Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Системотехніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Дослідження та впровадження моделі логістичної регресії для прогнозування фінансової поведінки клієнтів у веб-застосунку «Банківський асистент»
(тема)

Виконав:
студент _2_ курсу, групи ІТПм-24-2
Гриб А. С._____
(прізвище, ініціали)
Спеціальність _122 Комп'ютерні науки_____
(код і повна назва спеціальності)
Тип програми __освітньо-наукова____
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційні технології проектування_____
(повна назва освітньої програми)
Керівник _____ст. викл. Яцик М. В._____
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ ____Гребеннік І. В.___
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерних наук_____

Кафедра_____Системотехніки_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність_____122 Комп'ютерні науки_____

(код і повна назва)

Тип програми _____освітньо-наукова_____

(освітньо-професійна або освітньо-наукова)

Освітня програма_____Інформаційні технології проектування_____

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Грибу Антону Сергійовичу_____

(прізвище, ім’я, по батькові)

1. Тема роботи Дослідження та впровадження моделі логістичної регресії для прогнозування фінансової поведінки клієнтів у веб-застосунку «Банківський асистент». затверджена наказом університету від _24_ __листопада_____2025 р. № 1058Ст

2. Термін подання студентом роботи до екзаменаційної комісії _12___ __грудня____ 2025 р.

3. Вихідні дані до роботи Об’єкт дослідження – комп’ютерна технологія з автоматизації оцінювання кредитних ризиків з побудовою та впровадженням прогностичної статистичної моделі. Функція: оцінювання ризиків дефолту позичальника на основі cDоціально-фінансових показників.

4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ. 4.2.1 Математичний апарат моделей. 4.2.2 Логістична регресія як метод класифікації в задачах прогнозування. 4.2.3 Метрики оцінки якості моделі. 4.2.4 Порівняння моделі логістичної регресії з іншими моделями машинного навчання. 4.3.1 Аналіз предметної області та постановка здачі. 4.3.2. Аналіз предметної області. 4.3.3 Огляд існуючих рішень і прототипів системи. 4.3.4

Постановка задачі розробки системи. 4.4.1 Побудова та оцінювання моделі. 4.4.2 Постановка задачі розробки системи. 4.4.3 Оцінка прогностичної здатності моделі. 4.4.4 Порівняння результатів з іншими моделями. 4.5.1 Побудова веб-застосунку та впровадження моделі. 4.5.2 Опис архітектури системи. 4.5.3 Обґрунтування вибору програмного сервісу. 4.5.4 Логічне і фізичне моделювання даних системи. 4.5.5 Розробка класів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) 5.1 Тема кваліфікаційної роботи. 5.2 Мета, об'єкт та предмет кваліфікаційної роботи. 5.3 Постановка завдання. 5.4 Аналіз предметної області, актуальність. 5.5 Аналіз предметної області, прототипи. 5.6 Вибір статистичної моделі. 5.7. Вибір статистичної моделі, логістична регресія. 5.8. Реалізована архітектура системи. 5.9. Архітектура серверної програми 5.10. Логічна модель даних системи. 5.11. Фізична модель даних системи. 5.12. Опис математичного апарату. 5.13. Опис математичного апарату, функція правдоподібності. 5.14. Опис математичного апарату, логарифмічна функція правдоподібності. 5.15. Опис математичного апарату, порог ймовірності. 5.16. Розробка алгоритму моделі логістичної регресії, аналіз значень дефолту, доходу та боргів. 5.17 Розробка алгоритму роботи моделі логістичної регресії, аналіз матриці кореляцій. 5.18 Оцінювання моделі логістичної регресії, матриця помилок. 5.19 Оцінювання моделей, TPR та FPR. 5.20. Веб-інтерфейс, авторизація. 5.21. Оцінювання моделі логістичної регресії, ROC-крива. 5.22. Веб-інтерфейс, створення запиту. 5.25. Веб-інтерфейс, перегляд звітів. 5.24. Висновки кваліфікацийої роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
	Видача завдання	24.11.2025	Виконано
	Аналіз літературних джерел за отриманим завданням	24.11.2025 - 27.11.2025	Виконано
	Вивчення математичного апарату моделей машинного навчання	27.11.2025 - 29.11.2025	Виконано

	Вивчення методів оцінювання моделей машинного навчання	29.11.2025 – 01.12.2025	Виконано
	Створення вимог розроблюваної системи	01.12.2025 - 03.12.2025	Виконано
	Розробка моделей машинного навчання	03.12.2025 - 05.12.2025	Виконано
	Порівняння отриманих результатів моделей	05.12.2025 - 06.12.2025	Виконано
	Побудова веб-застосунку	06.12.2025 - 07.12.2025	Виконано
	Оформлення пояснювальної записки	08.12.2025	Виконано

Дата видачі завдання _24__ __листопада_____2025 р.

Студент _____

(підпис)

Керівник роботи _____ _____ст. викл. Яцик М. В._____

(підпис) (посада, прізвище, ініціали)

Я як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Керівник кваліфікаційної роботи  ст. викл. Яцик М. В

РЕФЕРАТ

Пояснювальна записка до бакалаврської кваліфікаційної роботи: 63 с., 26 рис., 1 додаток, 25 джерел інформації.

ВЕБ-ЗАСТОСУНОК, PYTHON, SCIKIT-LEARN, МОДЕЛЬ ЛОГІСТИЧНОЇ РЕГРЕСІЇ, ROC, AUC

Об’єктом дослідження є комп’ютерна система підтримки прийняття рішень у банківській сфері для оцінювання фінансово-кредитних ризиків.

Предметом дослідження є методи та моделі інтелектуального аналізу даних, зокрема модель логістичної регресії, та програмні засоби їх застосування для автоматизації оцінки кредитоспроможності у веб-застосунку.

Метою роботи є розробка та впровадження веб-застосунку «Банківський асистент» з інтегрованою моделлю логістичної регресії для прогнозування фінансової поведінки клієнтів банку (ймовірності дефолту позичальників).

Для досягнення мети у роботі вирішуються наступні основні задачі: аналіз предметної області та існуючих підходів до кредитного скорингу; дослідження математичних основ логістичної регресії та альтернативних моделей класифікації; проєктування архітектури програмної системи та бази даних; реалізація клієнт-серверного; навчання і тестування моделі на наборі даних; оцінка якості моделі за допомогою метрик точності, матриці невідповідностей, ROC-кривої та AUC.

Методи дослідження: системний аналіз предметної області; методи статистичного аналізу і машинного навчання (логістична регресія, аналіз точності класифікації); об’єктно-орієнтоване проєктування програмного забезпечення; моделювання та методи оцінювання якості прогнозних моделей.

Результатом роботи є програмна система – веб-застосунок «Банківський асистент», що дозволяє автоматично оцінювати кредитоспроможність клієнтів на основі введених даних про них, використовуючи модель логістичної регресії.

Практичне значення одержаних результатів полягає в підвищенні ефективності управління кредитними ризиками за рахунок автоматизації процесу скорингу. Розроблений застосунок може бути інтегрований у реальні бізнес-процеси банківських установ для підтримки прийняття рішень щодо видачі кредитів. Використання моделі логістичної регресії забезпечує інтерпретованість результатів та відповідність регуляторним вимогам.

Наукова новизна роботи полягає у вдосконаленні методики оцінювання кредитоспроможності шляхом поєднання класичного статистичного підходу (логістичної регресії) з сучасними практиками розробки веб-застосунків, розширенні набору критеріїв оцінки якості моделей (ROC, AUC та ін.), а також проведенні порівняльного аналізу з альтернативними моделями класифікації.

ABSTRACT

Explanatory note of the bachelor's qualification work: 63 pages, 26 figures, 1 appendices, 25 sources of information.

WEB APP, PYTHON, SCIKIT-LEARN, LOGISTIC REGRESSION MODEL, ROC, AUC

The object of the research of the qualification work is computer technology for the development of an information system aimed at supporting managerial decision-making in the financial and banking spheres.

The object of the study is a computer decision support system in the banking sector for assessing financial and credit risks.

The subject of the study is methods and models of intelligent data analysis, in particular the logistic regression model, and software tools for their application to automate creditworthiness assessment in a web application.

The purpose of the work is to develop and implement the web application "Banking Assistant" with an integrated logistic regression model for predicting the financial behavior of bank clients (probability of borrower default).

To achieve the goal, the following main tasks are solved in the work: analysis of the subject area and existing approaches to credit scoring; study of the mathematical foundations of logistic regression and alternative classification models; design of the architecture of the software system and database; implementation of client-server; training and testing of the model on a data set; assessment of the quality of the model using accuracy metrics, discrepancy matrix, ROC curve and AUC.

Research methods: systematic analysis of the subject area; statistical analysis and machine learning methods (logistic regression, classification accuracy analysis); object-oriented software design; modeling and methods for assessing the quality of predictive models.

The result of the work is a software system - a web application "Banking Assistant", which allows you to automatically assess the creditworthiness of clients based on entered data about them, using the logistic regression model.

The practical significance of the results obtained is to increase the efficiency of credit risk management by automating the scoring process. The developed application can be integrated into real business processes of banking institutions to support decision-making on lending. The use of the logistic regression model ensures the interpretability of the results and compliance with regulatory requirements.

The scientific novelty of the work lies in improving the methodology for assessing creditworthiness by combining the classical statistical approach (logistic regression) with modern web application development practices, expanding the set of criteria for assessing the quality of models (ROC, AUC, etc.), as well as conducting a comparative analysis with alternative classification models.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

СУБД – Система управління базою даних

API – Application program interface

AUC – Area Under Curve

DDD – Domain Driven Design

HTML – Hypertext Markup Language

HTTP – Hypertext transfer protocol

JWT – JSON Web Token

ORM – Object-Relational Mapping

REST – Representational state transfer

ROC – Receiver Operator Characteristics

SQL – Structured Query Language

UOW – Unit of Work

URI – Unique Resource Identifier

ВСТУП

Підвищення ефективності управління кредитними ризиками є актуальною задачою для фінансових установ. У сучасних умовах банки прагнуть автоматизувати оцінювання фінансової поведінки клієнтів, зокрема їх кредитоспроможності, з метою прискорення ухвалення рішень та мінімізації ризику дефолтів позичальників. Автоматизовані системи кредитного скорингу дозволяють банкам швидше приймати рішення про видачу чи відмову в кредиті на основі об'єктивних даних про клієнта, таких як рівень доходів, кредитна історія, поточні зобов'язання тощо. Впровадження таких систем допомагає зменшити людський фактор, підвищити точність прогнозів і побудувати оптимальну стратегію роботи з кожним клієнтом (наприклад, встановлення відповідної відсоткової ставки чи ліміту кредитування залежно від ймовірності дефолту). Таким чином, тема автоматизації оцінки фінансової поведінки клієнтів (кредитного скорингу) є надзвичайно актуальною та має важливе практичне значення для банківських і мікрофінансових організацій.

Аналіз сучасного стану проблеми показує, що більшість банків використовують або впроваджують власні системи кредитного скорингу. Класичним підходом до прогнозування ймовірності дефолту є статистичне моделювання на основі даних про позичальників. Зокрема, логістична регресія вже понад кілька десятиліть залишається стандартом де-факто у побудові моделей кредитного ризику. Це пояснюється тим, що логістична регресія забезпечує достатньо високу прогностичну точність, простоту реалізації та головне – інтерпретованість результатів для експертів у фінансах.

Моделі на основі логістичної регресії зрозумілі з точки зору впливу кожного фактору на ризик (через оцінку коефіцієнтів регресії та їх перетворення до відношення шансів), тому вони широко прийняті регуляторами і використовуються банками як еталонні при кредитному скорингу.

Водночас, останніми роками з розвитком великих даних та машинного

навчання з’явилися і нові підходи. Дослідження показують, що більш складні алгоритми (дерева рішень, ансамблеві методи – випадкові ліси, бустинг, нейронні мережі) іноді перевершують логістичну регресію за точністю прогнозу, особливо на великих масивах різнорідних даних.

Проте ці методи часто поступаються логістичній регресії в плані інтерпретованості та простоти, а також потребують значно більше даних для навчання.

Тому на практиці логістична регресія залишається базовим підходом і своєрідним “бенчмарком” для моделей кредитного ризику, а більш складні моделі нерідко застосовуються як надбудова або для валідації результатів.

Основна мета роботи полягає у проєктуванні та створенні клієнт-серверного веб-застосунку «Банківський асистент», який автоматизує процес оцінювання фінансової поведінки клієнтів банку на основі моделі логістичної регресії. Застосунок повинен забезпечувати введення даних про клієнта, розрахунок ймовірності його дефолту (неповернення кредиту) за допомогою навченої моделі, формування звіту з результатами оцінки та рекомендаціями для працівника банку, а також зберігання історії оцінок у базі даних. Для досягнення поставленої мети в роботі вирішено ряд взаємопов’язаних задач:

- проаналізувати предметну область кредитного скорингу, виявити сучасні методи та системи автоматизованого оцінювання кредитоспроможності;
- дослідити теоретичні основи логістичної регресії як методу двохкласифікаційного прогнозування, розглянути її математичну модель, методи оцінювання параметрів та метрики якості; порівняти логістичну регресію з іншими моделями (дерева рішень, випадковий ліс, тощо) для обґрунтування вибору моделі;
- сформулювати постановку задачі розробки інформаційної системи: визначити вимоги до функцій “Банківського асистента” та його користувачів, окреслити архітектуру клієнт-серверного застосунку;

- провести навчання і тестування моделі логістичної регресії на вибраному наборі даних клієнтів (тестовому або реальному датасеті кредитної історії); оцінити якість моделі за допомогою метрик класифікації: точності, повноти, специфічності, побудувати матрицю невідповідностей (confusion matrix) та ROC-криву, обчислити AUC; порівняти якість з альтернативними моделями (наприклад, дерево рішень) для демонстрації переваг або недоліків логістичної регресії.

Наукова новизна роботи полягає у поєднанні методів інтелектуального аналізу даних з практичною реалізацією веб-сервісу для кредитного скорингу. Таким чином, робота робить вклад у адаптацію відомих методів машинного навчання до задач кредитного скорингу з урахуванням специфіки банківських даних і вимог.

Практичне значення одержаних результатів полягає у створенні програмного продукту, який може бути використаний у фінансових організаціях для автоматизації оцінювання кредитних ризиків. Впровадження веб-застосунку «Банківський асистент» дозволить скоротити час обробки кредитних заявок, зробити процес ухвалення рішення більш прозорим і обґрунтованим, а також знизити рівень проблемних кредитів завдяки більш точному прогнозуванню дефолтів. Така система може бути інтегрована у внутрішні інформаційні системи банку з мінімальними витратами часу завдяки використанню стандартного REST API.

Отримані результати підтвердили доцільність застосування моделі логістичної регресії в задачах кредитного скорингу та показали високі показники якості моделі (AUC ~0.91) на тестових даних. Програмний код веб-застосунку пройшов тестування і може бути використаний як прототип для подальшого промислового впровадження.

1. МАТЕМАТИЧНИЙ АПАРАТ МОДЕЛЕЙ

1.1 Логістична регресія як метод класифікації в задачах прогнозування

Для побудови системи оцінювання можна використовувати різні статистичні моделі [1, 2]. Ці моделі можна отримати за допомогою лінійної регресії, логістичної регресії, дерев рішень, нейронних мереж тощо. Однак логістична регресія є найчастіше використовуваною математичною моделлю на практиці для побудови систем оцінювання в банках.

Лінійна регресія – модель, що відображає залежності змінної y від однієї іншої або від багатьох змінних x, використовуючи функцію $y = kx + b$.

У регресійному аналізі незалежні змінні відомі як предикторні змінні або регресори, залежні – як критеріальні [3, 4].

