприятиме покращенню фінансового благополуччя користувачів та допоможе їм досягати своїх довгострокових фінансових цілей з більшою впевненістю та успіхом.

1.4. Висновок до розділу 1

Отже, у даному розділі було проведено аналіз предметної області визначення класу досяжності мети. Було розглянуто постановку задачі класифікації досяжності мети накопичення на основі набору вхідних параметрів. Також було проведено огляд відомих методів класифікації, таких як дерева рішень, логістична регресія, випадковий ліс, метод опорних векторів, наївний баєсівський класифікатор та нейронні мережі. Кожен метод було проаналізовано з точки зору його переваг, недоліків та доцільності використання для вирішення поставленої задачі.

2. ПРОЕКТУВАННЯ СЕРВЕРНОЇ ЧАСТИНИ ТА МОДУЛЮ КЛАСИФІКАЦІЇ ДОСЯЖНОСТІ МЕТИ WEB-СЕРВІСУ ПРОГНОЗУВАННЯ ОПТИМАЛЬНИХ КРОКІВ ДЛЯ ДОСЯГНЕННЯ МЕТИ

В підрозділі 1.1 було згадано про поділ WEB-сервісу для прогнозування оптимальних кроків досягнення мети складається на два основних компонентів: серверної частини та клієнтської частини. Також функціональністьWEB-сервіс було поділено на модулі: класифікації досяжності та визначення доцільності витрат.

Серверна частина є двигуном сервісу та відповідає за обробку даних користувачів, виконання алгоритмів машинного навчання.

Клієнтська частина відповідає за взаємодію з користувачем та візуалізацію результатів роботи системи. Вона надає зручний інтерфейс для введення користувачем своїх фінансових даних, відображення прогнозів та рекомендацій, отриманих від серверної частини.

Модуль класифікації досяжності фінансових цілей відповідає за визначення досяжності мети.

Модуль визначення доцільності витрат аналізує поточні витрати користувача та надає рекомендації щодо оптимізації витрат для досягнення заданої мети накопичення. Він враховує класифікацію досяжності мети, отриману від модуля класифікації, та інші фінансові дані користувача.

Серверна та клієнтська частини взаємодіють між собою за допомогою API (Application Programming Interface), що забезпечує передачу даних та обмін інформацією між компонентами системи.

На рисунку 2.1 зображено архітектурну схему WEB-сервісу прогнозування оптимальних кроків для досягнення мети.

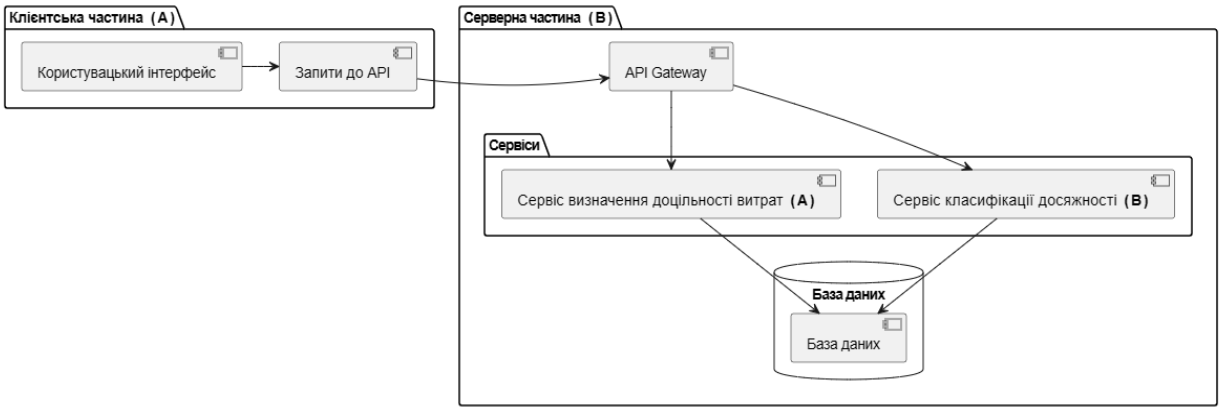


Рисунок 2.1 – Структурна схема архітектури WEB-сервісу.

Отже, у даному розділі буде розглянуто проектування серверної частини та модуля класифікації досяжності, що позначені на рис. 2.1 під нумерацією В.

2.1. Обґрунтування обраного методу машинного навчання

У розділі 1.2 було розглянуто різні підходи до вирішення поставленого завдання. Отже для вирішення задачі класифікації досяжності накопичення обрано метод Random Forest. Вибраний метод має переваги в контексті поставленої задачі, оскільки:

- Метод здатний ефективно працювати з великою кількістю вхідних ознак та складними нелінійними залежностями між ними. У нашому випадку вхідний набір даних містить 14 ознак, які можуть мати нетривіальні взаємозв'язки. Random Forest враховує важливість ознак та будує ансамбль дерев рішень, кожне з яких використовує випадкову підмножину ознак, що дозволяє уникнути перенавчання та підвищити узагальнюючу здатність моделі.
- Метод є стійким до шумів та викидів у даних. Оскільки наша задача пов'язана з фінансовими даними, які можуть містити помилки або нетипові значення, важливо, щоб модель не перенавчалася на окремих прикладах.

- Метод забезпечує оцінку важливості ознак. Це дозволяє проаналізувати, які саме характеристики мають найбільший вплив на результат класифікації. Такий аналіз може бути корисним для розуміння ключових факторів, що впливають на здійсненність мети накопичення.

Отже, враховуючи переваги методу Random Forest та успішні приклади його застосування в схожих задачах, можна зробити висновок про доцільність його використання для класифікації досяжності мети накопичення.

У даному підрозділі детально розглянемо принцип роботи та проектування методу машинного навчання Random Forest, що було обрано для вирішення задачі класифікації досяжності мети накопичення [4,15].

Процес побудови Random Forest можна описуються наступними кроками:

2.1.1. Формування випадкових підвибірок з навчальної вибірки

Для кожного дерева в ансамблі генерується випадкова підвибірка прикладів з початкової навчальної вибірки методом бутстрепінгу Нехай $D = \left(x^i, y^i\right)_{i=1}^{N}$ – навчальна вибірка, де x^i – вектор ознак і-го прикладу, y^i – відповідний клас, N – кількість прикладів. Тоді для кожного дерева $t = 1, \dots, T$ генерується випадкова підвибірка D_t з D методом бутстрепінгу, формула (2.1):

$$D_t = \left(x^{i_k}, y^{i_k}\right)_{k=1}^{N_t}, \tag{2.1}$$

де $i_k{}_{k=1}^{N_t}$ – випадкові індекси прикладів з $1, \dots, N$, N_t – розмір підвибірки.

2.1.2. Побудова дерев рішень

Для кожної підвибірки D_t будується окреме дерево рішень f_t, при цьому, на кожному з вузлів дерева випадковим чином вибирається підмножина ознак $F_t \subseteq 1, \dots, m$ для розбиття, де m – загальна кількість ознак. Розбиття відбувається за критерієм максимального зменшення неоднорідності, формула (2.2):

$$\Delta I(s,j) = I(D_i) - \frac{|D_t^L|}{|D_t|} I(D_t^L) - \frac{|D_t^R|}{|D_t|} I(D_t^R), \quad (2.2)$$

де s – поточний вузол, $j \in F_t$ – ознака для розбиття, D_t^L та D_t^R – підмножини прикладів, що потрапляють у лівий та правий дочірні вузли відповідно, I – міра неоднорідності.

2.1.3. Голосування та агрегація результатів

При класифікації нового прикладу x кожне дерево f_t в ансамблі дає свій прогноз класу $y_t = f_t(x)$ а фінальний прогноз класу визначається шляхом більшості голосів серед усіх дерев, формула (2.3):

$$y = arg \max_x \sum_{t=1}^{T} I(y_t = c), \quad (2.3)$$

де I – функція-індикатор, яка дорівнює 1, якщо умова виконується, і 0 в іншому випадку.

В даному методі використовується оцінка важливості ознак використовується міра зменшення неоднорідності. Для кожної ознаки j розраховується середнє зменшення неоднорідності по всіх деревах, формула (2.4):

$$ImpDec(j) = \frac{1}{T} \sum_{t=1}^{T} \sum_{s \in S_t(j)} \frac{|D_t^s|}{|D_t|} \Delta I(s,j), \quad (2.4)$$

де $S_t(j)$ – множина вузлів в дереві t, де відбулось розбиття по ознаці j, D_t^s – підмножина прикладів у вузлі s.

У цьому розділі було детально описано процес побудови Random Forest, який складається з формування випадкових підвибірок з навчальної вибірки,

побудови дерев рішень на цих підвибірках та агрегації результатів шляхом голосування. Також розглянуто метод оцінки важливості ознак на основі зменшення неоднорідності в вузлах дерев.

2.2. Розробка алгоритму роботи модуля класифікації

У даному підрозділі буде детально описано процес розробки алгоритму роботи модуля класифікації досяжності фінансових цілей накопичення з використанням методу машинного навчання Random Forest. Буде представлено схему алгоритму.

Відповідно до мети було розроблено схему алгоритму модуля визначення класу досяжності мети та представлено на рис. 2.2:

Рисунок 2.2 – Схема алгоритму роботи модуля визначення класу досяжності мети.

Першим кроком в модулі визначення класу мети накопичення буде завантаження набору даних (блок 1) із параметрами цілей накопичення та їх класами досяжності.

Далі переходимо до візуалізації результатів аналізу набору даних параметрів цілей накопичення. Спочатку проводиться аналіз впливу різних параметрів на досяжність цілі (блок 2), який дозволить визначити ступінь впливу різних параметрів на можливість досягнення мети накопичення. Результати аналізу впливу параметрів на досяжність цілі показано на рисунках у розділі 4.

Потім проводиться попередня обробка даних: видалення непотрібних стовпців із набору даних (блок 3).

Далі виконуємо обробку пропущених значень та викидів у даних (блок 4). Це може включати заповнення пропущених значень середнім, медіаною або іншими методами, а також видалення або заміну викидів, які можуть негативно вплинути на якість моделі.

Оскільки у нашому наборі даних досяжність цілі вже представлена у вигляді двох класів (досяжна або недосяжна), переходимо до аналізу кількості досяжних і недосяжних цілей у наборі даних. Це дозволить нам оцінити збалансованість класів і визначити, чи є необхідність у додатковій обробці даних для вирішення проблеми незбалансованості класів (блок 5).

Далі ділимо набір даних на залежні та незалежні змінні: Набір даних розділяємо на навчальну та тестову вибірки (блок 6). Визначаємо обсяг навчальної та тестової вибірок. Виконуємо стандартне масштабування (нормування) параметрів (блок 7).

Потім визначимо клас досяжності цілі методом випадкового лісу (Random Forest) (блок 8).

Представлений алгоритм описує послідовність кроків для визначення класу досяжності фінансових цілей накопичення за допомогою методу машинного навчання Random Forest. Він включає етапи завантаження та попередньої обробки даних, обробки пропущених значень та викидів, аналізу

впливу параметрів, розділення даних на навчальну та тестову вибірки, а також застосування моделі Random Forest для визначення класу досяжності цілі.

2.3. Проектування серверної частини

У даному підрозділі буде представлено проектування серверної частини WEB-сервісу для забезпечення ефективного функціонування та взаємодії користувачів з розробленим модулем класифікації. Серверна частина надає можливість користувачам отримувати персоналізовані рекомендації щодо досягнення фінансових цілей на основі аналізу їхніх даних за допомогою модуля класифікації.

Серверна частина є ключовим компонентом WEB-сервісу прогнозування оптимальних кроків для досягнення мети, оскільки вона відповідає за обробку запитів від клієнтської частини, взаємодію з базою даних, виконання бізнес-логіки та надання відповідей клієнту. Правильне проектування серверної частини є запорукою ефективної та надійної роботи всього WEB-сервісу.

У підрозділі представлені UML діаграми, що відображають структуру та поведінку серверної частини WEB-сервісу. Діаграми включають діаграми класів, діаграми послідовності, діаграми об'єктів та діаграми компонентів. Розглянемо кожну з них детальніше [8].

UML-діаграма класів – це структурна діаграма, яка показує класи системи, їх атрибути, методи та взаємозв'язки між ними. Вона надає статичне представлення структури системи.