При простій лінійній регресії можна аналізувати зв'язок між одним вхідним і одним вихідним елементом. Для цього застосовується лінійна функція $y = kx + b$, будується створюється відповідна пряма, що отримала назву лінії регресії. На рисунку 1.1 представлена ілюстрація лінійної регресії, де червона лінія - це лінія регресії

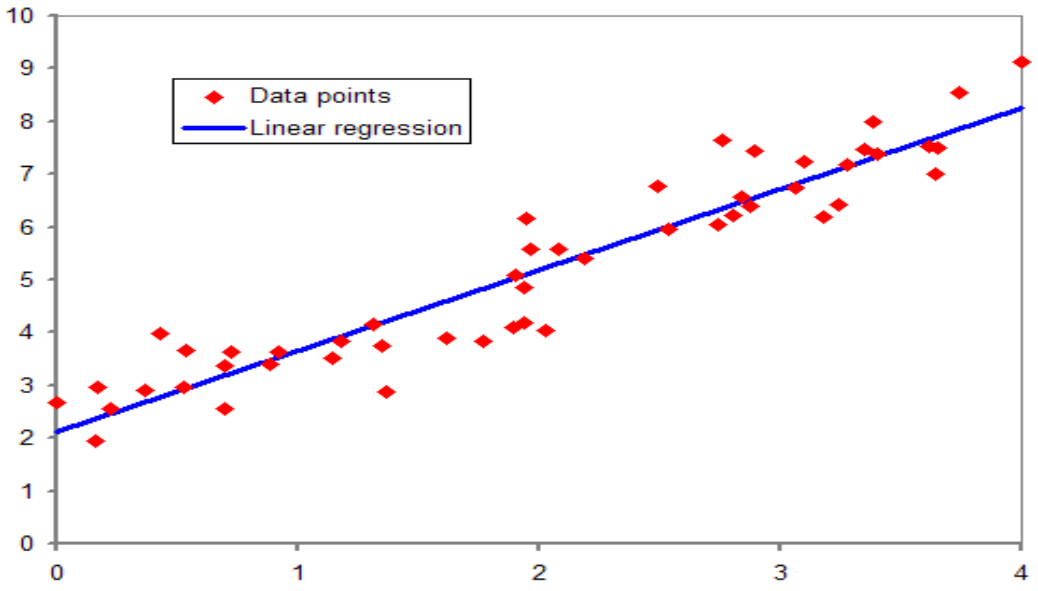

Рисунок 1.1 – Приклад лінійної регресії

Параметрами моделі є коефіцієнти k та b, які визначаються так, щоб сума квадратів відхилень точок, що відповідають фактичним даним, від регресійної лінії була найменшою. У ситуаціях, коли потрібно дослідити зв’язок між кількома незалежними змінними та однією залежною, використовується множинна лінійна регресія, яка описується такою формулою:

$$Y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n,$$

де n – число вхідних змінних.

Лінійна регресія служить для створення моделей лінійних зв'язків між безперервною залежною змінною та набором незалежних змінних. Проте, під час аналізу даних часто виникають ситуації, коли цільова змінна є категоріальною, і тоді застосування лінійної регресії стає ускладненим.

У цьому контексті, при дослідженні взаємозв'язків між групою вхідних змінних та категоріальною змінною отримала популярність логістична регресія. На відміну від традиційної регресії, в логістичній регресії не здійснюється прогнозування значення числової змінної на основі вибірки початкових значень. Значення функції в даному випадку є ймовірністю того, що конкретне вихідне значення належить до визначеного класу.

Логістична регресія – це статистичний підхід, що використовується для бінарної класифікації, тобто для прогнозування ймовірності віднесення об'єкта до одного з двох класів (наприклад, “платоспроможний клієнт” проти “неплатоспроможного клієнта”). Модель логістичної регресії визначає взаємозв'язок між групою незалежних змінних (факторів) та бінарною залежною змінною (класом) шляхом створення спеціалізованої сигмоїдної функції. На відміну від лінійної регресії, що може повертати будь-які числові значення, логістична модель обмежує результати в межах від 0 до 1, які тлумачаться як ймовірність належності до позитивного класу. Функція ймовірності успішного результату визначається за допомогою сигмоїди (логістичної функції):

$$\rho(x) = \frac{1}{1+\exp\left(-(\beta_0 + \beta_1 x_1 + \ldots_- + \beta_n x_n)\right)},$$

де $\rho(x)$ - оцінювана ймовірність, що об'єкт з ознаками $x = (x_1, \ldots, x_n)$ належить до класу "1";

β_0 – вільний член (зсув), β_1 – коефіцієнти при відповідних факторах x_i.

Функцію, яку описує це рівняння, зветься логістичною, а її графік - сигмоїдою, через наявність виразної S-подібної форми.

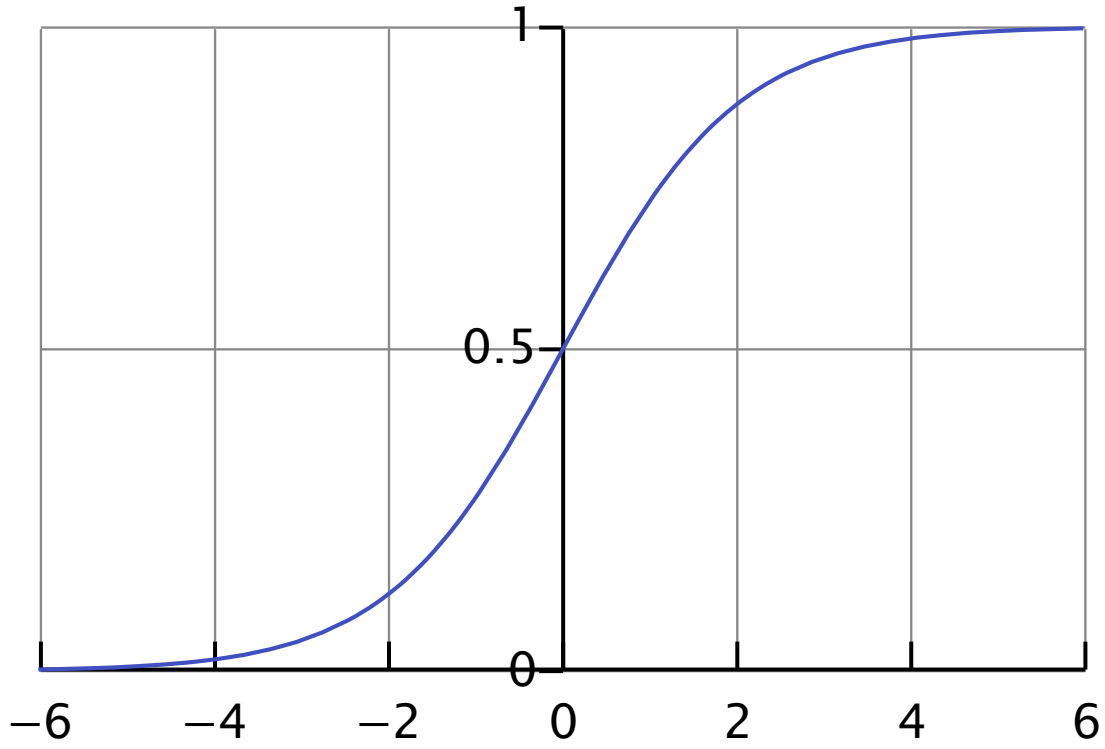

Рисунок 1.2 – Сигмоїда, логістична функція

Кожен коефіцієнт β_i характеризує вплив відповідної факторної змінної на логарифм відношення шансів (log-odds) події. Зокрема, якщо розглянути відношення шансів настання події (де шанс визначається як відношення ймовірності до 1 - ймовірності), то логістична регресія лінійно моделює логарифм шансів як:

$$\ln\frac{\rho}{1-\rho} = \beta_0 + \beta_1 x_1 + \ldots_- + \beta_n x_n.$$

Така параметризація дозволяє інтерпретувати експоненту коефіцієнта як відношення шансів (Odds Ratio) для події при зміні x_i на одиницю. Якщо коефіцієнт більший за одиницю, то зі збільшенням x_i шанс настання події (дефолту) зростає; якщо нижче, то відповідний фактор знижує ризик.

Завдяки цьому логістична регресія є дуже інтерпретованою моделлю – можна кількісно оцінити, як саме кожний фактор впливає на ймовірність дефолту клієнта, що важливо для фінансових аналітиків і регуляторів. Ця властивість (інтерпретованість через коефіцієнти та відношення шансів) є суттєвою перевагою логістичної регресії при використанні в кредитному скорингу.

У процесі виконання логіт-перетворення обох компонентів рівняння логістичної регресії формується стандартна модель лінійної регресії. Отримані результати будуть більшими 1 або меншими 0. Таким чином, обмеження на змінну y не враховуються за цих умов. По-іншому формулюється задача регресії для вирішення виниклої проблеми: буде прогнозовано неперервну змінну з значеннями на інтервалі [0; 1] для будь-яких значень незалежних змінних.

Використання логіт-перетворення в рівнянні логістичної регресії викликає певні труднощі. Отже, при створенні моделі лінійної регресії коефіцієнти регресії можна отримати за допомогою методу найменших квадратів. Необхідні нормальність та незалежність помилок.

Для визначення коефіцієнтів логістичної регресії таких варіантів немає. Отже, коефіцієнти логістичної регресії визначаються за допомогою алгоритму максимального правдоподібності, що дає змогу знайти деякі значення коефіцієнтів, при яких ймовірність виникнення є максимальною.

Розглянемо метод правдоподібності l (β| x) (likelihood function), що задає імовірність отримання значень параметрів $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ при фіксованому значенні x. Завдання полягає в отриманні таких значень для набору параметрів $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$, які підвищують метод правдоподібності: створюються оцінки максимальної правдоподібності (maximum likelihood estimates), для яких

значення параметрів найкраще підходять для набору даних.

Імовірність того, що вихідна змінна y буде дорівнювати 1 при заданому значенні x (ймовірність дефолту) визначається як $\rho(x) = P(y = 1 \mid x)$, тоді як вірогідність $y = 0$ при даному x (ймовірність відсутності дефолту) дорівнює $1 - \rho(x) = P(y = 0 \mid x)$.

Отже, враховуючи що $y_i = 0$ або $y_i = 1$, внесок і-го спостереження можна подати у вигляді $[\rho(x_i)]^{y_i} \times [1 - \rho(x_i)]^{1 - y_i}$.

Допущення, що спостереження є автономним, дає змогу представити функцію правдоподібності у вигляді добутку двох окремих складових - $l(\beta| x) = \prod_{i=1}^{n} [\rho(x_i)]^{y_i} \times [1 - \rho(x_i)]^{1 - y_i}$.

Відповідно до методу максимальної правдоподібності, оцінками невідомих параметрів є значення, які надають змогу максимізувати функцію $l(\beta| x)$.

У розрахунковому аспекті більш зручною є логарифмічна функція правдоподібності $L(\beta| x) = \ln [l(\beta| x)]$.

Оцінки максимальної правдоподібності отримують шляхом диференціювання $L(\beta| x)$ за кожним параметром і подальшого рівняння їх похідних до нуля.

Визначити коефіцієнти логістичної регресії можна з використанням градієнтних методів: методу зчеплених градієнтів, методів варіантної метрики та інших.

Ключовим є не лише математично змоделювати систему, але й вірно її інтерпретувати в контексті аналізу, що дасть змогу отримати всю потрібну інформацію про об'єкти та процеси, що вивчаються. В парній лінійній регресії $y = \beta_1 x + \beta_0$ коефіцієнт β_1 демонструє на скільки в середньому зміниться значення залежної змінної y, якщо незалежна змінна x зміниться на одну одиницю.

Інтерпретація у логістичної регресії є подібною, але стосується логістичної функції. Таким чином коефіцієнт β_1 може бути представлений як зміна значення

логістичної функції при зміні вхідної змінної однією одиницю свого виміру.

Це можна виразити як $\beta_1 = g(x + 1) - g(x)$.

Завдання класифікації та регресії відрізняються за типом вихідної змінної. Якщо вихідна змінна є безперервною, то це задача регресії, а якщо дискретною (мітка класу), то задача класифікації. Як зазначалося раніше, логістична регресія підтримує роботу з дихотомічною вихідною змінною, що дозволяє застосовувати цей метод для вирішення завдань бінарної класифікації. У бінарній класифікації кожне спостереження або предмет повинні бути віднесені до одного з двох класів (наприклад, A і B). Отже, з кожним результатом асоційована подія: об'єкт входить до класу A і об'єкт входить до класу B. Результатом стане оцінка ймовірності відповідного результату. Якщо під час аналізу буде з'ясовано, що ймовірність $P(A)$ належності об'єкта з певним набором значень ознак (вхідних змінних) до класу A перевищує ймовірність $P(B)$ його належності до класу B, то його буде віднесено до об'єктів класу A. Вочевидь, що за умови несумісності події, буде $P(B) = 1 - P(a)$.

Можна встановити поріг ймовірності, за умови перевищення якого ймовірність, пов'язана з конкретним класом, «перевищує» і об'єкт належить цьому класу. У найелементарнішому варіанті це може бути поріг однакової ймовірності, а саме 0,5. Якщо ймовірність $P(A) > 0{,}5$, то об'єкт відноситься до класу A, якщо ймовірність $P(A) < 0{,}5$, то об'єкт відноситься до класу B.

Іноді поріг встановлюється більш складно, наприклад, базуючись на надійності ухваленого рішення.

Отже, рішення про те, до якого класу належить об'єкт, може бути ухвалене лише в тому випадку, якщо ймовірність цієї події, визначена за допомогою логістичної регресії, перевищить 0,6.

1.2 Метрики оцінки якості моделі

Для кількісної оцінки якості моделі класифікації, такої як логістична регресія, застосовують низку метрик. Базовим інструментом є матриця

невідповідностей (confusion matrix), яка відображає розподіл правильних і помилкових класифікацій моделі [5]. Вона має вигляд таблиці 2×2 для бінарної класифікації, де по одному виміру йдуть фактичні класи (реальність), а по іншому – передбачені моделлю класи. Елементи матриці: TP (True Positives, істинно позитивні) – кількість випадків, де позичальник дефолтував і модель правильно це передбачила; TN (True Negatives, істинно негативні) – випадки, де клієнт справно погасив борг і модель вірно не виявила ризику; FP (False Positives, хибно позитивні) – кількість “помилкових тривог”, коли модель спрогнозувала дефолт, хоча клієнт насправді повернув кредит; FN (False Negatives, хибно негативні) – пропущені дефолти, коли модель не виявила ризик, а клієнт збанкрутував.