На рисунку 2.3 зображена UML-діаграма класів серверної частини WEB-сервісу.

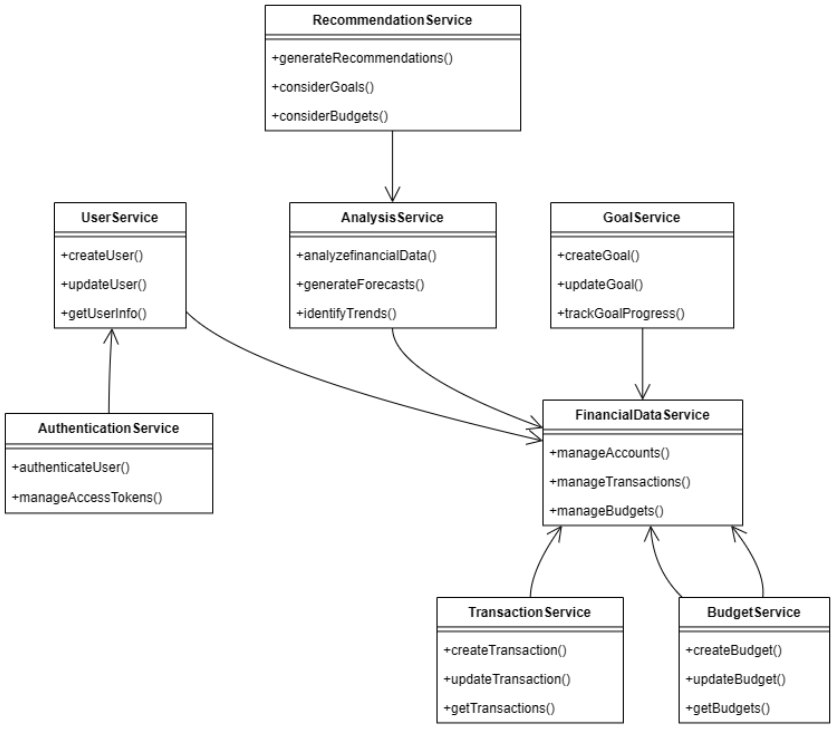


Рисунок 2.3 – UML-діаграма класів серверної частини WEB-сервісу.

Детальний опис компонентів діаграми класів серверної частини WEB-сервісу:

– Клас UserService відповідає за управління даними користувачів, включаючи створення, оновлення та отримання інформації про користувачів. Він також взаємодіє з класом AuthenticationService для автентифікації користувачів та управління токенами доступу.
– Клас FinancialDataService відповідає за управління фінансовими даними користувачів, такими як рахунки. Він тісно співпрацює з класами TransactionService та BudgetService для створення нових записів витрат чи бюджетів.
– Клас AnalysisService відповідає за аналіз фінансових даних користувачів з використанням алгоритмів машинного навчання та статистичних методів. Він генерує прогнози, виявляє тенденції та надає інформацію для прийняття фінансових рішень.

– Клас RecommendationService використовує результати аналізу від AnalysisService для генерації персоналізованих рекомендацій щодо оптимальних кроків для користувачів. Він враховує цілі, бюджети та поточний фінансовий стан користувача.
– Класи GoalService та BudgetService відповідають за управління фінансовими цілями та бюджетами користувачів відповідно. Вони дозволяють створювати, редагувати та відстежувати прогрес у досягненні цілей та дотриманні бюджетів.

UML-діаграма компонентів – це структурна діаграма, яка показує компоненти системи, їх інтерфейси та залежності. Вона надає високорівневе представлення фізичних будівельних блоків системи.

На рисунку 2.4 зображено діаграму компонентів, яка відображає детальну структуру серверної частини WEB-сервісу.

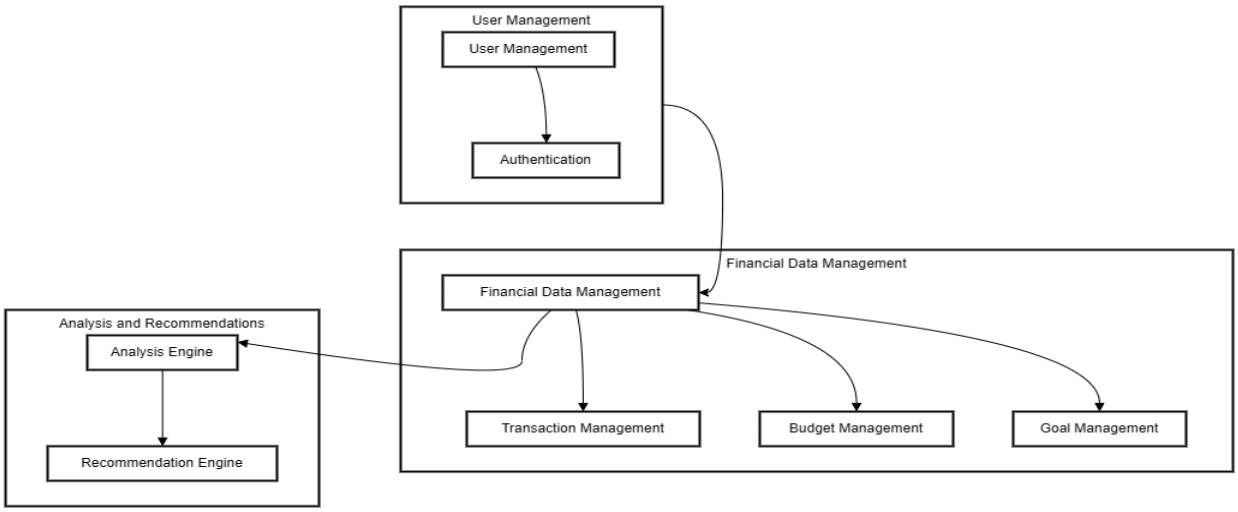


Рисунок 2.4 – UML-діаграма компонентів серверної частини WEB-сервісу.

Детальний опис складових діаграми компонентів серверної частини WEB-сервісу:

– Компонент User Management відповідає за управління користувачами, включаючи реєстрацію, автентифікацію та авторизацію. Він взаємодіє з

компонентом Authentication для забезпечення безпеки та керування доступом.

- Компонент Financial Data Management відповідає за управління фінансовими даними користувачів, включаючи рахунки, транзакції та бюджети. Він взаємодіє з компонентами Transaction Management та Budget Management для виконання відповідних операцій.
- Компонент Analysis Engine виконує аналіз фінансових даних користувачів, генерує прогнози та надає інформацію для прийняття рішень. Він використовує алгоритми машинного навчання та статистичні методи для обробки даних.
- Компонент Recommendation Engine використовує результати аналізу для генерації персоналізованих рекомендацій щодо оптимальних фінансових кроків для користувачів.
- Компоненти Goal Management та Budget Management забезпечують функціональність для управління фінансовими цілями та бюджетами користувачів відповідно.

UML-діаграма розгортання – це структурна діаграма, яка показує фізичну архітектуру системи, включаючи вузли (апаратні пристрої), компоненти, що працюють на цих вузлах, та з'єднання між ними.

UML-діаграма розгортання надає чітке уявлення про фізичну архітектуру серверної частини WEB-сервісу, показуючи основні апаратні вузли, компоненти, що працюють на цих вузлах, та взаємодію між ними.

На рисунку 2.5 зображено UML діаграму розгортання, яка ілюструє фізичне розміщення компонентів серверної частини WEB-сервісу на апаратних вузлах.

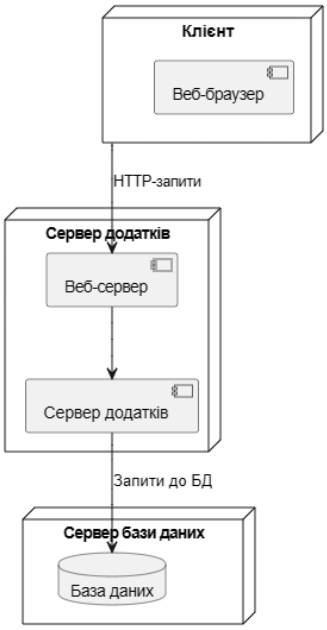


Рисунок 2.5 – UML-діаграма розгортання серверної частини WEB-сервісу.

Діаграма розгортання складається з трьох основних вузлів: Клієнт, Сервер додатків та Сервер бази даних.

Вузол «Клієнт»: Містить компонент: Веб-браузер, через який здійснюється доступ.

Вузол «Сервер додатків»: Включає два компоненти: Веб-сервер і Сервер додатків. Веб-сервер обробляє HTTP-запити від клієнтів і передає їх до серверу додатків. Сервер додатків містить основну логіку WEB-сервісу, обробляє запити та формує відповіді. Сервер додатків взаємодіє з базою даних для отримання та збереження необхідних даних.

Вузол «Сервер бази даних»: Містить компонент База даних. База даних зберігає всі необхідні дані для функціонування WEB-сервісу, такі як інформація про користувачів, фінансові дані, налаштування тощо.

Подана діаграма розгортання серверної частини WEB-сервісу містить наступні зв’язки між компонентами:

- Клієнти (Веб-браузер) надсилають HTTP-запити до Веб-сервера.

- Веб-сервер передає отримані запити до Сервера додатків для подальшої обробки.
- Сервер додатків, у разі потреби, надсилає запити до Бази даних для отримання або збереження даних.
- База даних повертає запитані дані до Сервера додатків.
- Сервер додатків формує відповідь і передає її назад до Веб-сервера.
- Веб-сервер відправляє сформовану відповідь до Клієнта.

UML-діаграма послідовності – це діаграма що демонструє, як об'єкти взаємодіють один з одним у часі для досягнення певного результату. Вона відображає послідовність повідомлень між учасниками взаємодії та дозволяє зрозуміти часові залежності та логіку взаємодії компонентів системи.

На рисунку 2.6 зображено діаграму послідовності створення нової цілі накопичення.

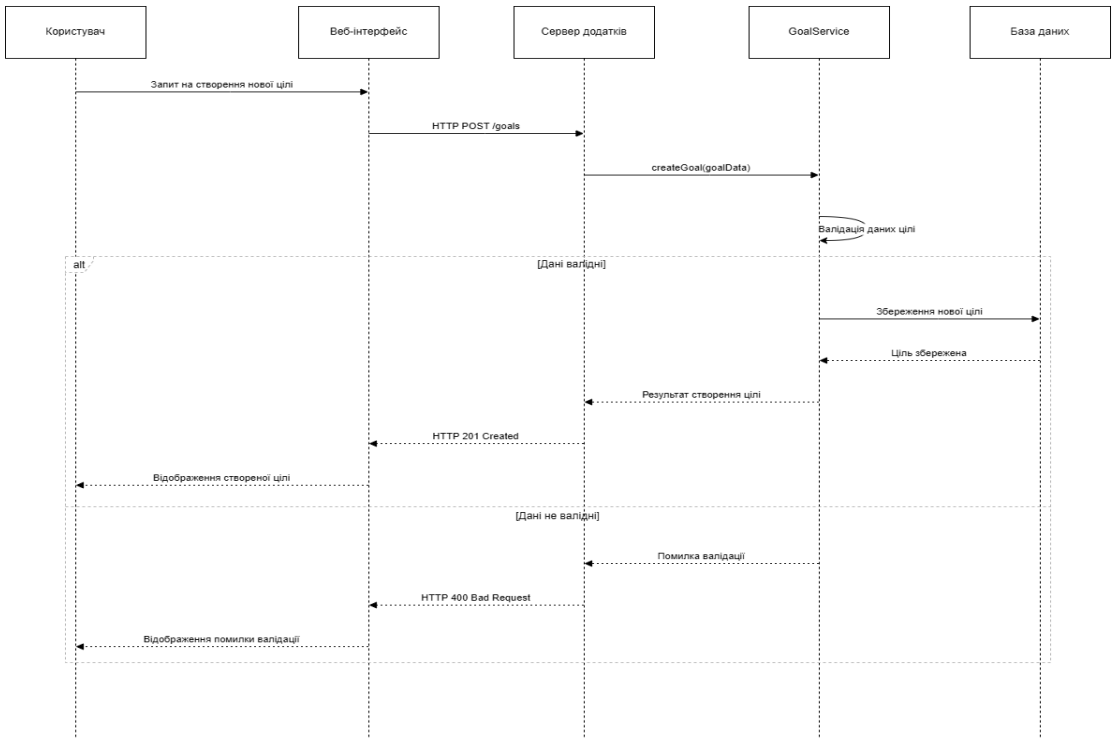


Рисунок 2.6 – UML-діаграма послідовності створення нової цілі.

Детальний опис UML-діаграми послідовності для процесу створення нової мети:

1. Користувач взаємодіє з веб-інтерфейсом WEB-сервісу та ініціює запит на створення нової цілі.
2. Веб-інтерфейс надсилає HTTP POST запит з даними цілі на відповідний маршрут (/goals) сервера додатків.
3. Сервер додатків отримує запит і передає дані цілі до сервісу GoalService для подальшої обробки.
4. GoalService проводить валідацію отриманих даних цілі, перевіряючи їх коректність і відповідність вимогам.
5. Якщо дані цілі валідні:
   - GoalService зберігає нову ціль у базі даних.
   - База даних підтверджує збереження цілі та повертає результат до GoalService.
   - GoalService передає результат створення цілі назад до сервера додатків.
   - Сервер додатків відправляє HTTP-відповідь зі статусом 201 Created до веб-інтерфейсу.
   - Веб-інтерфейс відображає створену ціль користувачу.
6. Якщо дані цілі не валідні:
   - GoalService повертає помилку валідації до сервера додатків.
   - Сервер додатків відправляє HTTP-відповідь зі статусом 400 Bad Request до веб-інтерфейсу.
   - Веб-інтерфейс відображає помилку валідації користувачу.

UML-діаграма послідовності демонструє покроковий процес створення нової цілі в WEB-сервісі. Вона показує взаємодію між користувачем, веб-інтерфейсом, сервером додатків, сервісом GoalService та базою даних. Діаграма також враховує різні сценарії, такі як успішне створення цілі та обробка помилок валідації даних.

UML-діаграма активності – це діаграма поведінки, яка показує потік керування від однієї активності до іншої. Вона описує динамічний аспект

системи, зображуючи послідовність дій, розгалуження, паралельні потоки та умови.

Діаграма активності дозволяє детально описати процес генерації персоналізованих рекомендацій, демонструючи послідовність дій, розгалуження та умови.

На рисунку 2.7 зображено UML-діаграму активності серверної частини WEB-сервісу.

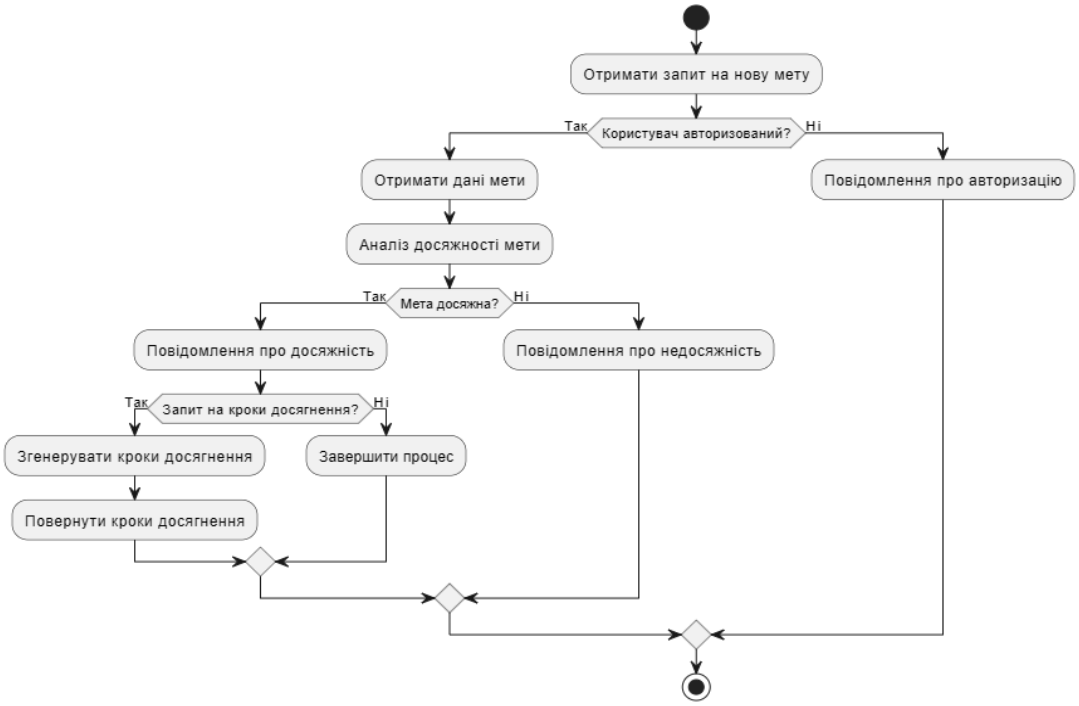


Рисунок 2.7 – UML-діаграма активності серверної частини WEB-сервісу.

Детальний опис компонентів діаграми активності серверної частини WEB-сервісу:

1. Серверна частина WEB-сервісу отримує запит на створення нової мети.
2. Перевіряється, чи авторизований користувач.
   - Якщо користувач авторизований, процес переходить до кроку 3.
   - Якщо користувач не авторизований, повертається повідомлення про необхідність авторизації.

3. Отримуються дані нової мети від користувача.
4. Перевіряється, чи є мета досяжною.
   - Якщо мета досяжна, повертається повідомлення про досяжність мети.
   - Якщо мета недосяжна, повертається повідомлення про недосяжність мети.
5. Перевіряється, чи є запит на генерацію оптимальних кроків досягнення мети.
   - Якщо є запит, генеруються оптимальні кроки досягнення мети.
   - Якщо немає запиту, процес завершується.
6. Повертаються згенеровані кроки досягнення мети у відповідь на запит.

Представлені UML діаграми надають вичерпне уявлення про структуру, поведінку та розгортання серверної частини WEB-сервісу. Вони слугують потужним інструментом для проектування, розробки, документування та комунікації архітектури системи між членами команди та зацікавленими сторонами .

2.4. Висновок до розділу 2

У даному розділі було детально описано процес розробки модуля класифікації досяжності фінансових цілей накопичення з використанням методу машинного навчання Random Forest. Представлено схему алгоритму роботи модуля класифікатора, яка включає етапи завантаження та попередньої обробки даних, аналізу впливу параметрів, розділення даних на навчальну та тестову вибірки, а також застосування моделі Random Forest для визначення класу досяжності цілі. Також було розглянуто проектування серверної частини WEB-сервісу з використанням UML діаграм, що відображають структуру та поведінку системи.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ ТА МОДУЛЮ КЛАСИФІКАЦІЇ ДОСЯЖНОСТІ МЕТИ WEB-СЕРВІСУ ПРОГНОЗУВАННЯ ОПТИМАЛЬНИХ КРОКІВ ДЛЯ ДОСЯГНЕННЯ МЕТИ

3.1. Обґрунтування вибору інструментів розробки

Для розробки серверної частини WEB-сервісу важливо обрати мову програмування та фреймворк, які найкраще відповідають вимогам проекту та забезпечують ефективну реалізацію необхідної функціональності. У цьому підрозділі буде проведено детальний аналіз переваг та недоліків популярних мов програмування та фреймворків для обґрунтування вибору оптимальних інструментів розробки.

Python – це інтерпретована, високорівнева, об’єктно орієнтована мова програмування зі строгою динамічною типізацією [9].

До переваг цієї мови програмування можна віднести.

– Простий у вивченні синтаксис.
– Багата екосистема бібліотек та фреймовркв.
– Динамічна типізація, що прискорює процес розробки.
– Кросплатформеність.
– Велика спільнота розробників.

Хоч Python і має значні переваги у вигляді великої кількості готових рішень, він також має низку суттєвих недоліків, таких як:

– Низка швидкодія у порівнянні з іншими мовами.
– Глобальний інтерпретатор блокувань (GIL).
– Динамічна типізація перешкоджає в знаходженні помилок в процесі написання коду.

Іншою мовою для рішення потреб розробки може бути JavaScript (Node.js) [9].

Це динамічна, об'єктно-орієнтована мова програмування, широко використовується для веб-розробки, що має наступні переваги:

– Можливість використання єдиної мови програмування для фронтенду та бекенду.
– Найбільша екосистема пакетів та бібліотек
– Швидке прототипування та розробка завдяки динамічній типізації

Хоч і JavaScript має суттєві переваги, що визначаються його універсальністю, він також має суттєві недліки:

– Відсутність строгої типізації.
– Callback Hell – неймовірна кількість вкладеність функцій зворотного виклику.
– Бідний на інструменти для аналізу даних як Python.
– Постійні зміни та відсутність підтримки старих версій.

Ще однією мовою програмування що може бути використана для рішення покриття потреб є Java. Це Об'єктно-орієнтована мова програмування з автоматичним керуванням пам'яттю націлена на розробку високопродуктивних продуктів з масштабованістю для Enterprise проектів [11].

До переваг цієї мови програмування відносять:

– Строга типізація.
– Перевірка помилок на етапі компіляції.
– Кросплатформеність.

Хоча Java і є ідеальним рішенням для Enterprise клієнтів та розробки великих систем, вона також має свої недоліки в інших сферах:

– Складний у вивченні, багатослівний синтаксис.
– Довший цикл розробки через необхідність компіляції.
– Менша гнучкість та швидкість розробки у порівнянні з динамічно типізованими мовами.

Зважаючи на специфіку WEB-сервісу, який передбачає роботу застосування алгоритмів машинного навчання, Python є найбільш доцільним вибором серед розглянутих мов програмування

3.1.1. Обґрунтування технології серверної частини

При розробці серверної частини WEB-сервісів постає питання вибору між використанням фреймворків та написанням коду з нуля. Хоча написання коду з нуля дає повний контроль над процесом розробки та дозволяє створювати унікальні рішення, використання фреймворків має ряд суттєвих переваг, які роблять їх привабливим вибором для більшості проектів.

Фреймворки – це набори інструментів, бібліотек та конвенцій, які спрощують та прискорюють процес розробки веб-додатків. Вони надають готові компоненти та шаблони для вирішення типових задач, таких як маршрутизація URL-адрес, обробка запитів, взаємодія з базами даних та генерація HTML-сторінок. Використання фреймворків дозволяє розробникам зосередитися на створенні бізнес-логіки та функціональності додатку, не витрачаючи час на низькорівневі деталі та рутинні задачі [12].

Django – це потужний веб-фреймворк Python з широким набором вбудованих інструментів, який дозволяє швидко розробляти повнофункціональні веб-додатки з чистою архітектурою.

Перевагами Django є:

– Повнофункціональний набір вбудованих інструментів.
– Власна ORM (Object-Relational Mapping) для роботи з базами даних.
– Вбудована система автентифікації та авторизації користувачів.
– Автоматична генерація адміністративного інтерфейсу для керування даними.

До недоліків Django можна віднести:

– Монолітна архітектура та жорстка структура.
– Високий поріг.
– Надмірність для простих проектів.

FastAPI – це сучасний, високопродуктивний веб-фреймворк Python, який поєднує в собі простоту використання та швидкість розробки з потужними функціями, такими як автоматична генерація документації та підтримка асинхронного програмування.

Переваги FastAPI

– Підтримка асинхронного програмування ASGI.
– Автоматична генерація інтерактивної документації API.
– Зручна валідація та серіалізація данних.
– Простий у вивченні.

До недоліків FastAPI відносять:

– Мала екосистема плагінів у порівнянні з іншими фреймворками.
– Відсутність вбудованої ORM.

Flask – це легкий та гнучкий мікрофреймворк Python, який надає основні інструменти для розробки веб-додатків, дозволяючи розробникам самостійно обирати та інтегрувати потрібні компоненти.

До його переваг відносять:

– Мінімалістичний та гнучкість, простота у використанні.
– Швидкий.
– Широкий вибір розширень.

Основні недоліки Flask:

– Відсутність вбудованих інструментів таких як: робота з базами даних, автентифікацією та авторизацією
– Ручна конфігурації та налаштування.
– Погана продуктивність в високонавантажених додатках.

Starlette – це легкий та високопродуктивний асинхронний веб-фреймворк Python, побудований на основі протоколу ASGI, який забезпечує швидкодію та простоту використання для розробки сучасних веб-додатків.