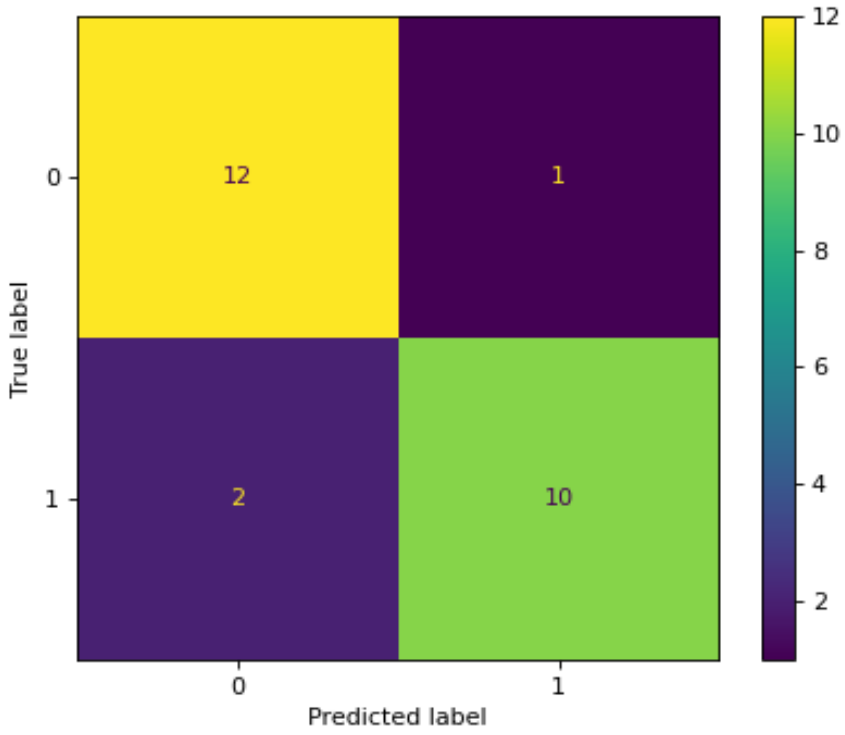

Рисунок 1.3 – Приклад матриці невідповідностей (confusion matrix)

На основі цих показників обчислюються такі метрики:

- Точність (Accuracy) – частка правильно класифікованих серед усіх,

виражається наступною формулою: $\frac{\text{True Positives+ True Negatives}}{\text{True Positives+True Negatives + False Positives+ False Negatives}}$. Це загальна частка вірних рішень моделі. Втім, при суттєвому дисбалансі класів ця метрика може вводити в оману (наприклад, якщо дефолти дуже рідкісні, модель може показувати високу точність, завжди передбачаючи “не дефолт”) ;

- Повнота (або чутливість, Recall, Sensitivity, також True Positive Rate): $\frac{\text{True Positives}}{\text{True Positives+ False Negatives}}$. Висока повнота означає, що модель виявляє більшість проблемних позичальників (мало пропускає дефолтів), але може бути за рахунок більшої кількості помилкових тривог;
- Специфічність (Specificity) – частка надійних клієнтів, яких модель правильно визначила як таких: $\frac{\text{True Negatives}}{\text{True Negatives + False Positives}}$. Висока специфічність означає мало хибних тривог.
- Прецизійність (Precision) - частка реальних дефолтів серед тих, кому модель спрогнозувала дефолт: $\frac{\text{True Positives}}{\text{True Positives+False Positives}}$. Ця метрика показує, наскільки можна довіряти позитивному прогнозу моделі.
- F1-вимір – середнє прецизійності та повноти: $2 * \frac{\text{Preicision*Recall}}{\text{Preicision + Recall}}$. Використовується як інтегральна метрика, особливо коли важливо знайти баланс між пропущеними дефолтами і хибними тривогами.

Окрім перелічених метрик, у кредитному скорингу широко застосовуються графічні методи оцінки, зокрема ROC-крива (Receiver Operating Characteristic) [6]. ROC-крива будується на основі варіювання порогу класифікації і відображає співвідношення між істинно позитивним

рівнем (TPR) та хибно позитивним рівнем (FPR) для різних значень порогу.

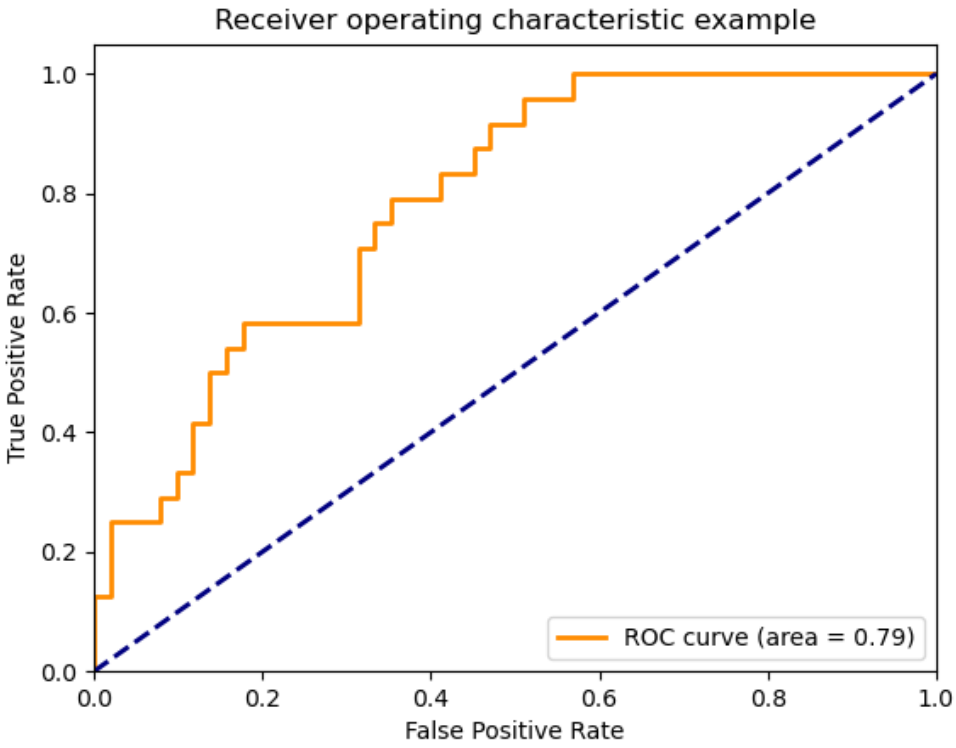

Рисунок 1.4 – Приклад ROC-кривої

На осі ординат відкладається TPR (чутливість), на осі абсцис – FPR (1 - специфічність). Кожна точка ROC-кривої відповідає певному порогу прийняття рішення моделі. ROC-крива дозволяє візуально оцінити здатність моделі розрізняти класи: що ближче вона проходить до верхнього лівого кута графіка (точка TPR=1, FPR=0), то краща модель. Для “ідеального” класифікатора ROC-крива пройде через кут (0,1), тоді як для випадкового моделі це приблизно діагональ від (0,0) до (1,1).

Кількісним показником, що узагальнює ROC-криву, є площа під ROC-кривою (AUC). Значення AUC змінюється від 0 до 1; практично цікавий діапазон – від 0.5 (модель не краща випадкового вгадування) до 1.0 (ідеальна модель). Чим більше AUC, тим вища загальна відокремлююча здатність моделі на всіх можливих порогах. В банківській сфері вважається,

що модель із AUC < 0.7 – слабка, AUC ~0.75 і вище – прийнятна для практичного використання (фактично пороговий рівень якості).

Наприклад, галузевим стандартом є досягнення AUC не менше 0.75 перед тим, як модель кредитного скорингу буде прийнята до використання. AUC зручно інтерпретувати: це ймовірність того, що модель присвоїть випадково вибраному дефолтному клієнту вищий скоринговий бал (ймовірність дефолту), ніж випадково вибраному надійному клієнту.

У подальших розділах ці метрики будуть застосовані для оцінки якості побудованої моделі логістичної регресії на тестових даних, зокрема буде побудовано ROC-криву і розраховано значення AUC. Для нашої моделі очікуються високі показники – наприклад, AUC на рівні ~0.9, що відповідатиме високій прогностичній точності (як буде показано, фактично отримано AUC ≈0.91, що свідчить про високоефективну модель).

1.3 Порівняння моделі логістичної регресії із іншими моделями машинного навчання.

Як зазначалося, логістична регресія – найпоширеніша математична модель для задач оцінки кредитоспроможності у банківській практиці. Однак, варто розглянути, які існують альтернативи та в чому їх переваги чи недоліки порівняно з логістичною моделлю.

У даній роботі обрано RandomForestClassifier, GradientBoostingClassifier та Support Vector Classifier (SVC) як альтернативні моделі [7, 8, 9]. Вибір зумовлений тим, що ці алгоритми репрезентують різні підходи до класифікації (ансамблеве бегінг у випадку Random Forest, ансамблеве бустинг у випадку Gradient Boosting, та метод опорних векторів для SVC) і відомі високою точністю прогнозування у задачах класифікації, зокрема у сфері фінансового ризику.

Порівняння логістичної регресії з цими моделями дозволяє оцінити, наскільки більш складні нелінійні алгоритми покращують якість прогнозування

фінансової поведінки клієнтів, а також які компроміси між точністю та інтерпретованістю виникають.

RandomForestClassifier (RF) – це ансамблевий алгоритм на основі рішень дерев, що реалізує метод випадкового лісу. Математична сутність Random Forest полягає в побудові багатьох незалежних дерев рішень на випадкових підмножинах даних та ознак, після чого для класифікації використовується голосування більшості серед прогнозів цих дерев.

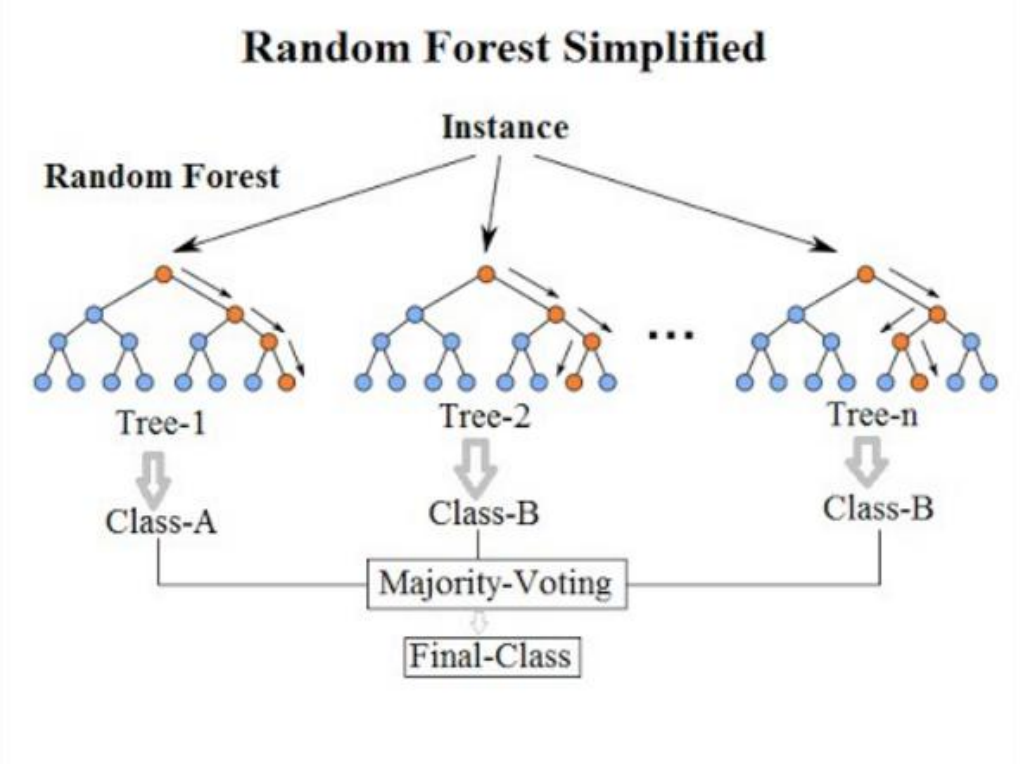

Рисунок 1.4 – Концептуальна візуалізація Random Forest

Такий підхід «мудрості множини» забезпечує високу стійкість моделі: випадковий ліс здатен виявляти складні нелінійні взаємозв'язки між ознаками і менш схильний до перевищування моделі (overfitting), ніж окреме дерево рішення. Переваги RandomForest у контексті фінансового прогнозування включають високу точність і здатність обробляти великі масиви різнорідних даних (алгоритм невимогливий до нормалізації ознак, стійкий до викидів тощо). До того ж, випадковий ліс може надавати показники важливості ознак, що

частково сприяє інтерпретації моделі.

Недоліки натомість пов’язані з меншою прозорістю: логіка ухвалення рішень розподілена між багатьма деревами, тому пояснити, чому модель класифікувала того чи іншого клієнта як ризикового, складніше, ніж у випадку логістичної регресії. Також модель Random Forest може бути обчислювально затратною (велика кількість дерев уповільнює роботу) і потребує достатнього обсягу пам’яті, що варто враховувати при реалізації у веб-застосунку.

GradientBoostingClassifier (GB) – представник алгоритмів градієнтного бустингу, який будує ансамбль моделей послідовно. Кожне наступне дерево рішень навчається на помилках попереднього, мінімізуючи залишкову помилку за допомогою градієнтного спуску (звичайно використовується функція логістичної втрати для бінарної класифікації). Таким чином, Gradient Boosting поступово підсилює загальний класифікатор, додаючи слабкі моделі, що виправляють похибки попередніх. Математично підхід реалізує адитивну модель: фінальний прогноз – це сума внесків багатьох дрібних дерев [10]. Переваги GradientBoosting особливо проявляються у високій точності та здатності вловлювати тонкі нелінійні закономірності в даних. На практиці градієнтний бустинг (зокрема, його популярна реалізація XGBoost) неодноразово демонстрував кращу якість прогнозування порівняно з логістичною регресією та класичними методами на задачах кредитного ризику. Крім того, модель бустингу гнучко настроюється через гіперпараметри (кількість дерев, глибина, швидкість навчання тощо), що дозволяє досягти оптимального балансу між чутливістю і специфічністю. Недоліки цього підходу пов’язані з обчислювальною складністю: послідовне навчання багатьох дерев є ресурсоємним і потребує більше часу. Як і випадковий ліс, бустинг є «чорним ящиком» з точки зору пояснення результатів – прямої інтерпретації внеску кожної ознаки не існує.

Support Vector Classifier (SVC) – це реалізація методу опорних векторів для класифікації. SVC шукає такий гіперплощинний роздільник у просторі ознак,

який максимально віддаляє представників двох класів один від одного. Ключова ідея методу – знайти оптимальну гіперплощину з максимальною відстанню до найближчих точок навчальної вибірки обох класів; ці найближчі точки називаються опорними векторами. Максимізація відстані теоретично забезпечує найкраще узагальнення на нових даних – модель намагається побудувати якнайширший «коридор» між класами, щоб зменшити помилки класифікації [11]. Для випадків, коли дані не є лінійно роздільними, SVM використовує ядрові функції (RBF, поліноміальні та інші) для відображення ознак у вищий вимір, де класи можна розділити лінійно. Перевагами SVC є висока гнучкість у моделюванні меж класифікації: за допомогою належного ядра цей метод здатен моделювати дуже складні нелінійні рішення. Він також добре працює з високорозмірними даними (коли кількість ознак велика порівняно з кількістю спостережень) і може бути ефективним при наявності невеликої кількості навчальних прикладів, оскільки рішення визначається лише опорними векторами. Недоліки SVC у фінансовому прогнозуванні пов’язані, по-перше, з відсутністю прозорості: модель не дає явних коефіцієнтів, які можна інтерпретувати економічно, а отже важко пояснити рішення (наприклад, чому клієнта віднесено до групи ризику) – на відміну від логістичної регресії, де вагові коефіцієнти безпосередньо вказують на вплив ознак. По-друге, SVC потребує ретельного підбору параметрів (тип ядра, параметр регуляризації C, параметри ядра тощо) для досягнення оптимальної точності; неправильний вибір параметрів може призвести до погіршення результатів. По-третє, алгоритм масштабується гірше на великих вибірках: час навчання зростає набагато швидше, ніж у випадку лінійних моделей чи ансамблів дерев, що може стати проблемою при великому обсязі клієнтських даних. Нарешті, SVC не надає прямо ймовірностей класів (потрібне додаткове калібрування), що дещо ускладнює інтеграцію його результатів у прикладні рішення типу «скорингового балу».