– Простий у використанні, мінімалістичний синтаксис.
– Підтримує асинхронне програмування.
– Сумісний з багатьма асинхронними бібліотеками та інструментами Python

Хоча Starlette, на перший погляд, виглядає як ідеальне рішення для нашого WEB-сервісу, але також він має низку недоліків:

– Мала кількість функцій та інструментів.

– Потребує глибокого розуміння асинхронного програмування та специфіки використання ASGI

Враховуючи масштаб, складність та специфіку серверної частини WEB-сервісу, Django є найбільш доцільним вибором серед розглянутих фреймворків Python. Недоліки Django, такі як монолітна архітектура та відносно високий поріг входження, є не критичними для даного проекту, оскільки передбачається розробка повнофункціонального веб-додатку з множиною взаємопов'язаних компонентів.

3.1.2. Обґрунтування технології модуля класифікатора

Для реалізації модуля класифікатора на основі Random Forest в мові Python існує декілька популярних бібліотек та фреймворків. Розглянемо переваги та недоліки найбільш розповсюджених з них.

Scikit-learn – це бібліотека машинного навчання з відкритим кодом, яка надає ефективну реалізацію RandomForestClassifier разом з іншими алгоритмами. До переваг бібліотеки Scikit-learn відносять:

- Проста та зручна у використанні, з добре документованим API.
- Висока швидкодія та оптимізація алгоритмів.
- Підтримка паралельних обчислень для прискорення навчання моделей.
- Велика спільнота користувачів та активна підтримка.

H2O – це відкритий фреймворк для машинного навчання, який пропонує розподілену реалізацію H2ORandomForestEstimator, що є потужним інструментом для обробки надвеликих обсягів даних. Загальним недоліком даного фреймворку є мала швидкодія на малих та середніх наборах даних.

Іншою реалізацією є LGBMClassifier з фреймворку LightGBM. LightGBM – це градієнтно бустинговий фреймворк, який також реалізує алгоритм Random Forest з фокусом на швидкодію та ефективне використання пам'яті. Реалізація має наступні переваги:

- Висока швидкість навчання та передбачення.
- Ефективне використання пам'яті завдяки техніці розріджених даних.

- Вбудована підтримка категоріальних змінних.

На перший погляд даний фреймворк з реалізацією методу Random Forest, є найоптимальнішим але він має суттєвий недолік а саме нестабільність на малих наборах даних.

Ще однією реалізацією є XGBRFClassifier з бібліотеки XGBoost. XGBoost – це високопродуктивна бібліотека машинного навчання, яка реалізує оптимізовані алгоритми градієнтного бустингу, до її переваг відносять:

- Надзвичайно висока швидкість навчання.
- Відмінна якість роботи, особливо на структурованих даних.

Хоча XGBoost є потужним інструментом для побудови високоякісних моделей Random Forest, його використання в даному проекті може бути надлишковим, враховуючи відносно невеликий обсяг даних

Враховуючи вимоги до модуля класифікатора, такі як швидкодія, простота використання та можливість ефективної обробки даних помірного розміру, бібліотека Scikit-learn є доцільним вибором для реалізації Random Forest в цьому проекті [13].

Отже, у підрозділ було проаналізовано популярні мови програмування та фреймворки для розробки серверної частини WEB-сервісу. Враховуючи специфіку проекту, що передбачає роботу з алгоритмами машинного навчання, мова програмування Python та фреймворк Django були обрані як оптимальні інструменти для реалізації серверної частини. Для модуля класифікатора на основі Random Forest бібліотека Scikit-learn була визначена як найбільш доцільна, зважаючи на її простоту використання, швидкодію та ефективність обробки даних помірного розміру.

3.2. Обґрунтування середовища розробки

При розробці серверної частини WEB-сервісу важливо обрати зручне та ефективне середовище розробки (IDE), яке забезпечить продуктивність, зручність та якість процесу розробки. У цьому підрозділі буде проведено

порівняльний аналіз популярних IDE для розробки на мові Python, таких як PyCharm, Visual Studio та Visual Studio Code, з метою обґрунтування вибору оптимального середовища розробки для даного проекту.

PyCharm – це інтегроване середовище розробки (IDE), спеціально створене для мови програмування Python. PyCharm надає широкий набір інструментів та функцій, що спрощують та прискорюють процес розробки Python-проектів [31].

Переваги PyCharm:

– Інтелектуальне автодоповнення, підказки.
– Вбудований дебагер.

На рисунку 3.1 зображено загальний вигляд вікна інтегрованого середовища розробки PyCharm.

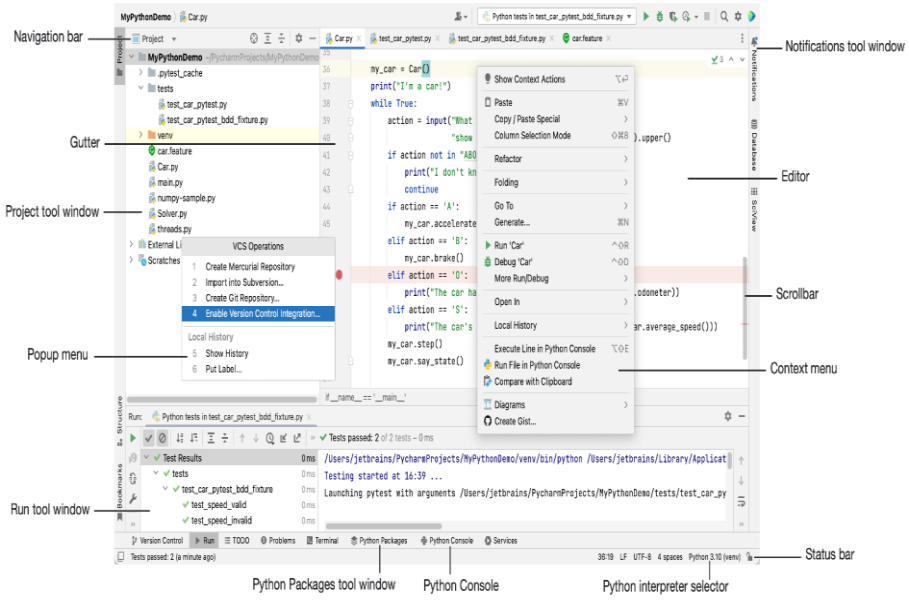


Рисунок 3.1 – Загальний вигляд вікна інтегрованого середовища розробки PyCharm.

Visual Studio – це потужне інтегроване середовище розробки від компанії Microsoft, яке підтримує широкий спектр мов програмування, включаючи Python. Visual Studio надає багатий набір інструментів для розробки, налагодження та розгортання програмних проектів [32].

Переваги Visual Studio:

– Підтримка множини мов.
– Вбудований дебагер.
– Інструменти профілювання та оптимізації.
– Розширена екосистема плагінів.

Хоча Visual Studio є популярним та найбільш потужним середовищем розробки, але також має свої недоліки, такі як:

– Високе споживання ресурсів.
– Складний інтерфейс.
– Тривале налаштування та освоєння.

На рисунку 3.2 зображено загальний вигляд вікна інтегрованого середовища розробки Visual Studio.

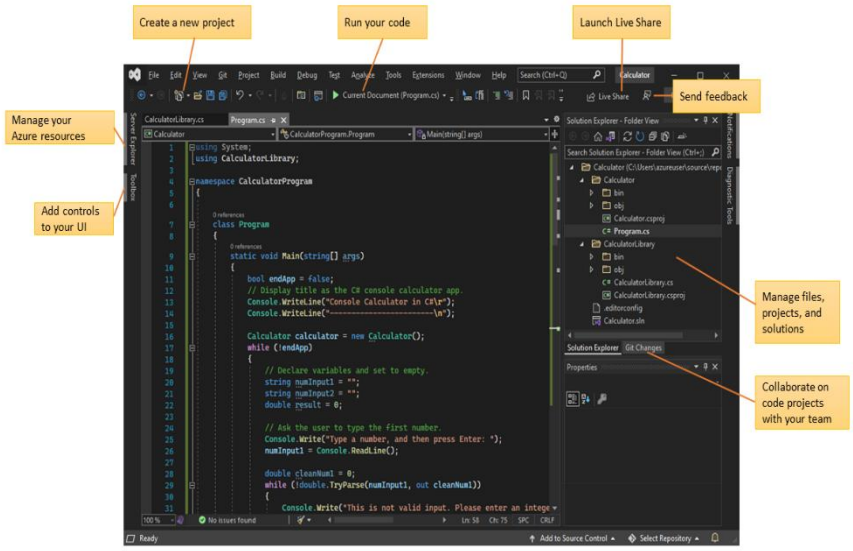


Рисунок 3.2 – Загальний вигляд вікна інтегрованого середовища розробки Visual Studio.

Visual Studio Code – це легкий та швидкий редактор коду від компанії Microsoft, який підтримує множину мов програмування, включаючи Python. Visual Studio Code позиціонується як зручний та розширюваний редактор для швидкої розробки та редагування коду [33].

Переваги

– Швидкий запуск, низьке споживання ресурсів.
– Підтримка множини мов.
– Вбудований термінал.
– Багата екосистема розширень.
– Інтеграція з Git.
– Кросплатформеність, безкоштовність.

Оскільки Visual Studio Code це редактор коду, то він має низку недоліків зав’язаних на функціональності середовища розробки.

На рисунку 3.3 зображено загальний вигляд вікна редактора коду Visual Studio Code.

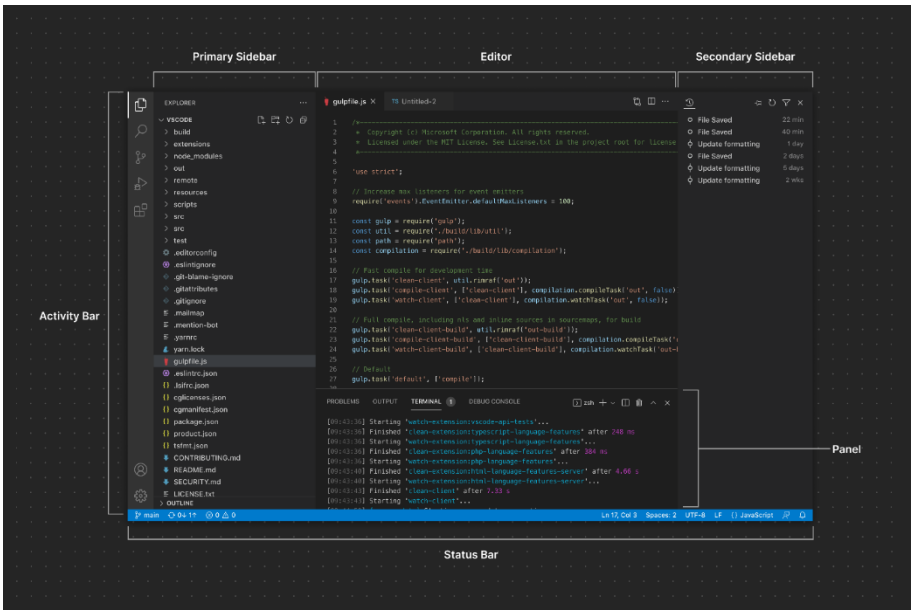


Рисунок 3.3 – Загальний вигляд вікна редактора коду Visual Studio Code.

Враховуючи специфіку та вимоги до розробки серверної частини WEB-сервісу, найбільш доцільним вибором середовища розробки є PyCharm. Його переваги, такі як вбудована підтримка Django, інтелектуальні інструменти для роботи з Python кодом та зручний дебагер, роблять його оптимальним рішенням для ефективної та продуктивної розробки. Хоча PyCharm має певні недоліки, як-от відносно високе споживання ресурсів, його переваги переважають для даного проекту.

Таким чином, вибір PyCharm як основного середовища розробки для серверної частини WEB-сервісу є обґрунтованим рішенням, що забезпечить ефективність, продуктивність та якість процесу розробки.