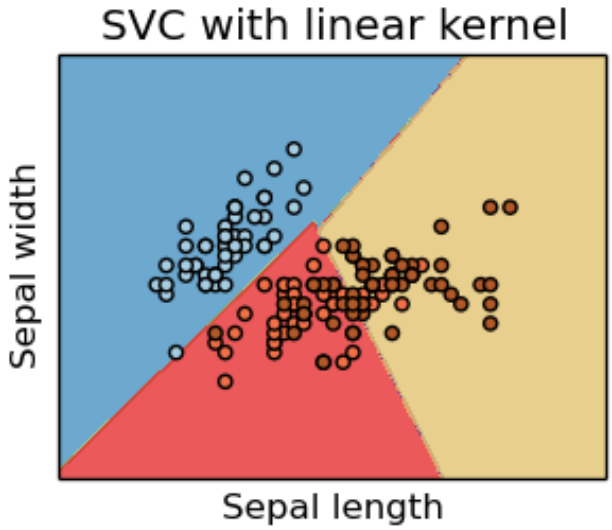

Рисунок 1.5 – Концептуальна візуалізаця SVC

Обрані для порівняння моделі – RandomForest, GradientBoosting та SVC – представляють сучасні підходи, які потенційно можуть перевершити логістичну регресію за якістю прогнозування. Кожна з них має математично більш потужний апарат для виявлення складних патернів: ансамблі дерев враховують нелінійні взаємодії ознак, а метод опорних векторів шукає оптимальні межі між класами. У контексті завдання прогнозування фінансової поведінки (кредитного ризику) такі моделі часто демонструють кращі метрики класифікації, що підтверджено літературою.

Водночас, аналіз їх переваг і недоліків показує, що підвищення точності досягається ціною зниження інтерпретованості та ускладнення реалізації. Саме тому доцільно порівнювати ці алгоритми з логістичною регресією: це дає змогу оцінити, чи є виграш у ефективності прогнозування та чи виправдовує він збільшення складності моделі. Такий порівняльний аналіз надалі допоможе обґрунтувати вибір моделі для впровадження у веб-застосунку «Банківський асистент» з урахуванням як точності, так і пояснюваності результатів.

2. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

2.1 Аналіз предметної області

Фінансова поведінка клієнта у контексті банківської діяльності – це сукупність характеристик та дій позичальника, що визначають його здатність та готовність своєчасно виконувати фінансові зобов'язання (насамперед погашати кредити). Оцінка кредитоспроможності клієнтів та управління кредитними ризиками є невід'ємною частиною діяльності будь-якого банку або фінансової організації. Традиційно в банках оцінювання проводилося кредитними експертами вручну на основі аналізу довідок про доходи, кредитної історії, застави, соціально-демографічних даних клієнта тощо. При цьому використовувалися як суб'єктивні підходи (рішення експерта на основі його досвіду), так і формалізовані підходи у вигляді скорингових карт (бальної системи). Сучасна тенденція – перехід до автоматизованих систем оцінки, які будуються на основі статистичних моделей або алгоритмів машинного навчання, що навчились на великих масивах історичних даних. Такий підхід, званий скоринговим моделюванням, дає змогу банкам підвищити об'єктивність та швидкість прийняття рішень.

Суть скорингової моделі - за набором показників про клієнта (вік, дохід, сімейний стан, кількість відкритих кредитів, наявність прострочень у минулому, тощо) модель прогнозує певний скоринговий бал або безпосередньо ймовірність дефолту. На основі цього прогнозу банк приймає рішення: якщо ймовірність дефолту низька – кредит схвалюється, якщо висока – відхиляється або, можливо, пропонується менший ліміт/вища ставка. Результати індивідуального оцінювання також впливають на загальну оцінку ризиків кредитного портфеля банку.

В Україні та світі накопичено значний досвід використання скорингових систем. Існують спеціалізовані бюро кредитних історій та агентства, які агрегують дані про позичальників і надають скорингові оцінки. Наприклад, Українське бюро кредитних історій (УБКІ), створене за участі провідних

банків, формує кредитні рейтинги позичальників. Приклад: УБКІ для оцінки кредитного ризику фізичної особи враховує такі показники, як наявність та тривалість прострочень в минулому, кількість відкритих та закритих кредитів і їх суми, частота звернень за кредитами тощо. Результат часто подається у вигляді числового балу або класу ризику. Інший приклад – міжнародне бюро Experian, що діє в багатьох країнах: воно надає скоринговий бал від 0 до 999, де вищий бал означає кращу кредитну історію; зазначено, що для розрахунку використовуються моделі на основі дерев рішень із близько 30 вхідними змінними. Крім бюро, великі банки розробляють власні внутрішні скорингові системи, адаптовані під їхній портфель і політики ризик-менеджменту.

З точки зору доступності реалізації, варто відзначити, що багато відкритих даних чи API для готових скорингових систем у відкритому доступі немає (це комерційна таємниця банків). Однак існують демо-версії або спрощені сервіси для оцінки кредитного рейтингу онлайн. Наприклад, сервіс myFICO estimator – інструмент від компанії FICO (США), що дозволяє користувачу відповісти на ~10 питань про свою фінансову ситуацію і отримати орієнтовний скоринговий бал. Питання стосуються кількості кредитних карток, обсягу заборгованості, історії погашення боргів тощо. На основі цих відповідей система видає приблизну оцінку кредитного рейтингу та звіт з поясненнями. Це приклад того, як на основі нескладної моделі (ймовірно, зважена балова система або спрощена логістична модель) можна швидко дати користувачу уявлення про його кредитоспроможність.

Отже, аналізуючи предметну область, можна зробити висновок, що автоматизація процесу оцінювання кредитних ризиків є зрілою, але закритою сферою: точні моделі і їх параметри, які використовуються банками, публічно не розголошуються. Тим не менш, відомо про успішний досвід використання логістичної регресії, дерева рішень та ансамблевих алгоритмів у цій галузі.

Ця робота спирається на загальні принципи, описані в літературі та практиці: ключові фінансові показники клієнта, побудова на їх основі

прогнозної моделі та подальше впровадження її у програмний додаток для автоматизації.

2.2 Огляд існуючих рішень та прототипів системи

Незважаючи на обмежений відкритий доступ до повнофункціональних скорингових систем банків, деякі аналоги та прототипи можна проаналізувати для отримання ідей щодо функціоналу і підходів.

Як вже згадано, myFICO estimator – це приклад простого веб-додатку для оцінки кредитного рейтингу споживача. Він запитує у користувача базову інформацію (число кредитів, загальний борг, наявність прострочок тощо) і на виході надає звіт із орієнтовним скоринг-балом та поясненнями, які фактори найбільше вплинули на оцінку. Такий сервіс, хоч і спрощений, демонструє важливість зрозумілості для користувача: окрім самого числового результату, надається інтерпретація – наприклад, “Ваш бал знижений через високий обсяг непогашених боргів”. Це підтверджує, що при розробці власного “Банківського асистента” слід передбачити генерацію пояснень для менеджера, на основі яких той зможе обґрунтувати рішення клієнтові.

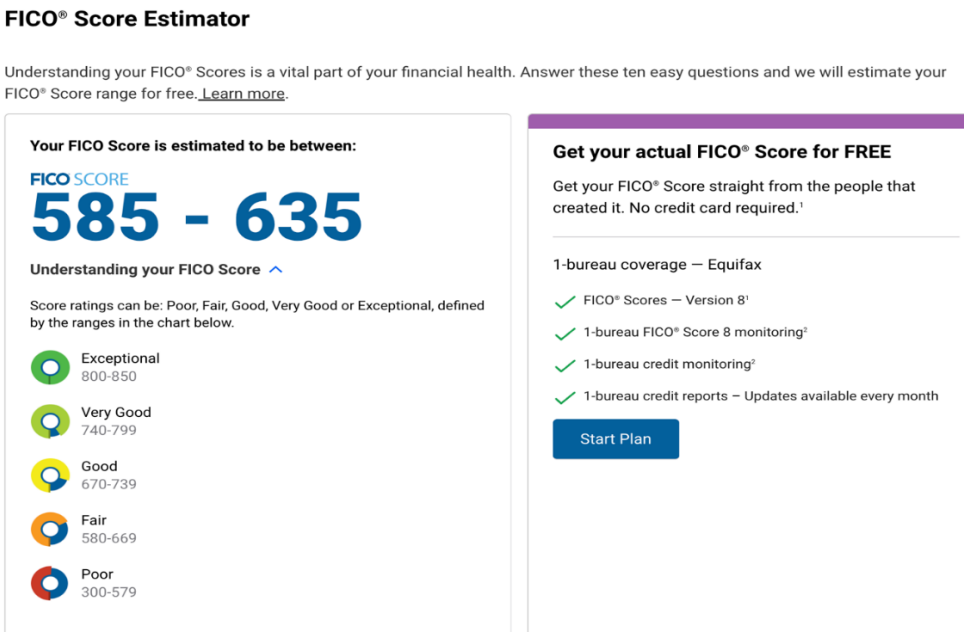

Рисунок 2.1 – Приклад звіту кредитного скорингу від myFICO estimator

Ще один приклад системи – УБКІ (Українське бюро кредитних історій). Хоч прямого доступу до їх алгоритмів немає, з описів випливає, що видається звіт, який включає інтегральний кредитний бал та перелік факторів: прострочки, кредити, запити кредитної історії тощо. Тобто, звіт містить як числову оцінку, так і структуровану інформацію про фінансову активність позичальника. Для нашого застосунку це означає, що просто показати “ймовірність дефолту = 20%” недостатньо – бажано також вивести перелік використаних клієнтських показників та їх значення, можливо вказати які з них найбільше впливають (наприклад: “Низький дохід – високий ризик”, “Короткий стаж роботи – помірний ризик” і т.д.). Такі пояснення можна формувати або за допомогою аналізу коефіцієнтів моделі, або заздалегідь закласти деякі правила інтерпретації.

Ще один приклад – згадана система Experian (міжнародний кредитний офіс). Вона показує, що скорингові оцінки можуть бути представлені у різних шкалах (не обов’язково 0-1 або 0-100%). У Experian шкала 0-999, і чим вищий бал, тим надійніший клієнт. У нашій роботі ми використовуватимемо ймовірність дефолту у відсотках як основний результат, але при бажанні банк може конвертувати цю ймовірність у власну рейтингову шкалу (наприклад, ABC клас або числовий бал). Принцип формування звіту Experian цікавий тим, що вони застосовують модель дерев, та ще й, ймовірно, з великою кількістю факторів – отже, інтерпретація може бути складнішою. Але вони вирішують це тим, що надають підсумковий бал і рекомендації (наприклад: “Ваш бал 700, ризик низький, ви маєте право на стандартні умови кредитування”).

Загалом, огляд прототипів і рішень дозволяє виокремити ключові вимоги до функціоналу нашої системи:

- збір необхідних вхідних даних про клієнта (або введення їх користувачем чи менеджером);
- розрахунок скорингового показника (ймовірності дефолту) за допомогою моделі;

- збереження результату в базі даних для подальшого аналізу або використання (наприклад, історія заявок клієнта);
- різні ролі користувачів (співробітник банку, який переглядає усі результати, сам клієнт, який може отримати свій результат, тощо), що потребує механізмів автентифікації і розмежування доступу;
- інтерфейс повинен бути зручним та захищеним.

2.3 Постановка задачі розробки системи

На основі проведеного аналізу можна сформулювати конкретну постановку задачі для магістерської роботи. Необхідно спроєктувати та реалізувати інформаційну систему – веб-застосунок «Банківський асистент», що автоматизує оцінку кредитоспроможності клієнтів за допомогою моделі логістичної регресії. Система орієнтована на використання у банківській установі співробітниками відділу кредитного аналізу, а також може мати зовнішній інтерфейс для клієнтів.

Основні функціональні можливості системи:

- реєстрація та авторизація користувачів. Система повинна підтримувати створення облікових записів користувачів різних ролей і безпечний вхід (login) з використанням імені користувача та паролю. Паролі мають зберігатися у базі даних у хешованому вигляді. Має бути як мінімум дві ролі: Адміністратор (співробітник банку з розширеними правами) та Клієнт (звичайний користувач, що оцінює свій рейтинг);
- створення нового скорингового звіту. Користувач-менеджер або клієнт заповнює електронну форму з даними про позичальника: необхідні параметри (паспортні дані, вік, дохід, сімейний стан, працевлаштування, сума та строк бажаного кредиту, тощо). Після відправлення форми серверна частина застосунку обробляє дані, застосовує модель логістичної регресії та генерує звіт про оцінку

кредитоспроможності. Звіт включає: імовірність дефолту (у відсотках), рекомендацію (наприклад, “Рекомендовано відхилити заявку” або “Можна видавати кредит”). Звіту присвоюється унікальний ідентифікатор та зберігається у базі даних;

- перегляд існуючих звітів. Адміністратор має доступ до списку всіх згенерованих звітів (наприклад, у вигляді таблиці: № заявки, ПІБ клієнта, дата, результат оцінки, рішення). Він може відкрити деталізацію кожного звіту. Звичайний клієнт (аутентифікований) може переглядати тільки власні звіти (якщо він подавав заявку через систему, або якщо йому надано доступ за ідентифікатором).

Критичною вимогою є також точність та надійність роботи прогнозної моделі. В рамках задачі потрібно використати наявний набір даних (наприклад, з відкритих джерел – умовний датасет позичальників із ознаками та міткою дефолту), на якому провести навчання моделі логістичної регресії і її тестування. Модель повинна продемонструвати прийнятні показники (Accuracy, AUC ≥ 0.75). Бажано досягти показників, що наближаються до промислових зразків (класичні скорингові моделі на основі логістичної регресії можуть мати AUC ~0.80–0.85 на добре підготовлених даних).

3. ПОБУДОВА ТА ОЦІНЮВАННЯ МОДЕЛІ

3.1 Постановка задачі розробки системи

У цьому розділі аналізується створення моделі логістичної регресії (за допомогою бібліотеки scikit-learn), візуалізація інформації (з використанням бібліотек pandas та matplotlib) та обробка даних (бібліотека pandas).