3.3. Опис набору даних параметрів цілей для навчання та тестування модуля класифікації

Для ефективного навчання та тестування модуля класифікації досяжності необхідно мати репрезентативний набір даних, який містить приклади цілей з різними параметрами та відповідними класами досяжності. У цьому підрозділі буде детально описано набір даних, який використовується для навчання та тестування розробленого модуля класифікації. Буде представлено структуру набору даних, роз’яснено значення кожного атрибута та проаналізовано розподіл класів досяжності в наборі даних. Розуміння характеристик набору даних є важливим для оцінки якості та ефективності роботи модуля класифікації

Опис набору даних параметрів цілей для навчання та тестування модуля класифікації зображено на рисунках 3.4-3.5.

	start_date	end_date	initial_goal	monthly_income	monthly_savings	food_expenses	travel_expenses
0	01.01.2020	25.04.2021	35030	8918	545	1170	945
1	04.01.2021	22.09.2022	24615	9456	1759	888	884
2	17.01.2020	28.08.2022	33787	8350	1571	915	891
3	12.01.2021	26.11.2022	7502	9627	2745	1283	606
4	04.06.2020	24.07.2021	35445	8262	2631	1506	318

Рисунки 3.4 – Вигляд фрагменту набору даних перших 7 параметрів.

	entertainment_expenses	internet_expenses	utility_expenses	mobile_expenses	subscription_expenses	healthcare_expenses	education_expenses	charity_expenses	target
0	153	53	253	92	269	115	718	107	0
1	342	195	429	132	131	293	956	440	1
2	209	113	224	51	153	108	537	76	0
3	723	157	453	72	90	613	349	60	1
4	785	17	478	57	105	392	871	121	1

Рисунок 3.5 – Вигляд фрагменту набору даних останніх 9 параметрів.

Представлені зображення демонструють структуру та зміст набору даних, який використовується для навчання та тестування модуля класифікації. Кожен рядок відповідає окремому прикладу фінансової цілі накопичення, а стовпці містять значення різних атрибутів, що характеризують цю ціль.

Для кращого розуміння значення кожного атрибута та діапазону змін, які можуть бути у цих атрибутів, доцільно провести детальний аналіз. Оскільки star_date та end_date це параметри бажаного періоду, за який мало б відбутись накопичення, то в подальшому їх не використовуватимемо для навчання, а замість них буде використано їх різницю.

Атрибути, представлені в таблиці, охоплюють різні аспекти фінансового планування, такі як початкова ціль накопичення, щомісячний дохід, заощадження та витрати в різних категоріях. Ці атрибути є ключовими факторами, що впливають на досяжність фінансової цілі та будуть використані для навчання моделі класифікації. Розуміння значення кожного атрибута та його можливого діапазону змін є важливим для коректної інтерпретації результатів

класифікації та прийняття обґрунтованих рішень щодо досяжності фінансових цілей. Опис атрибутів подано в табл. 3.1:

Таблиця 3.1 – Опис атрибутів набору даних.

Атрибут	Значення	Діапазон змін
Initial goal	Кількісна грошова ціль	15445…26123
Monthly income	Грошове нарахування щомісяця	6524…10117
Monthly savings	Кошти що планується заощадити	366…3366
Travel expenses	Витрати на подорожі	0…1261
Food expenses	Витрати на їжу	485…2634
Entertainment expenses	Витрати на розваги	76…826
Internet expenses	Витрати за використання інтернет трафіку	47…261
Utility expenses	Витрати на комунальні послуги	155…516
Mobile expenses	Витрати за мобільні послуги	29…176
Subscription expenses	Витрати на інші підписки	0…337
Healthcare expenses	Витрати на лікування	178…824
Education expenses	Витрати на освіту	0…1045
Charity expenses	Витрати на благодійність	0…547
Target (output target)	Клас досяжності (Вихідний параметр)	0…1

Отже, набір даних містить 21153 екземплярів даних з яких 12831 є досяжними тобто мають target=1, а 8322 є не досяжними оскільки їх клас був визначений як 0.

3.4. Програмна реалізація модуля класифікації

Перш за все потрібно реалізувати головний модуль, що буде класифікувати чи є нова мета досяжною чи ні. Наступний код є прикладом побудови моделі

класифікації з використанням Random Forest для передбачення цільової змінної на основі CSV-файлу.

Спочатку код імпортує необхідні бібліотеки: «pandas» для роботи з даними, train_test_split з scikit-learn для розділення даних на тренувальний та тестовий набори, RandomForestClassifier для побудови моделі випадкового лісу, accuracy_score, precision_score, recall_score та f1_score для оцінки якості моделі, а також dump та load з joblib для збереження та завантаження моделі [13-14]

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from joblib import dump, load.
```

Далі код завантажує дані з чотирьох CSV-файлів, та об'єднує їх в один DataFrame за допомогою pd.concat().

```
files = ["goals_2022.csv", "goals_2021.csv", "goals_2021.csv", "goals_2020.csv"]
data = pd.concat([pd.read_csv(file) for file in files], ignore_index=True).
```

Потім код виконує попередню обробку даних. Він перетворює стовпці 'start_date' та 'end_date' на тип datetime, а потім видаляє їх з DataFrame.

```
data['start_date'] = pd.to_datetime(data['start_date'])
data['end_date'] = pd.to_datetime(data['end_date'])
data = data.drop(['start_date', 'end_date'], axis=1).
```

Після цього код розділяє дані на вхідні ознаки «X» та цільову змінну «y». Вхідні ознаки містяться у всіх стовпцях, крім “target”, а цільова змінна знаходиться у стовпці «target».

```
X = data.drop('target', axis=1)
y = data['target'].
```

Далі дані розділяються на тренувальний та тестовий набори за допомогою “train_test_split()”. 80% даних використовується для тренування моделі, а 20% - для тестування.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42).
```

Потім створюється екземпляр класу “RandomForestClassifier” з параметрами «n_estimators=100» (кількість дерев у лісі) та «random_state=42» (для відтворюваності результатів). Модель навчається на тренувальному наборі даних за допомогою методу fit().

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train).
```

Після навчання модель використовується для передбачення цільової змінної на тестовому наборі даних за допомогою методу predict(). Передбачені значення зберігаються у змінній y_pred.

```
y_pred = rf_model.predict(X_test).
```

Далі, код оцінює якість моделі, розраховуючи метрики accuracy, precision, recall та f1-score. Ці метрики порівнюють передбачені значення (y_pred) з фактичними значеннями “y_test” і виводять результати на екран.

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-score: {f1}").
```

Далі цю модель буде збережено за допомогою методу «dump» з модулю “joblib”, що дозволить використовувати вже навчену модель без необхідності щоразу навчати її.

```
dump(rf_model, 'model.joblib').
```

Цей код демонструє процес побудови моделі випадкового лісу для класифікації на основі даних з кількох CSV-файлів, попередню обробку даних, розділення даних на тренувальний та тестовий набори, навчання моделі,

передбачення цільової змінної та оцінку якості моделі за допомогою різних метрик.

3.5. Програмна реалізація модулів серверної частини

Хоча реалізація інших модулів серверної частини є необхідною для функціонування WEB-сервісу, вона так чи інакше залежить від результатів модуля класифікації, тому її опис не є таким важливим, як захист персональних даних користувачів та забезпечення конфіденційності їхньої фінансової інформації є пріоритетним завданням. Без надійної системи авторизації та автентифікації, навіть найбільш досконалі алгоритми прогнозування можуть виявитися марними. Тому, далі буде особливо детально описано програмну реалізацію модуля авторизації.

Для захисту даних користувача було реалізовано модуль авторизації що містить такі кінцеві точки взаємодії: sign-in, sign-up, login, access-token, refresh-token. Далі буде описано взаємодію та логіку опрацювання кінцевої точки sign-in.

Клас «SignInView» успадковується від класу «APIView» і представляє собою представлення для входу користувача в систему. Спочатку визначається дозвіл доступу «permission_classes = (AllowAny,)», який дозволяє будь-якому користувачу, незалежно від автентифікації, мати доступ до цього представлення.

```
class SignInView(APIView):
    permission_classes = (AllowAny,)
```

Далі, за допомогою декоратора «@swagger_auto_schema», налаштовується автоматична генерація документації Swagger для цього представлення. Параметри декоратора включають:

- request_body=SignInSerializer() – вказує, що запит повинен містити тіло, яке відповідає серіалізатору SignInSerializer.
- Responses – визначає очікувані відповіді від цього представлення. У даному випадку, при успішному запиті.

```
@swagger_auto_schema(
        request_body=SignInSerializer(),
        responses={
            200: "Response { 'phone_number': string }\n"
            "OTP code would be sent to mobile number of the user.
"
            "From user perspective, OTP verification screen would
be shown AND only after "
            "successful verification the user would be signed
in.",
        },
    )
```

Метод «POST» визначає логіку обробки POST-запиту до цього представлення. Спочатку створюється екземпляр серіалізатора «SignInSerializer» з даними запиту «request.data». Потім викликається метод «is_valid(raise_exception=True)» серіалізатора, який перевіряє валідність даних. Якщо дані не пройшли валідацію, буде викинуто виключення «ValidationError». Якщо дані пройшли валідацію, метод повертає об'єкт «Response» з даними серіалізатора «serializer.data» та статусом «HTTP_200_OK», що означає успішну обробку запиту.

```
    def post(self, request):
            serializer = SignInSerializer(data=request.data)
            serializer.is_valid(raise_exception=True)
            return Response(serializer.data, status.HTTP_200_OK)
```

Таким чином, цей код реалізує представлення для входу користувача в систему, приймає дані запиту, виконує валідацію за допомогою серіалізатора «SignInSerializer».

Оскільки наш WEB-сервіс використовує чутливі данні користувача для реєстрації то потрібно реалізувати додатковий аналіз даних для реєстрації нового користувача. Наступним кроком буде реалізація серіалізатора для реєстрації нового користувача.

Клас «UserRegisterSerializer» є серіалізатором Django REST Framework, який використовується для реєстрації нового користувача. Спочатку визначаються атрибути серіалізатора:

– id (serializers.UUIDField): Унікальний ідентифікатор користувача, який є тільки для читання.
– first_name (serializers.CharField): Ім'я користувача, яке є обов'язковим і повинно мати мінімальну довжину 2 символи.
– last_name (serializers.CharField): Прізвище користувача, яке є обов'язковим і повинно мати мінімальну довжину 2 символи.
– email (serializers.EmailField): Адреса електронної пошти користувача, яка є обов'язковою і повинна мати мінімальну довжину 4 символи.
– password (serializers.CharField): Пароль користувача, який є обов'язковим, повинен мати мінімальну довжину 2 символи і проходити валідацію за допомогою validators.validate_password. Цей атрибут є тільки для запису.

Далі визначається внутрішній клас Meta, який містить метадані серіалізатора:

– model (User): Модель Django, яка використовується для реєстрації користувача.
– fields (list): Список полів, які включені в серіалізатор.

```
class UserRegisterSerializer(serializers.ModelSerializer):
    id = serializers.UUIDField(read_only=True)
    first_name      =      serializers.CharField(min_length=2,
required=True)
    last_name       =      serializers.CharField(min_length=2,
required=True)
    email        =        serializers.EmailField(min_length=4,
required=True)
    password       =       serializers.CharField(min_length=2,
required=True,      validators=[validators.validate_password],
write_only=True)
```

```
    class Meta:
        model = User
        fields = ["id", "first_name", "last_name", "email",
"password", "phone_number", "language"].
```

Таким чином, клас UserRegisterSerializer забезпечує серіалізацію та валідацію даних для реєстрації нового користувача в системі з використанням Django REST Framework.

3.6. Висновок до розділу 3

У даному розділі було детально описано процес програмної реалізації модуля класифікації досяжності фінансових цілей накопичення з використанням методу машинного навчання Random Forest. Було обґрунтовано вибір мови програмування Python та бібліотеки Scikit-learn як оптимальних інструментів для реалізації модуля класифікації, враховуючи їх простоту використання, швидкодію та ефективність обробки даних помірного розміру.

Представлено детальний опис набору даних параметрів цілей для навчання та тестування модуля класифікації, який містить 21153 екземплярів даних, серед яких 12831 є досяжними, а 8322 – недосяжними. Набір даних включає різноманітні атрибути, такі як початкова ціль накопичення, щомісячний дохід, заощадження та витрати в різних категоріях.

Також у розділі було коротко згадано про програмну реалізацію модулів серверної частини, таких як модуль авторизації та серіалізатор для реєстрації нового користувача, що забезпечують безпеку та коректну обробку даних користувачів.