Спочатку необхідно провести аналіз даних з набору, що буде застосовано для навчання та перевірки моделі логістичної регресії. На рисунку 3.1 представлені стовпці з датасету, їхні типи даних та наявність значень NULL. цьому підрозділі розглядається побудова моделі логістичної регресії (за допомогою бібліотеки scikit-learn), візуалізація даних (за допомогою бібліотек pandas та matplotlib) і робота з даними (бібліотека pandas) [12, 13, 14].

```
[5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 700 entries, 0 to 699
Data columns (total 9 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       700 non-null    int64
 1   ed        700 non-null    int64
 2   employ    700 non-null    int64
 3   address   700 non-null    int64
 4   income    700 non-null    int64
 5   debtinc   700 non-null    float64
 6   creddebt  700 non-null    float64
 7   othdebt   700 non-null    float64
 8   default   700 non-null    int64
dtypes: float64(3), int64(6)
memory usage: 49.3 KB
```

Рисунок 3.1 – Інформація про атрибуту набору даних та їхні типи

Для більш поглибленого розуміння даних були реалізовані наступні дії: аналіз агрегованих показників кожної з колонки (кількість записів, середнє значення, значення стандартної девіації, мінімальне та максимальне значення) наведено на рисунку 3.2; візуалізація залежності наявності дефолту (default) від щомісячного доходу (income) та боргом (creddebt) наведено на рисунку 3.3; візуалізація залежності наявності дефолту (default) від щомісячного доходу (income) та іншими кредитними зобов'язаннями (othdebt) наведено на рисунку 3.4; візуалізація кількості позичальників, що дійшли до дефолту, та навпаки наведено на рисунку 3.5; візуалізація матриці кореляцій (матриця, що показує коефіцієнти кореляції між двома змінними) наведено на рисунку 3.6.

```
memory usage: 49.3 KB
```

```
[6]: df.describe()
```

[6]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
count	700.000000	700.000000	700.000000	700.000000	700.000000	700.000000	700.000000	700.000000	700.000000
mean	34.860000	1.722857	8.388571	8.278571	45.601429	10.260571	1.553553	3.058209	0.261429
std	7.997342	0.928206	6.658039	6.824877	36.814226	6.827234	2.117197	3.287555	0.439727
min	20.000000	1.000000	0.000000	0.000000	14.000000	0.400000	0.011696	0.045584	0.000000
25%	29.000000	1.000000	3.000000	3.000000	24.000000	5.000000	0.369059	1.044178	0.000000
50%	34.000000	1.000000	7.000000	7.000000	34.000000	8.600000	0.854869	1.987567	0.000000
75%	40.000000	2.000000	12.000000	12.000000	55.000000	14.125000	1.901955	3.923065	1.000000
max	56.000000	5.000000	31.000000	34.000000	446.000000	41.300000	20.561310	27.033600	1.000000

Рисунок 3.2 - Агреговані показники кожної з колонки (кількість записів,середнє значення, значення стандартної девіації, мінімальне та максимальне значення)

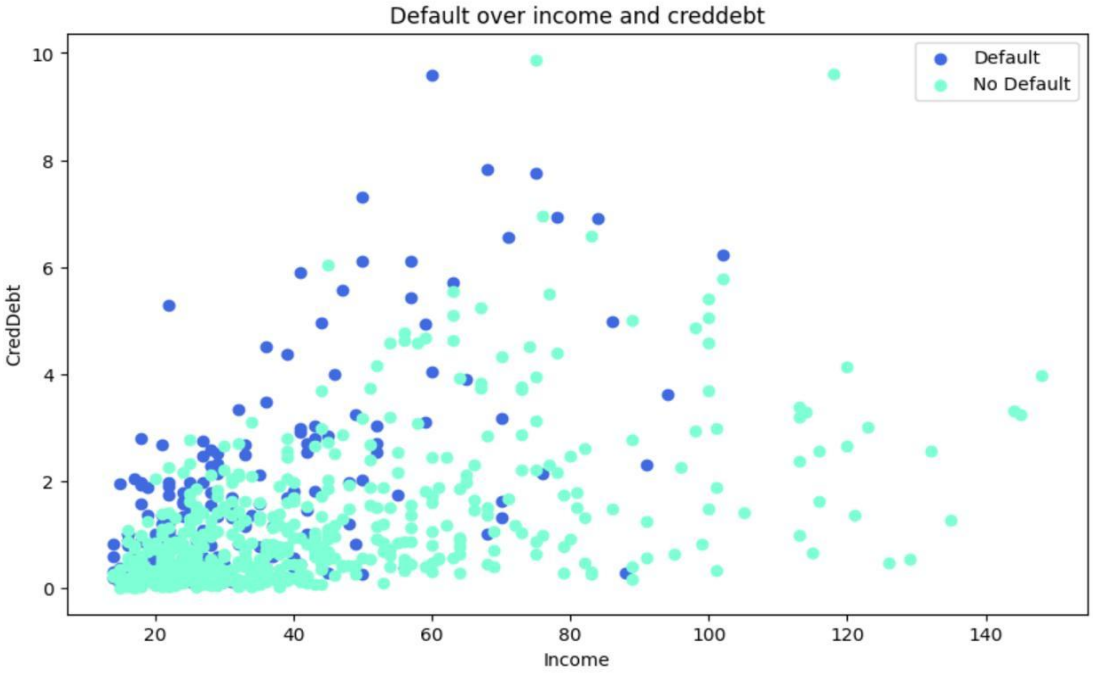

Рисунок 3.3 - Залежність наявності дефолту (default) від щомісячного доходу (income) та боргом (creddebt)

З рисунку 3.4 можна дійти до висновку, що наявна позитивна кореляція між вхідною змінною боргу та вихідною ймовірністю дефолту. З цього - чим більший борг позичальника, тим вища ймовірність дефолту. А вплив боргу на ймовірність дефолту більш значний, ніж вплив щомісячного доходу.

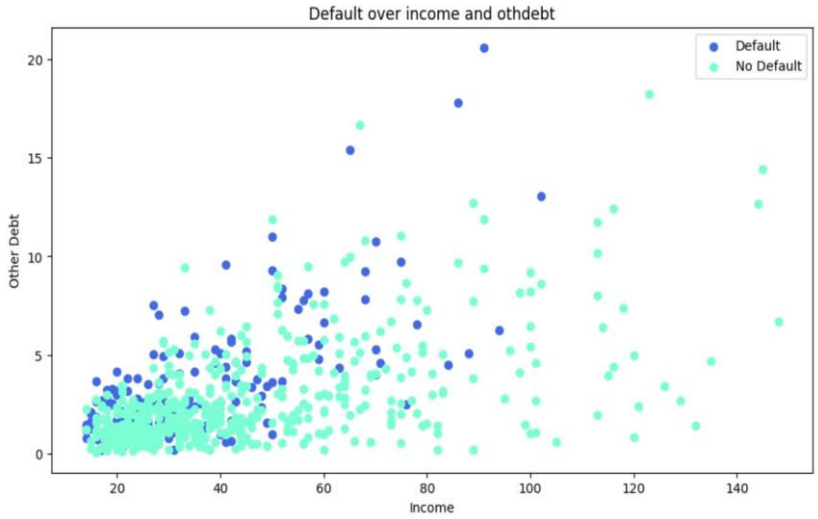

Рисунок 3.4 - Залежність наявності дефолту (default) від щомісячного доходу (income) та іншими боргами (othdebt)

З рисунку 3.5 – також існує позитивна кореляція між показником creddebt та ймовірністю дефолту

```
[8]: df.default.value_counts().plot(
         kind="bar",
         color=["royalblue", "aquamarine"],
     )
     plt.xticks(rotation=0);
```

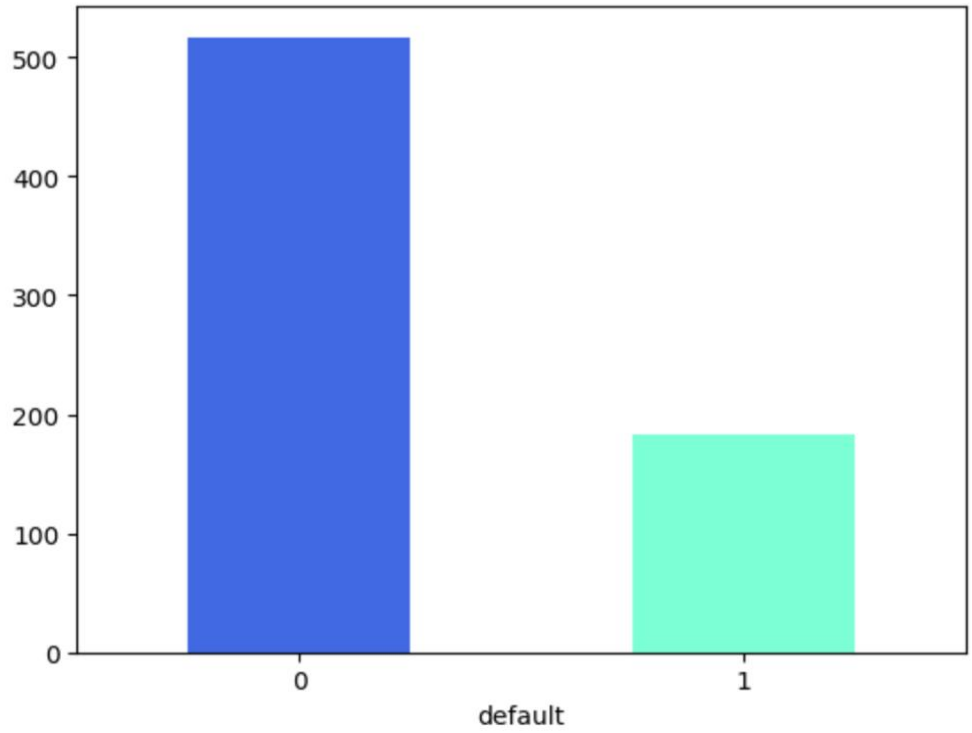

Рисунок 3.5 - Кількість позичальників, що дійшли до дефолту, та які не дійшли

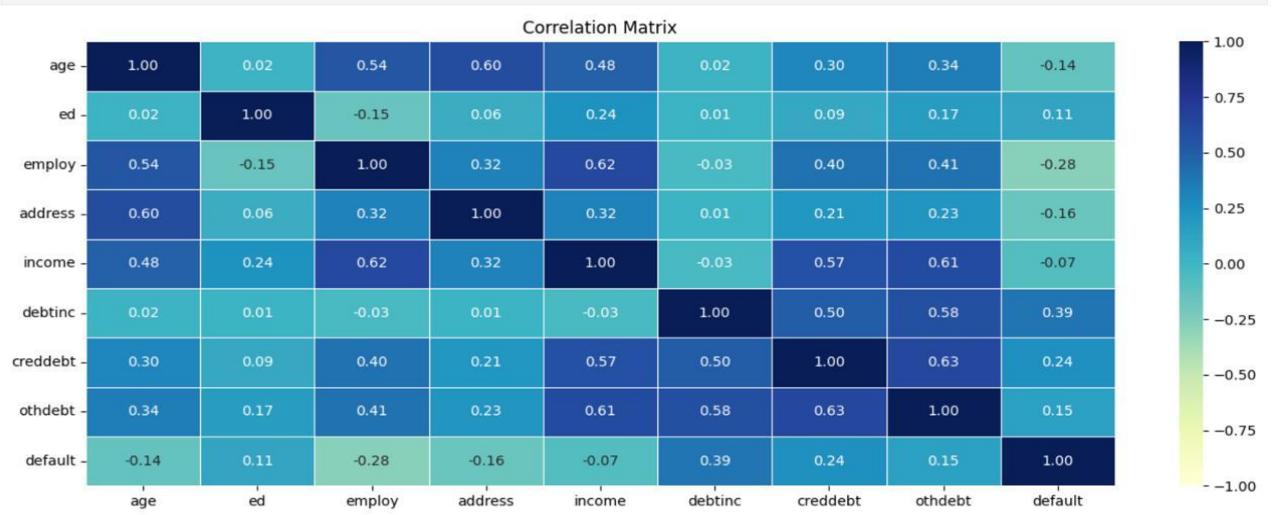

Рисунок 3.6 – Матриця кореляцій

З кореляційної матриці можна зробити такі висновки [15]:

- Змінні "age" та "default" мають незначний негативний зв'язок. Це може вказувати на те, що з віком може зменшуватись ймовірність дефолту позичальника (як правило).
- Змінні "education" та "default" демонструють незначний негативний зв'язок. Це може означати, що особи з вищою освітою рідше доходять до дефолту.
- Змінні "employ" та "default" демонструють незначний негативний зв'язок. Особи з більшим досвідом роботи рідше виявляють дефолт.
- Змінні "adress" (термін проживання за адресою) та "default" демонструють незначний негативний зв'язок. Це може вказувати на те, що особи, які тривалий час проживають за однією адресою, частіше мають змогу виплачувати борги.
- Змінні "income" (дохід щомісяця) та "default" мають значний негативний взаємозв'язок. Це може вказувати на те, що особи з вищею заробітною платою частіше мають значно меншу ймовірність несплати боргів.
- Змінні "debt_to_income" (частка боргів від доходу) та "default"

демонструють значний позитивний зв'язок. Можна дійти припущення, що особи з вже існуючим борговим навантежнням (відносно доходу) частіше мають набагато більшу ймовірність несплати.

- Змінні "credit_card_debt" (обсяг боргів за кредитною карткою) та "default" демонструють значний позитивний зв'язок. Люди з більшим навантаженням боргів по кредитній картці мають набагато більший ризик несплати.
- Змінні "other_debts" (інші борги) та "default" демонструють значний позитивний зв'язок. Як і в попередніх випадках з боргами – будь-яке боргове навантаження збільшує ризик дефолту.

Було створено та навчено модель логістичної регресії, використовуючи вищезгаданий датасет. Процес створення та навчання представлено на рисунках 3.7 – 3.9.

[15]:
```
target = df.drop("default", axis=1)


values = df.default.values
```

[16]:
```
target.head()
```

[16]:

	age	ed	employ	address	income	debtinc	creddebt	othdebt
0	41	3	17	12	176	9.3	11.359392	5.008608
1	27	1	10	6	31	17.3	1.362202	4.000798
2	40	1	15	14	55	5.5	0.856075	2.168925
3	41	1	15	14	120	2.9	2.658720	0.821280
4	24	2	2	0	28	17.3	1.787436	3.056564

[17]:
```
values
```

[17]:
```
array([1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
```

Рисунок 3.7 – створення датафрейму без значення default та масиву значень default

```
[20]: # Split into train & test set
target_train, target_test, values_train, values_test = train_test_split(
    target,
    values,
    test_size = 0.2
)
```

Рисунок 3.8 – розбиття даних на тренувальні та тестові

Initial training

```
[19]: np.random.seed(42)
```

```
[21]: model = LogisticRegression()
model.fit(target_train, values_train)
model_score = model.score(target_test, values_test)
model_score
```

Рисунок 3.9 – створення об’єкту моделі логістичної регресії та її тренування

3.2 Оцінювання прогностичної здатності моделі

В якості способу оцінювання прогностичної якості моделі застосовано ROC-аналіз (Receiver Operator Characteristics). Будь-який бінарний класифікатор являє собою модель. Класифікатор включає в себе параметр - поріг відсікання, змінюючи який, ми отримуємо різні поділення на два класи: клас з позитивними результатами та клас з від’ємними результатами. У нашому прикладі позитивним результатом є наявність дефолту у клієнта, а від’ємним результатом - його відсутність.

Для поліпшення розуміння зазначеного раніше була створена матриця помилок класифікації (confusion matrix), що показана на рисунку 3.11.

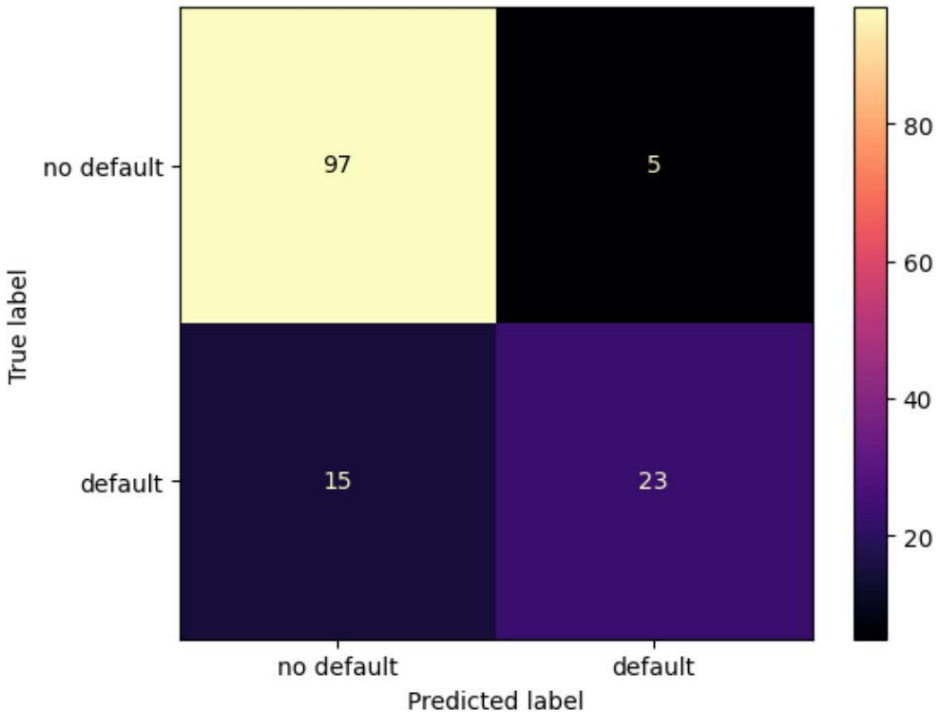

Рисунок 3.10 – Confusion matrix

Матриця, зображена на рисунку 3.10, була створена на даних з тестової частини датасету. На матриці по осі ординат - True label – показує кількість боржників, які насправді зтикнулися з дефолтом та які насправді не стикнулися з ним; по осі абсцис - Predicted label – показує кількість боржників, які згідно прогнозу стикнулися з дефолтом та які згідно прогнозу не стикнулися із ним.