Таким чином, у третьому розділі було успішно реалізовано модуль класифікації досяжності фінансових цілей накопичення та ключові аспекти програмної реалізації серверної частини WEB-сервісу, що забезпечує надійну основу для подальшої інтеграції та розгортання системи.

4. ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ СЕРВЕРНОЇ ЧАСТИНИ ТА МОДУЛЯ КЛАСИФІКАЦІЇ ДОСЯЖНОСТІ МЕТИ WEB-СЕРВІСУ ПРОГНОЗУВАННЯ ОПТИМАЛЬНИХ КРОКІВ ДЛЯ ДОСЯГНЕННЯ МЕТИ

Перш за все потрібно упевнитись, що розроблена серверна частини правильно функціонує, для цього необхідно здійснити запит класифікації нової мети накопичення. Даний запит повинен приймати всі параметри необхідні для класифікації мети на досяжність. Після успішного виконання запити тіло відповіді має містити інформацію про досяжність – клас досяжності. Далі проведемо запит та проаналізуємо тіло відповіді.

На рисунку 4.1 зображено виконання запиту на створення нової мети накопичення та тіло відповіді.

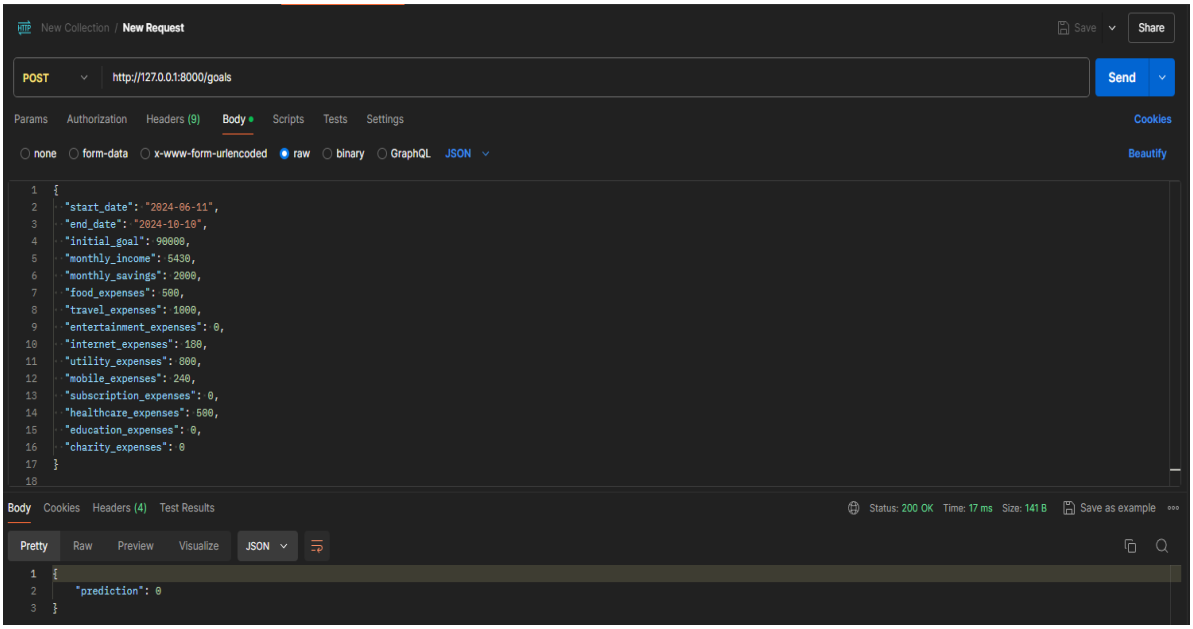


Рисунок 4.1 – Результат виконання запиту класифікації нової мети.

Отже, даний запит містив усі необхідні параметри для визначення досяжності мети, такі як початкова сума накопичення, щомісячний дохід, витрати в різних категоріях тощо.

Після успішного виконання запиту, отримано відповідь від серверної частини. Отже тестування серверної частини показало, що відповідь містить ключову інформацію про досяжність мети – клас досяжності, який вказує на те, чи є задана мета накопичення досяжною за вказаними параметрами. А отже, серверна частина функціонує правильно та готова до подальшої роботи.

Для того щоб упевнитись, що розробка пройшла успішно, проведемо тестування модуля класифікації. Після завантаження набору та нормалізації даних, видалення непотрібних атрибутів, програма видає статистичні показники набору даних – див. табл. 4.1:

Таблиця 4.1 – Статистичні показники набору даних.

	count	**mean**	**std**	**min**	**max**
initial_goal	21153	20870.28	3129.26	15445.00	26123.00
monthly_income	21153	8181.13	958.60	6524.00	10117.00
monthly_savings	21153	1776.55	683.27	316.00	3366.00
food_expenses	21153	1280.34	442.03	503.00	2634.00
travel_expenses	21153	503.97	290.77	0.00	1261.00
entertainment_expenses	21153	453.71	207.36	100.00	826.00
internet_expenses	21153	124.60	42.66	50.00	261.00
utility_expenses	21153	324.76	100.19	155.00	516.00
mobile_expenses	21153	90.45	34.90	30.00	176.00
subscription_expenses	21153	177.52	71.60	50.00	337.00
healthcare_expenses	21153	494.54	177.69	178.00	824.00
education_expenses	21153	507.85	289.35	0.00	1045.00
charity_expenses	21153	251.40	145.75	0.00	547.00
target	21153	0.61	0.39	0.00	1.00

Подана таблиця була вирахувана автоматично, за допомогою програмних засобів, таких як pandas, та містить наступні характеристики для кожного з атрибутів:

- count – кількість значень атрибута. У цьому випадку, кількість записів для кожного атрибута дорівнює 21153.
- mean – середнє арифметичне значення атрибута.
- std – стандартне відхилення, міра розкиду значень атрибута відносно середнього.
- min – мінімальне значення атрибута в наборі даних.
- max – максимальне значення атрибута в наборі даних.

Атрибут "target" – цільова змінна, яка приймає значення 0 або 1. Її середнє значення 0.61 вказує, що приблизно 61% цілей в наборі даних є досяжними.

Для подальшого тестування та визначення достовірності модуля класифікації було підготовлено тестовий набір даних набір даних, було використано 200 екземплярів даних з яких: 150 є досяжними, а інші. Результати тестування модуля класифікації досяжності мети представлено у табл. 4.2

Таблиця 4.2 – Результати тестування модуля класифікації досяжності.

	Результат класифікації	
	Achievable YES	Achievable NO
На вході досяжні екземпляри (загальна кількість досяжних екземплярів P=150)	Вірнопозитивний True Positive – $TP = 147$	Хибнонегативний False Negative – $FN = 3$
На вході не досяжні екземпляри (загальна кількість недосяжних екземплярів N=50)	Хибнопозитивний False Positive – $FP = 2$	Вірнонегативний True Negative – $TN = 48$

Отже, провівши тестування розробленого модуля класифікації мети на досяжність розрахуємо основні показники якості на основі даних табл. 4.2.

В машинному навчанні, оцінка класифікаторів є ключовим моментом для визначення їхньої ефективності. Для цього використовуються різні метрики, чотири з яких найпоширеніші:

Достовірність показує загальну частку правильно класифікованих зразків

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \cdot 100\% = \frac{147 + 48}{200} \cdot 100\% = 98\%$$

Прецизійність визначає частку позитивних прогнозів, які дійсно є позитивними.

$$Precision = \frac{TP}{TP + FP} \cdot 100\% = \frac{147}{147 + 2} \cdot 100\% = 98\%$$

Відновлення визначає частку дійсно позитивних зразків, які були правильно ідентифіковані.

$$Recall = \frac{TP}{TP + FN} \cdot 100\% = \frac{147}{147 + 3} \cdot 100\% = 98\%$$

F1-міра комбінує прецизійність та відновлення, даючи загальну оцінку продуктивності.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot 98 \cdot 98}{98{,}9 + 98} = 98\,\%$$

Було проведене ручне тестування розробленого класифікатора на основі Random Forest методу машинного навчання. Оскільки було використано бібліотеку sklearn, тому маємо можливість впевнитись в правильності ручного тестування протестувавши модель програмно (рис. 4.2)

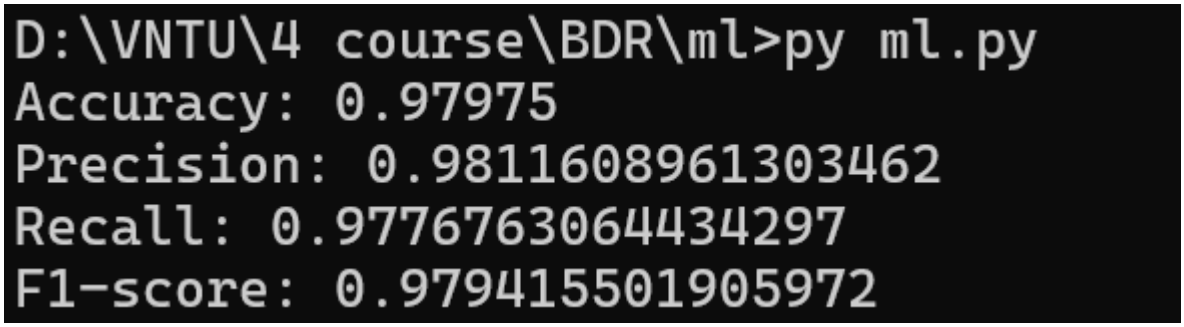


Рисунок 4.2 – Програмне тестування розробленого модуля класифікації.

З рисунку видно що ручне та програмне тестування майже однакові що свідчить про достовірність оцінки класифікатора.

Для доведення, що мету роботи було досягнуто, потрібно дослідити та порівняти отримані результати з результатами аналогу Sun Life Budget.

Таблиця 4.3 – Результати тестування аналога Sun Life Budget.

	Результат класифікації	
	Achievable YES	Achievable NO
На вході досяжні екземпляри (загальна кількість досяжних екземплярів P=150)	Вірнопозитивний True Positive – $TP = 147$	Хибнонегативний False Negative – $FN = 3$
На вході не досяжні екземпляри (загальна кількість недосяжних екземплярів N=50)	Хибнопозитивний False Positive – $FP = 6$	Вірнонегативний True Negative – $TN = 44$

Достовірність:

$$\text{Accuracy} = \frac{147 + 44}{200} \cdot 100\% = 95\%$$

Прецизійність:

$$Precision = \frac{147}{147 + 6} \cdot 100\% = 96\%$$

Відновлення:

$$Recall = \frac{147}{147 + 3} \cdot 100\% = 98\%$$

F1-міра:

$$F1 = \frac{2 \cdot 95 \cdot 98}{95 + 98} = 96{,}5\%$$

Провівши розрахунки потрібно продемонструвати різницю між цими, що було зроблено в табл. 4.4:

Таблиця 4.4 – Порівняння показників якості розробленого модуля з аналогом Sun Life Budget

	Розроблений модуль	Аналог (Sun Life Budget)
Accuracy	98%	95%
Precision	98%	96%
Recall	98%	98%
F1	98%	96,5%

З табл. 4.4 видно, що розроблений модуль класифікації має достовірність класифікації 98%, а аналог 95%, тобто розроблений модуль має на 3% кращий параметр достовірності. Отже, мета роботи – досягнута, достовірність визначення класу мети підвищена на 3% у розробленому модулі.

Отже, розділі було проведено тестування та аналіз результатів роботи модуля класифікації досяжності мети та серверної частини. Для тестування серверної частини було виконано запит, створення нової мети, та продемонстровано його працездатність. Для тестування модуля класифікації було використано спеціально підготовлений набір даних, що складається з 200 екземплярів, серед яких 150 є досяжними та належать до класу 1, а інші 50 – недосяжними та належать до класу 0.

Результати тестування показали високу ефективність розробленого модуля класифікації. Було розраховано основні показники якості, такі як достовірність (accuracy), прецизійність (precision), відновлення (recall) та F1-міра. Розроблений модуль продемонстрував достовірність класифікації на рівні 98%, прецизійність – 98%, відновлення – 98% та F1-міру – 98%.