Confusion matrix для бінарної вихідної змінної складається з 4 компонентів – True Positives (події, коли дійсна вихідна змінна дорівнює 1, а класифікована також дорівнює 1), False Positives (події, коли дійсна становить 0, а класифікована - 1), True Negatives (події, коли дійсна - 0, а класифікована - 0) та False Negatives (події, коли дійсна - 1, а класифікована - 0).

Два показники вираховуються на базі Confusion Matrix. Перший - TPR (True Positive Rate або Чутливість – частка коректно ідентифікованих випадків, де вихідна змінна дорівнює 1). Мова йде про вміння моделі коректно ідентифікувати «неблагонадійних» позичальників і зменшувати

втрати, які виникають через кредитування недобросовісних клієнтів. Модель з високою чутливістю має вищий ризик «помилкових спрацьовувань». Другий - FPR (False Positive Rate або Специфічність – частка неправильно ідентифікованих випадків, де вихідна змінна дорівнює 0). Тут мова йде про вміння моделі коректно ідентифікувати «хороших» позичальників і зменшувати втрати, пов’язані з відмовою у кредитуванні добросовісного клієнта. Модель з великою специфічністю має вищу ймовірність «помилкових позитивів».

На графіку ROC-кривої, показаному на рисунку 3.11, на осі Y розміщується Sensitivity (істинно позитивні випадки), на осі X - або Specificity (істинно негативні випадки), або 1 мінус Specificity (хибно позитивні випадки). Чим більше вигин має крива (наближаючись до верхнього лівого кута), тим вища прогностична здатність моделі. Навпаки, чим ближче вона знаходиться до діагональної прямої, тим менш ефективною є модель.

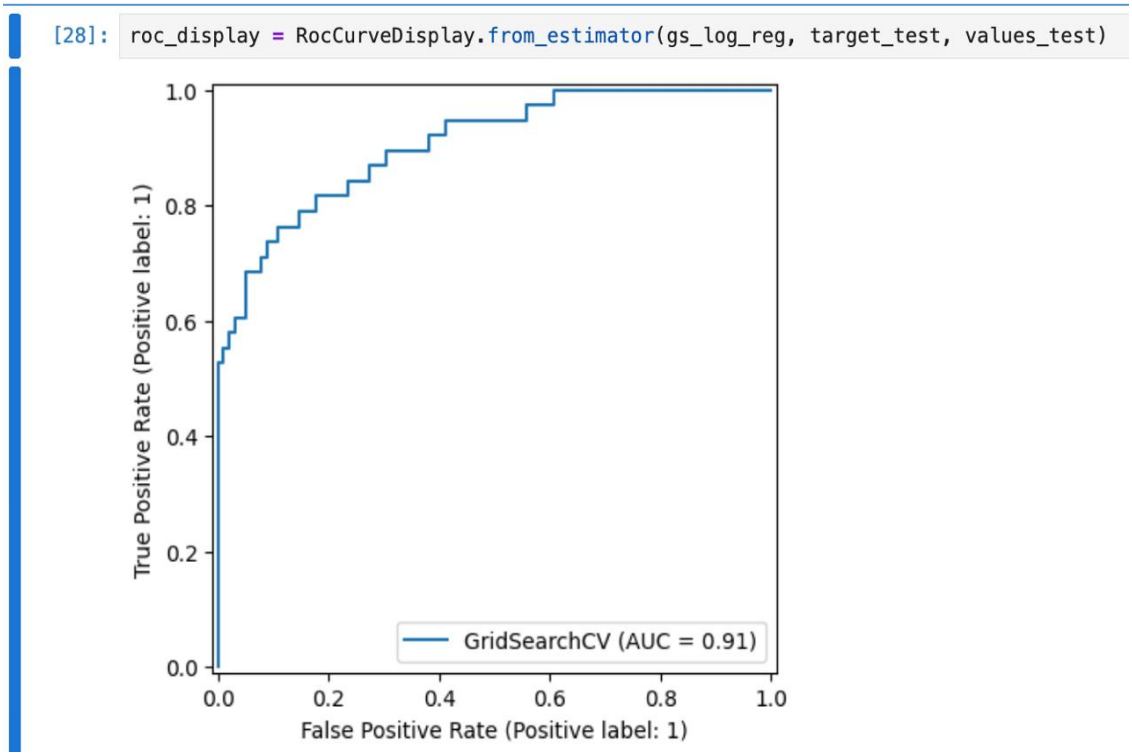

Рисунок 3.11 – ROC

На рисунку 3.11 також можна побачити значення AUC (Area Under Curve), яке набуває значення в межах [0; 1] і демонструє обчислене значення площі під ROC-кривою.

Для досконалого класифікатора ROC-крива проходить через верхній лівий кут, де частка істинно позитивних випадків дорівнює 100% або 1.0 (досконала чутливість), а частка помилкових позитивних прикладів дорівнює нулю. Оцінка площі під ROC-кривими (Area Under Curve або AUC) є унікальним способом їх порівняння. Теоретично вона варіюється від 0 до 1.0, але оскільки модель завжди представлена кривою, що лежить вище позитивної діагоналі, зазвичай зазначають зміни від 0.5 ("недоцільний" класифікатор) до 1.0 ("досконала" модель). Цю оцінку можна отримати шляхом прямого обчислення площі під многогранником, який обмежений праворуч і знизу осями координат, а зліва вгорі – експериментально визначеними точками. Згідно з ROC-кривою та значенням AUC – 0.91, ця модель демонструє високу ефективність у плані прогностичної точності.

3.2 Порівняння результатів із іншими моделями

В попередніх розділах було згадано такі моделі машинного навчання, як Random Forest, Gradient Boosting та Support Vector Classifier і було обґрунтовано їх вибір для порівняння. Було побудовано моделі таким же самим чином як і логістичну регресію. На рисунках 3.12, 3.14 та 3.14 можна побачити матриці похибок для моделей, а на рисунках 3.13, 3.15 та 3.17 – ROC-криві та відповідно AUC.

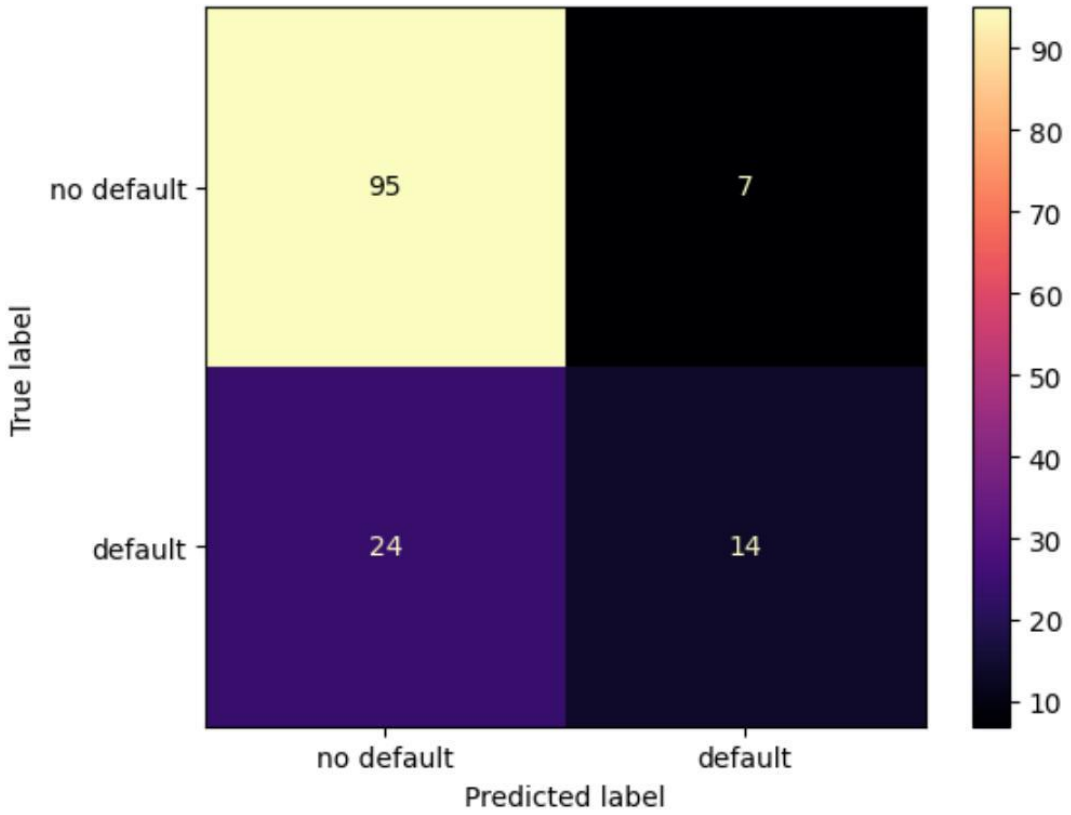

Рисунок 3.12 – Confusion matrix для моделі Random Forest

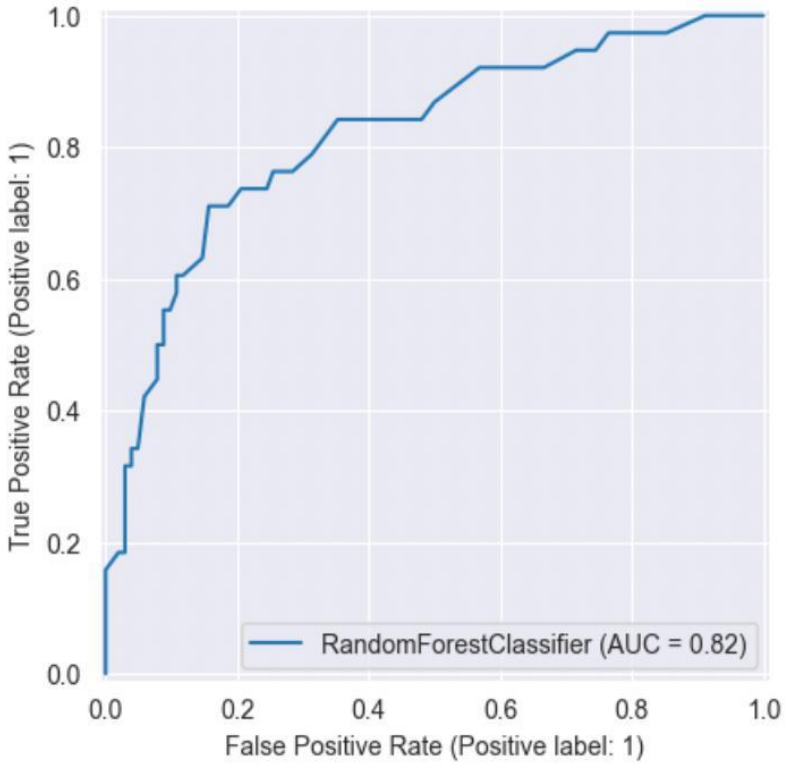

Рисунок 3.13 – ROC для моделі Random Forest

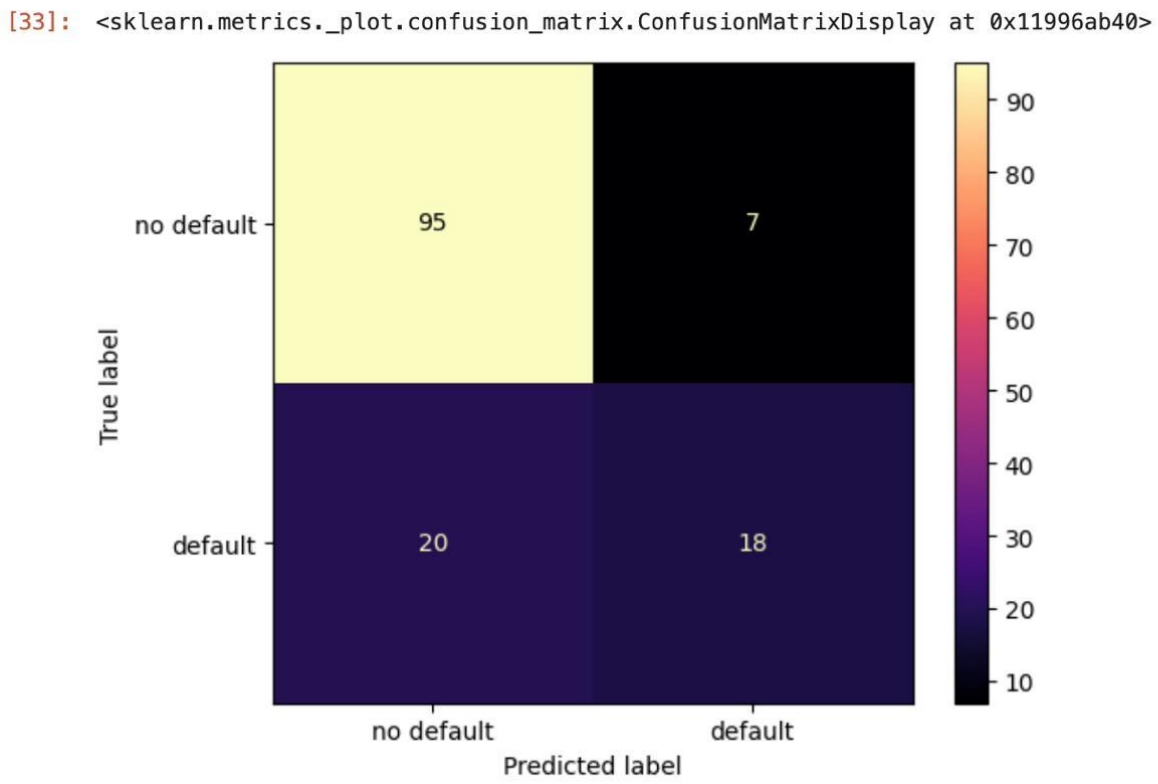

Рисунок 3.14 – Confusion matrix для моделі Gradient Boosting

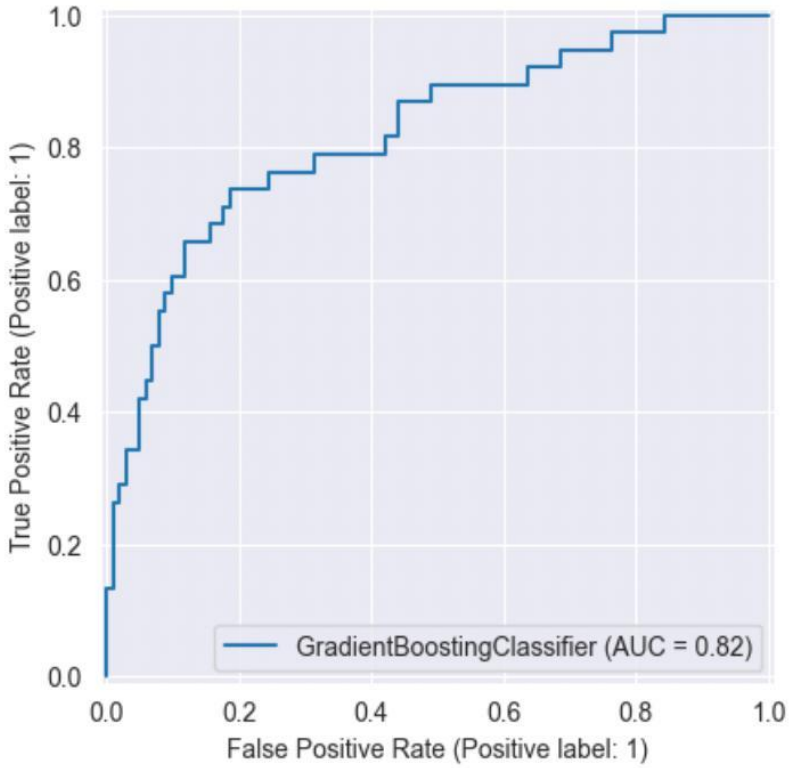

Рисунок 3.15 – ROC для моделі Gradient Boosting

```
[36]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x11989ac30>
```

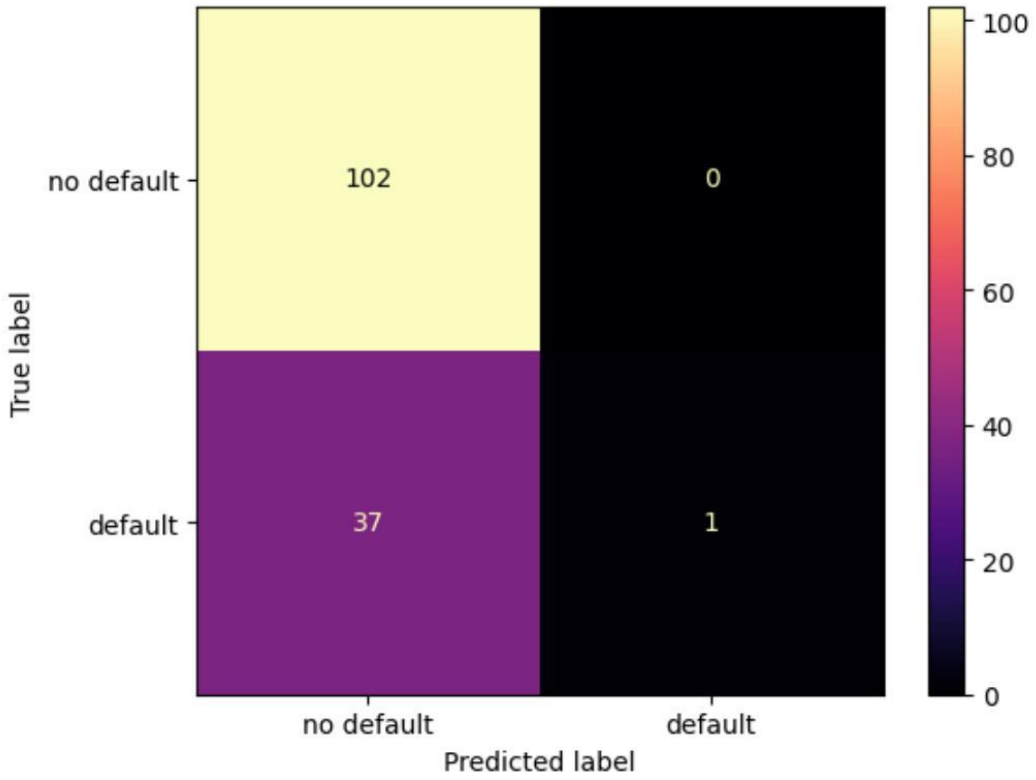

Рисунок 3.16 – Confusion matrix для моделі SVC

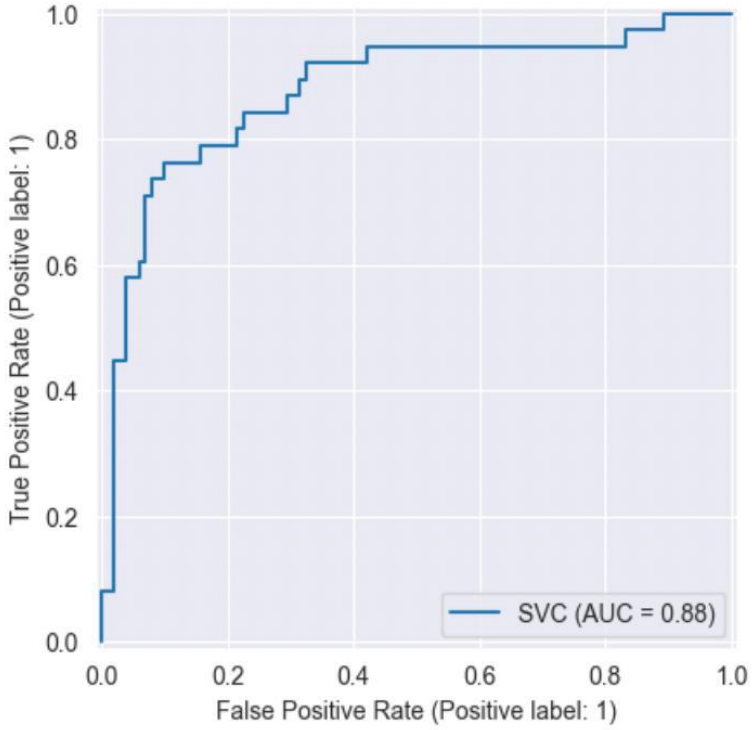

Рисунок 3.17 – ROC для моделі SVC

На основі проведеного експерименту логістична регресія продемонструвала кращу якість прогнозування кредитного дефолту порівняно з трьома альтернативними моделями машинного навчання – Random Forest (метод випадкового лісу), Gradient Boosting (градієнтний бустинг) та SVC (Support Vector Classifier, метод опорних векторів). Зокрема, логістична регресія досягла найвищого значення AUC (площі під ROC-кривою) та найменшої кількості хибнопозитивних класифікацій. Для порівняння, AUC моделей Random Forest і Gradient Boosting склав лише ~0,82, для SVC – близько 0,88, тоді як у логістичної регресії цей показник був вищим (тобто модель краще розрізняла платоспроможних і неплатоспроможних клієнтів).

Аналогічно, кількість хибно позитивних спрацьовувань (False Positives, FP) – випадків, коли модель помилково віднесла надійного позичальника до групи ризику – в альтернативних моделей була значно більшою (21 у Random Forest, 20 у Gradient Boosting та 37 у SVC) порівняно з логістичною регресією, яка дала найменше FP. Іншими словами, проста логістична модель помилялася рідше при ідентифікації «безпечних» клієнтів як небезпечних, ніж більш складні алгоритми.

Виявлена перевага логістичної регресії за кількістю хибнопозитивних помилок має важливе практичне значення для задач прогнозування дефолту. Високий рівень FP означає, що більше «невинних» клієнтів помилково позначаються як ризиковані. У контексті кредитування кожен такий хибнопозитивний випадок може призводити до відмови у позиці або до невиправдано жорстких умов для клієнта, який насправді здатен вчасно обслуговувати борг. Це не тільки створює незручності для благонадійних позичальників, але й несе втрати для банку у вигляді втраченої вигоди та може негативно вплинути на репутацію фінансової установи.

Модель з меншою кількістю FP (як-от логістична регресія в даному випадку) є більш ефективною в даному випадку, оскільки вона краще збалансовує виявлення ризикових клієнтів із максимальним залученням

надійних позичальників. Поряд з цим, вищий AUC логістичної регресії підтверджує її здатність більш ефективно ранжувати клієнтів за рівнем ризику дефолту. Модель з високим AUC надійніше розрізняє дефолтних і платоспроможних боржників при різних порогах, що важливо для кредитного скорингу, де треба чітко відділяти групу високого ризику.

Причини кращої ефективності логістичної регресії. Незважаючи на відносну простоту, логістична регресія може перевершувати більш складні алгоритми у практичних задачах з кількох причин. По-перше, характеристика задачі може бути близькою до лінійної. Логістична регресія являє собою лінійну модель і припускає лінійний зв’язок між вхідними ознаками та логарифмом шансів дефолту.

Якщо в реальності вплив ознак на ймовірність дефолту є майже лінійним (або складні нелінійні взаємодії мінімальні), тоді проста модель на кшталт логістичної регресії здатна адекватно захопити ці залежності. Більш гнучкі моделі – такі як Random Forest, Gradient Boosting чи SVC – у такому випадку не отримують значної додаткової вигоди, зате можуть почати підлаштовуватися під шум даних. Це підводить до другої причини: логістична регресія має менший ризик перенавчання (overfitting) порівняно з ансамблевими методами, особливо на обмежених вибірках.

Простота логістичної моделі означає меншу кількість параметрів і більш жорсткі допущення, що діє як своєрідний запобіжник від надмірного підгонування моделі під тренувальні дані. Навпаки, моделі Random Forest та Gradient Boosting містять сотні дерев рішень і безліч параметрів, а SVC використовує складні нелінійні перетворення – це дає їм велику виразну потужність, але за недостатнього обсягу даних чи неправильної калібрації вони можуть підхоплювати випадкові структури, що не повторюються на тестових даних. У даному випадку кращі результати логістичної регресії свідчать, що вона краще узагальнила закономірності в даних, тоді як складніші алгоритми ймовірно частково перенавчилися, що і відобразилось у нижчому AUC і

більшій кількості помилкових спрацьовувань.

По-третє, вагомою перевагою логістичної регресії є її інтерпретованість. На відміну від методів «чорного ящика», логістична модель дозволяє безпосередньо оцінити вклад кожної ознаки через коефіцієнти при змінних, які легко переводяться в відношення шансів (odds ratio).

Така прозорість надзвичайно важлива у сфері кредитного ризику, де регулятори та аудитори вимагають зрозумілості рішень моделі.

Отримані результати демонструють, що простіші моделі можуть перевершувати складніші алгоритми в задачах кредитного скорингу, коли структура даних цьому сприяє. Логістична регресія показала найкращий баланс між точністю та помилками класифікації: вона забезпечила найвищий AUC (тобто кращу диференціацію ризику) і найменше хибних тривог щодо благонадійних клієнтів. Причини цього криються у відповідності лінійної моделі природі даних, в кращій узагальнювальній здатності (меншому перенавчанні) та високій інтерпретованості, що є критичними факторами у фінансовому домені. Отже, навіть за наявності сучасних потужних методів машинного навчання, логістична регресія як базовий підхід залишається надзвичайно ефективним і надійним інструментом для прогнозування кредитного дефолту в прикладних умовах.

4. ПОБУДОВА ВЕБ-ЗАСТОСУНКУ ТА ВПРОВАДЖЕННЯ МОДЕЛІ

4.1 Опис архітектури системи

Для створення веб-додатку з можливістю автоматизованої оцінки кредитоспроможності застосовувалася клієнт-серверна архітектура. Архітектура включає дві ключові компоненти – це клієнт і сервер[16].

У цьому випадку браузер функціонує як клієнт, за допомогою якого здійснюється взаємодія з сервером. Браузер слугує інтерфейсом для користувача. Інтерфейс виступає як засіб подання даних, так і інструмент для маніпуляцій над ресурсами користувача..

Сервер, в свою чергу, відіграє роль посередника між клієнтським додатком та базою даних. Сервер виконує функцію отримання запитів від клієнта, обробки даних у сховищі залежно від типу запиту та формування відповіді для клієнта. У створюваній системі сервер додатково застосовує розроблену модель логістичної регресії для визначення кредитоспроможності.

Система бази даних є місцем, де виконуються операції з даними: їх додавання, оновлення, читання, видалення та зберігання. У розроблюваній системі дані включають інформацію про користувачів та їх звіти за оцінкою кредитних ризиків.

RESTful API – це набор правил побудови API у клієнт-серверній архітектурі, який засновано на принципах Representational State Transfer. Його імплементація згодом надає змогу масштабувати сервіси а також інтегрувати інші нові сервіси.

Систему було створено, дотримуючись архітектури RESTful API. Метою RESTful API є просте і зрозуміле забезпечення доступу до даних і функціоналу веб-сервісу. Це реалізується через стандартні HTTP-методи (GET, POST, PUT, DELETE) та URI, які описують ресурси.

RESTful API стандартизує використання веб-сервісів, спрощує підтримування та розширення.

Властивості архітектури, які залежать від обмежень, накладених на REST-

системи:

1) Client-Server. Система повинна бути розділена на клієнтів та сервер(и). Розподіл інтерфейсів означає, що користувачі не пов’язані зі сховищем даних, яке залишається в межах кожного сервера, завдяки чому мобільність коду користувача підвищується. Сервери не мають зв'язку з інтерфейсом користувача чи станом, тому їх можна зробити простішими та легшими для масштабування. Сервери й клієнти можуть бути замінені та розроблятися автономно, за умови, що інтерфейс залишається незмінним.
2) Stateless. Сервер не має зберігати жодної інформації про клієнтів. У запиті має бути вся важлива інформація для обробки запиту та, за потреби, ідентифікації клієнта.
3) Кешування. Кожна відповідь повинна бути маркована, чи вона кешується, щоб уникнути повторного використання клієнтами застарілої або некоректної інформації у відповідь на наступні запити.
4) Єдиний інтерфейс. Встановлює зв'язок між клієнтами та серверами. Це полегшує та ізолює архітектуру, що дає можливість кожному компоненту розвиватися автономно. Чотири основні принципи єдиного інтерфейсу:
    a. Стандартизовані методи. Клієнти взаємодіють із сервером через набір стандартних HTTP-методів, які визначають операції, що можуть виконуватися з ресурсами..
    b. Ідентифікація ресурсів: Ресурси ідентифікуються за допомогою URI (уніфікованих ідентифікаторів ресурсів), що забезпечує чіткий та узгоджений спосіб доступу до різних об'єктів..
    c. Представлення ресурсів: Ресурси представлені за допомогою стандартних форматів, таких як JSON, XML або HTML, що дозволяє клієнтам і серверам розуміти та обробляти дані однаковим чином.

d. Взаємодії без збереження стану: кожен запит від клієнта до сервера повинен містити всю інформацію, необхідну для розуміння та обробки запиту, що гарантує, що серверу не потрібно підтримувати стан сеансу.

У створюваному веб-додатку спілкування між клієнтом і сервером реалізується через протокол HTTP. У веб-застосунку були застосовані HTTP запити таких методів:

- POST – для генерування об’єкту (ресурсу);
- GET – для зчитування об’єкту (ресурсу);
- PATCH – для часткового оновлення об’єкту (ресурсу);
- DELETE – видалити об’єкт (ресурс).

У процесі створення системи в ролі комунікаційної форми між клієнтом і сервером було вибрано JSON (JS object notation). Тобто репрезентація ресурсів та подальші маніпуляції над ними реалізовано за домогою цього формату.

4.2 Обгрунтування вибору програмного сервісу для реалізації функції автоматизованої оцінки кредитного скорингу

Для створюваної системи була обрана реалізація з бібліотеки scikit-learn, що є бібліотекою для мови Python і належить до бібліотек машинного навчання.

Цю бібліотеку вибрано через кілька переваг, що надаються при застосуванні моделі логістичної регресії саме з scikit-learn, а саме – докладна документація з практичними прикладами побудови, поліпшення, оцінювання та різних методів навчання моделі. Ще однією важливою причиною обрання цієї бібліотеки є наявність великого ком’юніті, а отже, численних посібників та матеріалів від досвідчених фахівців щодо використання її можливостей. Слід також підкреслити, що простий інтерфейс і швидкість функціонування бібліотеки обумовлені реалізацією її коду на мові C.