Також було проведено порівняння результатів розробленого модуля з аналогом. Тестування аналога на тому ж наборі даних показало достовірність класифікації 95%, прецизійність – 96%, відновлення – 98% та F1-міру – 96,5%. Таким чином, розроблений модуль класифікації перевершив аналог за всіма показниками якості, зокрема, достовірність класифікації була підвищена на 3%.

Отримані результати підтверджують, що мета роботи – підвищення достовірності визначення класу досяжності мети сервісу прогнозування кроків досягнення мети накопичення коштів за рахунок використання машинного навчання – була успішно досягнута. Розроблений модуль класифікації на основі методу Random Forest продемонстрував високу ефективність та перевагу над існуючим аналогом.

ВИСНОВКИ

Поставлені перед бакалаврською дипломною роботою задачі виконано в повному обсязі, а саме:

- Проаналізовано відомі методи визначення класу досяжності мети та обрано напрямок дослідження.
- Розроблено алгоритм роботи модуля класифікації на основі обраного методу та спроєктовано серверну частину WEB-сервісу.
- Реалізовано модуль класифікації та серверну частину WEB-сервісу на мові Python з використанням фреймворку Django та бібліотеки Scikit-learn.
- Проведено тестування розробленого програмного модуля.

В ході виконання бакалаврської роботи було проведено детальний аналіз предметної області, розглянуто постановку задачі класифікації досяжності мети накопичення коштів та обґрунтовано вибір методу машинного навчання Random Forest як найбільш перспективного для її вирішення.

Розроблено алгоритм роботи модуля класифікації та спроєктовано серверну частину WEB-сервісу з використанням UML діаграм класів, компонентів, розгортання, послідовності та активності.

Здійснено програмну реалізацію модуля класифікації та серверної частини WEB-сервісу на мові Python з використанням фреймворку Django для серверної частини та бібліотеки Scikit-learn для реалізації методу Random Forest. Описано набір даних параметрів цілей для навчання та тестування модуля класифікації.

Проведено ретельне тестування розробленого модуля класифікації на спеціально підготовленому наборі даних. Результати тестування показали високу ефективність розробленого модуля, який продемонстрував достовірність класифікації на рівні 98%, перевершивши аналог Sun Life Budget на 3%. Таким чином, мета роботи – підвищення достовірності визначення класу досяжності мети сервісу прогнозування оптимальних кроків для досягнення мети за рахунок машинного навчання – була успішно досягнута.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Перспективи використання алгоритмів прогнозування оптимальних кроків для досягнення цілі [Електронний ресурс] / Д.С. Щетінін, О. С. Прудивус // Матеріали конференції ВНТУ електронні наукові видання, Молодь в науці: дослідження, проблеми, перспективи (МН-2024) – Електрон. текст. дані. – 2024. – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/21700
2. Simplifying Decision Trees: A Survey1 – Leonard A. Breslow and David W. Aha [Електроний ресурс] Режим доступу: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a0e3e28c7635e69a1418d09030fcbdc2ff49a517
3. LOGISTIC REGRESSION Joseph M. Hilbe Arizona State University [Електроний ресурс] Режим доступу: https://encyclopediaofmath.org/images/6/69/Logistic_regression.pdf
4. Consistency of Random Forests and Other Averaging Classifiers Gerard ´ Biau Luc Devroye Gabor ´ Lugosi [Електроний ресурс] Режим доступу: https://www.jmlr.org/papers/volume9/biau08a/biau08a.pdf
5. Support Vector Machines David Meyer Technische Universit¨at Wien, Austria [Електроний ресурс] Режим доступу: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=4850faab0aab5c4b40fcdd5a77d9e7626a163db5
6. Naive Bayes classifiers – Kevin P. Murphy [Електроний ресурс] Режим доступу: https://datajobs.com/data-science-repo/Naive-Bayes-%5BKevin-Murphy%5D.pdf
7. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ: БАЗОВІ ПОЛОЖЕННЯ – І.А. Терейковський, Д.А. Бушуєв, Л.О. Терейковська Grefenstette [Електроний ресурс] Режим доступу: https://ela.kpi.ua/server/api/core/bitstreams/9fee52b6-83fc-4e99-8541-c2767f634c7c/content

8. What are the used UML diagrams? A Preliminary Survey [Електроний ресурс] Режим доступу: https://ceur-ws.org/Vol-1078/paper1.pdf
9. The Python Tutorial¶ [Електроний ресурс] Режим доступу: https://docs.python.org/3/tutorial/index.html
10. ECMAScript® 2025 Language Specification [Електроний ресурс] Режим доступу: https://tc39.es/ecma262/
11. Java Documentation [Електроний ресурс] Режим доступу: https://docs.oracle.com/en/java/
12. Frameworks= (Components+Patterns) [Електроний ресурс] Режим доступу: https://dl.acm.org/doi/pdf/10.1145/262793.262799
13. Scikit-learn: Machine Learning in Python [Електроний ресурс] Режим доступу: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?ref=https:/
14. Pandas documentation [Електроний ресурс] Режим доступу: https://pandas.pydata.org/docs/
15. Analysis of a Random Forests Model [Електроний ресурс] Режим доступу: https://www.jmlr.org/papers/volume13/biau12a/biau12a.pdf
16. ВНТУ – Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс]: Стаття - Режим доступу: https://iq.vntu.edu.ua/method/getfile.php?fname=124285.pdf&x=1&card_id=46522&id=124285
17. Барановський В. С. Програмний модуль для класифікації тональності речень [Електронний ресурс] / В. С. Барановський, О. К. Колесницький // Матеріали XLVII науково-технічної конференції підрозділів ВНТУ, Вінниця, 14-23 березня 2018 р. – Електрон. текст. дані. – 2018. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2018/paper/view/5157

ДОДАТОК А

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Програмний WEB-сервіс прогнозування оптимальних кроків для досягнення мети. Частина 2. Серверна частина

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 91,75% Схожість 8,25%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку Озеранський В.С.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Прудивус О.С.

Керівник роботи Колесницький О.К.

ДОДАТОК Б

Лістинг програми

model.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from joblib import dump, load

# Loading data from files
files = [""]
data = pd.concat([pd.read_csv(file) for file in files], ignore_index=True)

# Converting dates to datetime objects
data['start_date'] = pd.to_datetime(data['start_date'])
data['end_date'] = pd.to_datetime(data['end_date'])

# Dropping unnecessary columns
data = data.drop(['start_date', 'end_date'], axis=1)

# Splitting data into features and target variable
X = data.drop('target', axis=1)
y = data['target']

# Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initializing and training the RandomForestClassifier model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
rf_model.fit(X_train, y_train)

# Predicting on the test set and calculating metrics
y_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Printing model accuracy metrics
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-score: {f1}")

# Saving the model
dump(rf_model, 'model.joblib')

# Prediction for new data
new_data = pd.DataFrame({
    'initial_goal': [20000],
    'monthly_income': [1500],
    'monthly_savings': [100],
    'duration': [24]
})
rf_model = load('model.joblib')

# Predicting for new data and printing the result
prediction = rf_model.predict(new_data)
print(f"Prediction for new data: {prediction}")
```

authorization.py

```
class UserRegisterView(CreateAPIView):
    # View for user registration
    queryset = User.objects.all()
```

```
    serializer_class = UserRegisterSerializer
    permission_classes = (AllowAny,)

class SignInView(APIView):
    # View for user sign-in
    permission_classes = (AllowAny,)

    @swagger_auto_schema(
        request_body=SignInSerializer(),
        responses={
            200: "Response { 'phone_number': string }\n"
                 "OTP code would be sent to mobile number of the
user. "
                 "From user perspective, OTP verification screen
would be shown AND only after "
                 "successful verification the user would be signed
in.",
        },
    )
    def post(self, request):
        # Handles user sign-in process
        serializer = SignInSerializer(data=request.data)
        serializer.is_valid(raise_exception=True)
        return Response(serializer.data, status=status.HTTP_200_OK)

class ResetPasswordByEmailView(APIView):
    # View for resetting password via email
    serializer_class = ResetPasswordByEmailSerializer
    permission_classes = (AllowAny,)

    @swagger_auto_schema(
        request_body=ResetPasswordByEmailSerializer(),
        responses={
            200: "In this case Email provider is triggered with the
following text and redirect to: "
```

```
                "brockers://reset-password?token=token",
            400: "This email is not registered. Please, double-check
your input or create an account",
        },
    )
    def post(self, request):
        # Handles password reset via email
        serializer    =    self.serializer_class(data=request.data,
context={"language": request.language})
        serializer.is_valid(raise_exception=True)
        return Response(serializer.data, status=status.HTTP_200_OK)

class UserDeleteAPIView(APIView):
    # View for marking user for deletion
    permission_classes = (IsPrivateUser,)

    def delete(self, request, _args, *_kwargs):
        """
        Marks the authenticated user as under deletion and creates
a DELETE_USER task for them.
        Returns a 204 No Content response.
        """
        request.user.is_under_deletion = True
        request.user.save(update_fields=["is_under_deletion"])
        Task.objects.create(user=request.user,
type=TaskType.DELETE_USER.value)
        return Response(status=status.HTTP_204_NO_CONTENT)

@method_decorator(
    name="post",
    decorator=swagger_auto_schema(
        operation_description="Endpoint to takes a refresh token and
blacklists it.",
        responses={status.HTTP_205_RESET_CONTENT:                "",
status.HTTP_400_BAD_REQUEST: ""},
```

```
        tags=["auth"],
    ),
)
class LogoutView(APIView):
    # View for user logout
    def post(self, request):
        try:
            request.user.block_refresh_token()
            return Response(status=status.HTTP_205_RESET_CONTENT)
        except Exception:
            return Response(status=status.HTTP_400_BAD_REQUEST)
```

serializers.py

```
class UserRegisterSerializer(serializers.ModelSerializer):
    """
    Serializer for registering a new user.
    Attributes:
        id (serializers.UUIDField): Unique identifier for the user
(read-only).
        first_name (serializers.CharField): First name of the user
(required).
        last_name (serializers.CharField): Last name of the user
(required).
        email (serializers.EmailField): Email address of the user
(required).
        password (serializers.CharField): Password for the user
(required, write-only).
        role (serializers.ChoiceField): Role of the user (required,
write-only).
    Meta:
        model (User): Model used for user registration.
        fields (list): List of fields included in the serializer.
    Methods:
        validate_email(email): Validates the uniqueness of the email
address.
```

validate_phone_number(phone_number): Validates the uniqueness of the phone number.

create(validated_data): Creates a new user using the provided data.

"""

```
    id = serializers.UUIDField(read_only=True)
    first_name = serializers.CharField(min_length=2, required=True)
    last_name = serializers.CharField(min_length=2, required=True)
    email = serializers.EmailField(min_length=4, required=True)
    password = serializers.CharField(min_length=2, required=True, validators=[validators.validate_password], write_only=True)
    role = serializers.ChoiceField(choices=ProfileRoles.CHOICES, required=True, write_only=True)

    class Meta:
        model = User
        fields = ["id", "first_name", "last_name", "email", "password", "phone_number", "role", "title", "language"]

    def validate_email(self, email):
        if email and User.objects.filter(email__iexact=email, is_active=True).exists():
            raise serializers.ValidationError([_("A user is already registered with this email address.")])
        return email

    def validate_phone_number(self, phone_number):
        email = self.initial_data.get("email")
        if (
            phone_number
            and User.objects.filter(
                Q(phone_number__iexact=phone_number),
(Q(is_active=True) | ~Q(email__iexact=email))
```

```
            ).exists()
        ):
            raise serializers.ValidationError([_("A user is already
registered with this phone number.")])
        return phone_number

    def create(self, validated_data):
        user = User.objects.create_user(**validated_data)
        user.send_otp_code(user.phone_number)
        return user
```

ДОДАТОК В

ІЛЮСТРАТИВНА ЧАСТИНА

«ПРОГРАМНИЙ WEB-СЕРВІС ПРОГНОЗУВАННЯ ОПТИМАЛЬНИХ КРОКІВ ДЛЯ ДОСЯГНЕННЯ МЕТИ. ЧАСТИНА 2. СЕРВЕРНА ЧАСТИНА»

Виконав: студент 4 курсу, групи 1КН-20б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Прудивус О. С ..
(прізвище та ініціали)

Керівник: к.т.н., проф. каф. КН
Колесницький О. К .
(прізвище та ініціали)
«04» 06 2024 р.

Вінниця ВНТУ - 2024 рік

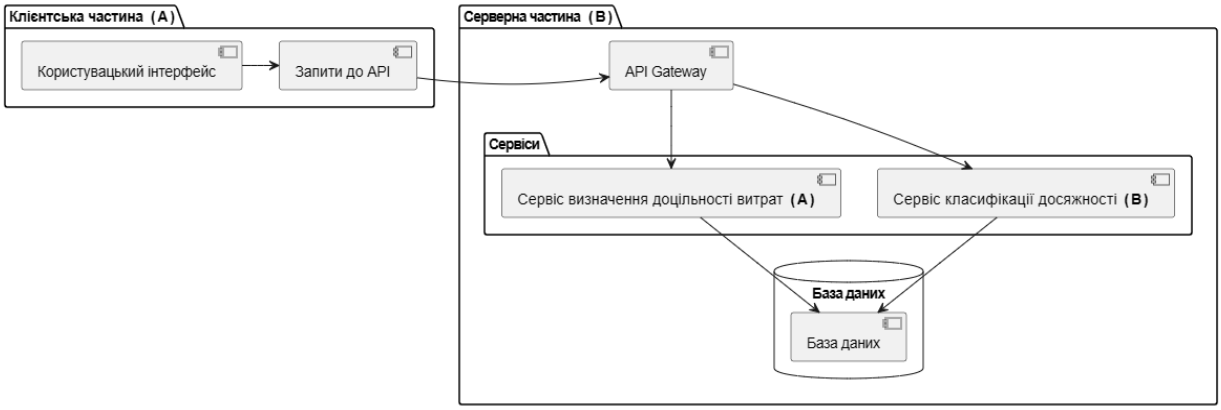


Рисунок В.1 – Загальний вигляд архітектури WEB-сервісу.

Рисунок В.2 – Схема алгоритму роботи модуля класифікатора визначення досяжності мети.

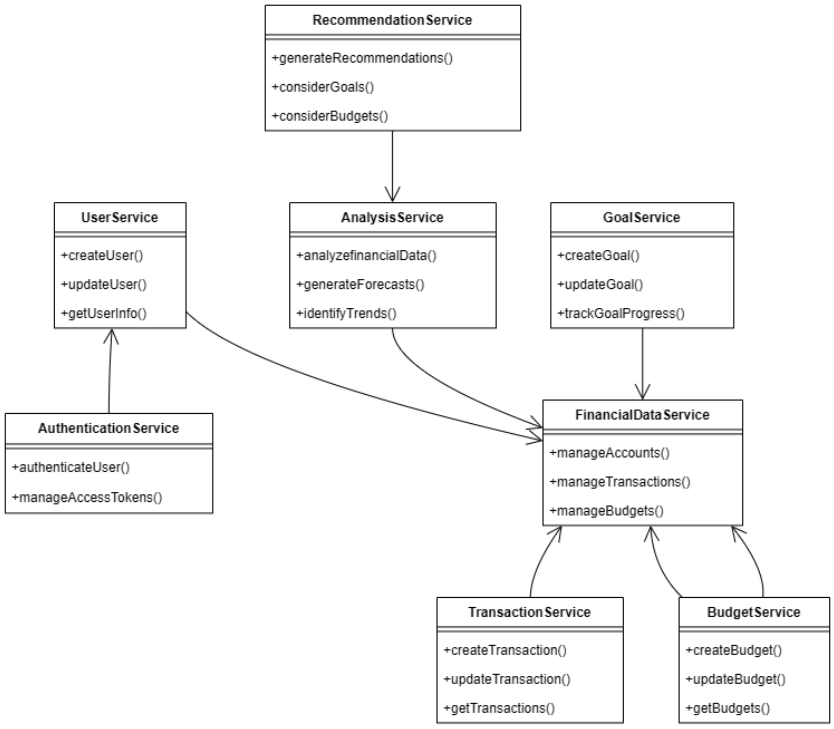


Рисунок В.3 – UML-діаграма класів серверної частини WEB-сервісу.

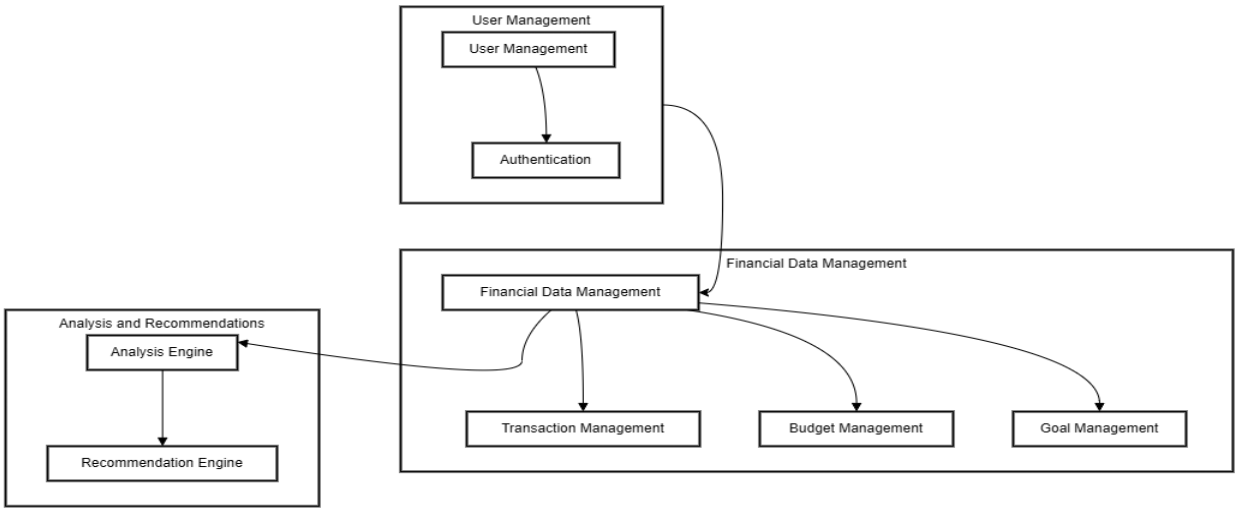


Рисунок В.4 – UML-діаграма компонентів серверної частини WEB-сервісу.

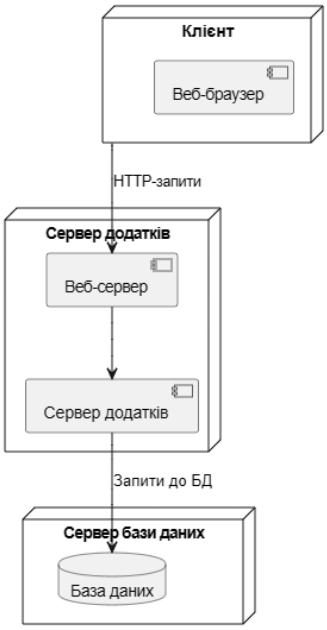


Рисунок В.5 – UML-діаграма розгортання серверної частини WEB-сервісу.

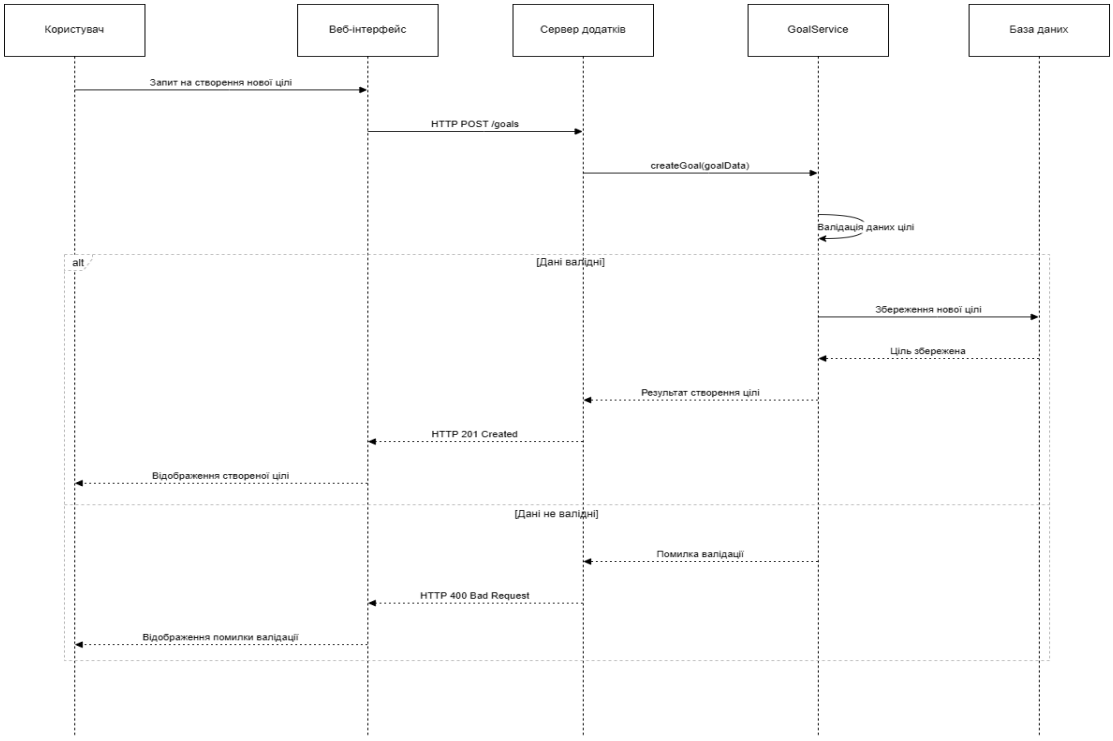


Рисунок В.6 – UML-діаграма послідовності створення нової цілі.

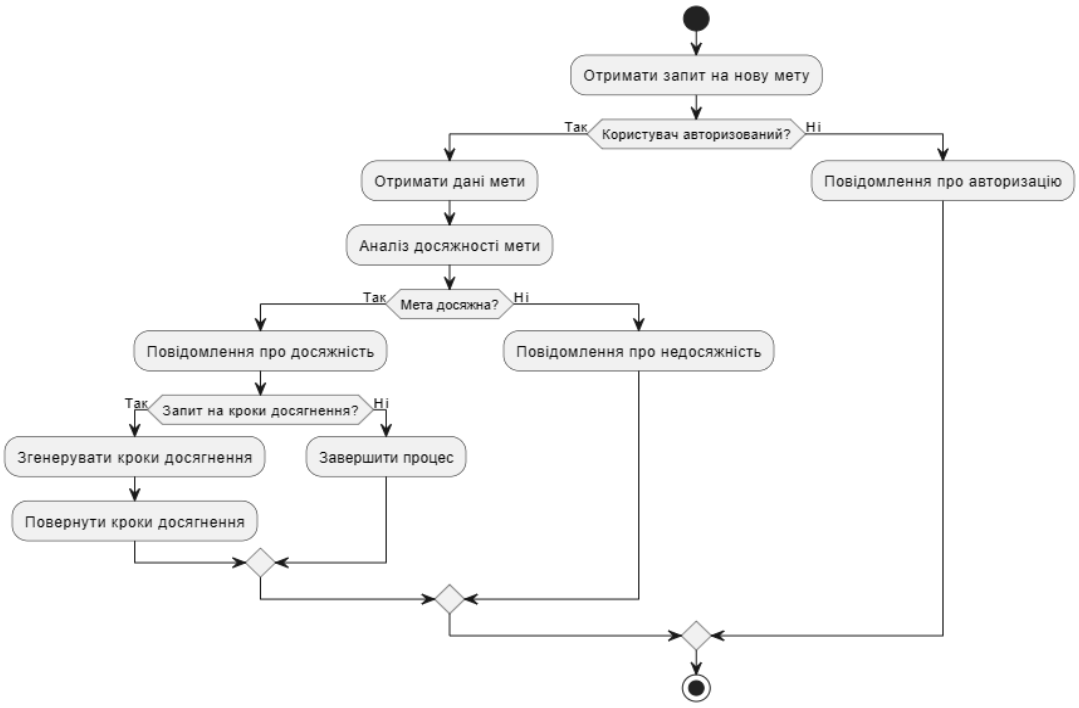


Рисунок В.7 – UML-діаграма активності серверної частини WEB-сервісу.