Модель генерує прогноз про дефолт або його відсутність на основі таких вхідних даних:

- Age (вік позичальника);
- Education (освіта позичальника);
- employ years (досвід роботи позичальника на поточному місці роботи);
- address years (скільки років позичальник живе за поточною адресою);
- income (заробітна плата);
- debt_to_incomce (співвідношення боргів до заробітної плати);
- credit_debt (обсяг боргового навантаження за кредитною карткою);
- other_debt (обсяг усіх інших боргів).

Серверну частину було розроблено за допомогою:

- Python;
- Fastapi (ASGI бібліотека, вибір зумовлений підтримкою асинхронного коду, підтримка Pydantic для валідації даних та зручною системою впровадження залежностей) [17];
- SQLAlchemy (ORM бібліотека зі зручним та багатим на функціонал інтерфейсом) [18];
- Alembic (автоматично генерує міграції до бази даних та виконує їх) [19].

Клієнтська частина була побудована з використанням наступних технологій:

- JavaScript;
- React (швидка бібліотека розроблена Facebook, компоненти якої поєднуть в собі JS та HTML код у форматі JSX);
- React Router V6 (просунутий маршрутизатор на клієнтській частині додатку) [20];
- Redux (надає простіший спосіб управління станом додатку);
- React Material UI (готові CSS компоненти, побудовано на основі Bootstrap та Material CSS фреймворків) [21].

4.3 Логічне і фізичне моделювання даних системи

Логічну модель даних зображено на рисунку 4.1.

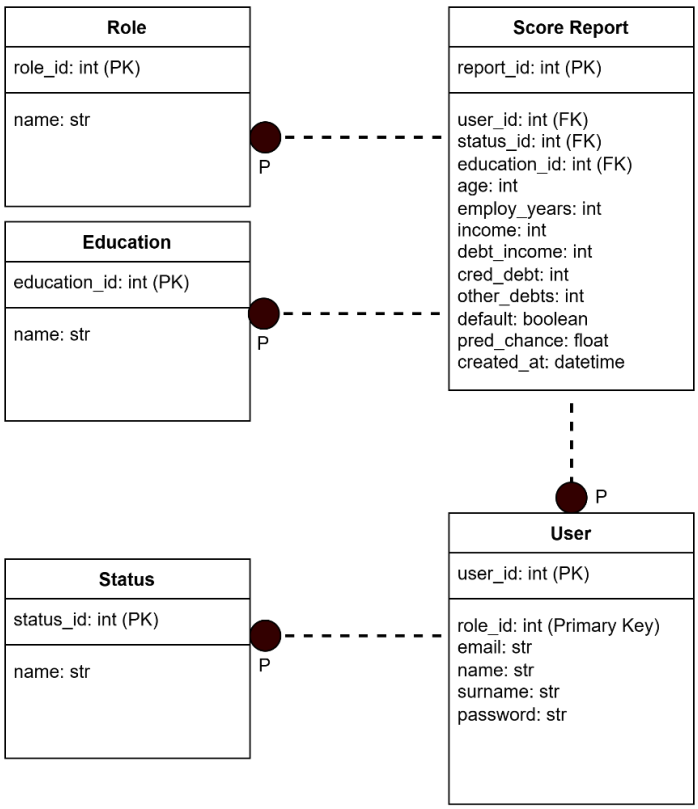

Рисунок 4.1 – логічна модель даних

В ході розробки логічної моделі було визначено наступні сутності:

- сутність «finance report» – містить дані про звіт (вхідні показники для виконання прогнозування дефолту, результат прогнозування дефолту, та відсоток впевненості у коректному прогнозуванні);
- сутність «user» – містить особисту інформацію про користувача, який користується системою;
- сутність «role» – містить інформацію про наявні ролі користувачів;
- сутність «status» – містить інформація про наявні статуси фінансового звіту у системі;
- сутність «education» – містить інформацію про наявні типи освіти у системі (у подальшому ті – на яких було побудовано модель);

Була розроблена фізична модель даних, яка демонструє Primary Key та Foreign Key, атрибути, їх типи даних а також «NULL» і «NOT NULL», кардинальність зв’язків. Розроблену модель зображено на рисунку 3.14.

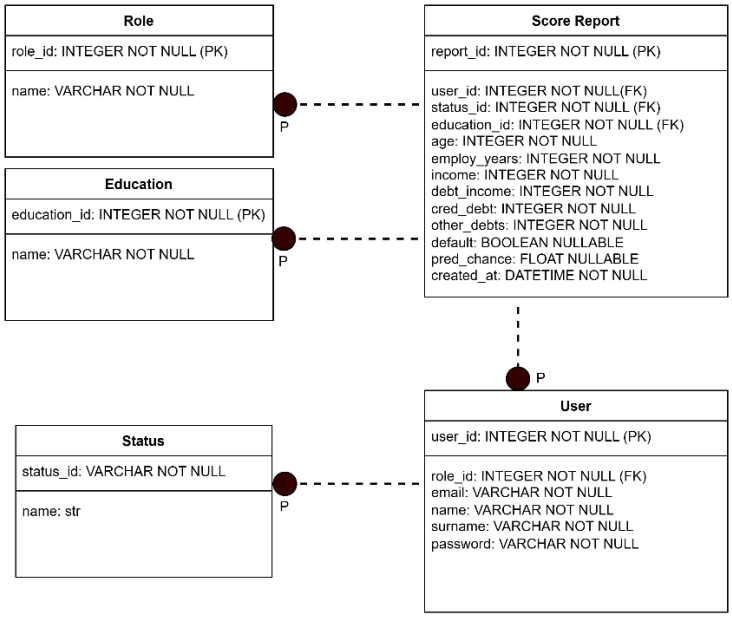

Рисунок 4.2 – Фізична модель даних

Для кожної сутності встановлені потрібні атрибути з визначеним типом даних, більшість атрибутів не дозволяють значення NULL.

4.4 Розробка класів, методів класів

В якості архітектури програмної частини серверу було обрано підходи з Domain Driven Design (DDD). Принцип цього підходу полягає в виокремленні бізнес логіки таким чином, що ця логіка стає незалежною від конкретної реалізації. Таким чином уся логіка залежить від абстракцій, а при виконанні певних сценаріїв впроваджуються дійсні імплементації цих абстракцій.

Далі буде наведено частини коду, які були реалізовані у серверній частині і які найкраще відображають філософію цього підходу. А саме – приклад доменної моделі, приклад імплементації патерну Unit of Work, та приклад SQLAlchemy імплементації патерну репозиторія [22, 23, 24]:

- доменна модель користувача:

```
class User(BaseModel):
        """

        Domain model of User
        """

        user_id: int
        user_email: str

        user_role_id: int

        user_first_name: str
        user_last_name: str

        user_password: str

        class Meta:
            """
            This is required for proper mapping of orm attrs
            To pydantic model attrs
            """
            mode_orm = True
```

- репозиторій користувача (визначені методи add, get, list, delete):

```
class PSQLUserRepository(AbstractUserRepository):
    def __init__(self, db_session: AsyncSession) -> None:
        self.db_session = db_session

    async def add(self, user_to_add: model.User, /) -> None:
        user = User(
            id=user_to_add.user_id,
            email= user_to_add.user_email,
            first_name= user_to_add.user_first_name,
            last_name= user_to_add.user_last_name,
            password= user_to_add.user_password,
            role_id= user_to_add.user_role_id,
        )
        await self.db_session.merge(user)
        await self.db_session.flush()

    async def get(self, user_id: int = None, user_email: str = None) ->
model.User:
        stmt = select(User)

        for attr, col in ((user_id, User.id), (user_email, User.email)):
            if attr:
```

```
                stmt = stmt.where(col == attr)

        result = await self.db_session.execute(query)

        if (retrieve_user := result.scalars().one_or_none()) is not None:
            return model.User.get_from_orm(retrieve_user)
        raise UserNotFoundException(user_id)

    async def list(self, page: int = None, size: int = None) ->
Sequence[model.User]:
        stmt = select(User)
        if page and size:
            query = query.offset((page – 1) * size).limit(size)

        return from_sequence_orms_to_internal(
           User,
           (await self.db_session.execute(stmt)).scalars().all(),
        )

    async def delete(self: 'PSQLUserRepository', user_id: int) -> None:
        if select(User).where(model.User.user_id == id_) is None:
            raise UserNotFoundException(user_id)
        await self.session.delete(user)
```

- доменна модель звіту:

```
class DebtScoringReport(BaseModel):
        """
        Domain model of DebtScoringReport
        """


        id: int

        age: int = Field(ge=0)
        employ_years: int = Field(ge=0)
        address_years: int = Field(ge=0)

        user_id: int
        education_id: int
        status_id: int

        income: int = Field(ge=0)
        debtinc: float = Field(ge=0)
        creddebt: float = Field(ge=0)

        othdebt: float = Field(ge=0)
        default: bool
        prediction_chance: float
```

```
        class Config:
            """
            This is required for proper mapping of orm attrs
            To pydantic model attrs
            """
            mode_orm = True
```

- UOW для SQLAlchemy:

```
class PSQLUnitOfWork:
    users: PSQLUserRepository

    def __init__(self, db_session_factory: DBAsyncSessionFactory) ->
None:
        """
        Init method accepts session maker object (session factory).
        """
        self.db_session_factory = db_session_factory

    def __enter__(self) -> PSQLUnitOfWork:
        """
        This method is called when entering context manager.
        Should populate users and scoring report repositories with db
        session object.
        """
        self.db_session = self.db_session_factory()
        self.users = PSQLUserRepository(self.db_session)
        self.scoring_reports = =
PSQLScoringReportRepository(self.db_session)
        return self

    def __exit__(self, *args: Tuple[Any], **kwargs: dict[str, Any]) ->
None:
        """
        This method is called when program exits context manager
        """
        self.db_session.exit()

    def commit(self) -> None:
        """
        Method to perform commit. Happens explicitly only
        """
        self.db_session.make_commit()

    def rollback(self) -> None:
        """
        Method for rolling back transactions
```

```
"""
 self.db_session.make_rollback()
```

Доменні моделі - це центральна частина системи, яка описує бізнес-логіку та правила предметної області. Вона представляє об’єкти, їх поведінку та взаємозв’язки. Основна мета — відобразити сутність реального світу в коді через єдину «мову домену», якою користуються і розробники, і експерти домену.

Репозиторій виступає посередником між доменною моделлю та шаром зберігання даних. Він інкапсулює деталі доступу до бази даних або інших джерел даних, дозволяючи доменним об’єктам працювати з ними як із колекціями об’єктів у пам’яті. Таким чином, репозиторії забезпечують чистоту доменної моделі, ізолюючи її від технічних аспектів зберігання.

У класі UOW зазвичай відповідає за управління транзакціями [25]. Він відстежує зміни зроблені над ресурсами під час цієї транзакції. Він збирає усі зміни (створення, оновлення, видалення) і забезпечує їх атомарне збереження в базі даних. Це гарантує узгодженість даних і спрощує керування транзакціями у складних системах.

ВИСНОВКИ

На сьогоднішній день у фінансовій сфері активно застосовуються моделі машинного навчання для автоматизації прийняття рішень, управління ризиками та персоналізації послуг. Сучасні дослідження відзначають, що впровадження штучного інтелекту (ШІ) і машинного навчання дозволяє банкам виявляти приховані фінансові ризики та підвищувати точність прогнозування. Прикладом є алгоритми логістичної регресії, які широко використовуються у банківській сфері для оцінювання кредитоспроможності позичальників, що підвищує ефективність ухвалення рішень щодо видачі кредитів.

У рамках даної магістерської роботи була досліджена, побудована і оцінена модель логістичної регресії для оцінки кредитних ризиків із подальшим впровадженням в побудований веб-застосунок «Банківський асистент». Проведений аналіз предметної області дозволив виявити основні проблеми та потреби користувачів у процесі кредитного скорингу, що визначило вимоги до системи. В ході виконання було досліджено та порівняно на практичному і теоритичному рівнях логістичну регресію із іншими моделями машинного навчання в рамках поставлених задач.

Моделювання показало високі результати точності та ефективності прогнозування. Зокрема, побудована логістична модель продемонструвала здатність надійно класифікувати фінансову поведінку клієнтів за допомогою заданих параметрів. За результатами оцінювання якість моделі є високою. У разі потреби, модель можна порівняти чи поєднати з іншими алгоритмами (деревами рішень, випадковими лісами тощо), але вже отримані метрики свідчать, що обрана модель є достатньо точною та надійною для поставленої задачі.

Розроблена система автоматизує оцінку кредитних ризиків і здатна суттєво прискорити процес ухвалення рішень щодо кредитування. Інтеграція логістичної регресії у веб-застосунок продемонструвала, що такий підхід може бути успішно реалізований у реальних умовах експлуатації банківських ІТ-систем.

Сфера потенційного застосування створеної системи охоплює банківські, фінансові та мікрофінансові організації, де необхідно автоматизувати оцінку кредитоспроможності і керувати ризиками при наданні позик. Такий інструмент може використовуватися як співробітниками фінансових установ (для підтримки процесу прийняття рішень), так і безпосередньо клієнтами, які планують отримати кредит, – останнім це дозволяє попередньо прогнозувати свої шанси на позитивне рішення щодо позики. Разом з тим, застосування логістичної регресії має і свої обмеження. Модель є лінійною за природою, тому може не враховувати складні нелінійні взаємозв'язки між ознаками клієнтів і їхньою фінансовою поведінкою. Для підтримання високої точності в довгостроковій перспективі необхідно регулярно оновлювати модель новими даними, щоб вона враховувала актуальні зміни в економічному середовищі та поведінці позичальників. Подальші дослідження можуть бути спрямовані на розширення набору вхідних даних (наприклад, врахування поведінкових чи макроекономічних показників), а також на експериментальне порівняння логістичної регресії з більш складними алгоритмами (ансамблевими методами, нейронними мережами) з метою підвищення точності прогнозування. Важливим напрямом розвитку є інтеграція розробленої моделі в масштабні банківські IT-інфраструктури та оцінка її продуктивності й витривалості під реальними навантаженнями. Таким чином, впроваджена модель логістичної регресії вже підтвердила свою ефективність у вирішенні задач прогнозування фінансової поведінки клієнтів, а її подальше вдосконалення та адаптація до нових умов дозволять ще більш повно реалізувати потенціал машинного навчання у банківській практиці.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Поляруш І. М. Скоринг, як вдосконалений механізм оцінки потенційного позичальника банком – демонстрація процесу обробки даних // Ефективна економіка. URL : http://www.economy.nayka.com.ua/?op=1&z=4510 (дата звернення: 30.11.2025)

2. Machine learning powered financial credit scoring: вебсайт. URL: https://link.springer.com/article/10.1007/s10462-025-11416-2 (дата звернення 29.11.2025)

3. Visual introduction to linear regression: вебсайт. URL: https://mlu-explain.github.io/linear-regression/ (дата звернення 30.11.2025)

4. The mathematics behind linear regression: вебсайт. URL: https://pujappathak.medium.com/the-mathematics-behind-linear-regression-fb4db1ebd7b5 (дата звернення 01.12.2025)

5. How to interpret a confusion matrix for a machine learning model: вебсайт. URL: https://www.evidentlyai.com/classification-metrics/confusion-matrix (дата звернення 01.12.2025)

6. AUC and the ROC curve in Machine Learning: вебсайт. URL: https://www.datacamp.com/tutorial/auc (дата звернення 02.12.2025)

7. Random Forest Explain: . URL: https://medium.com/data-science/random-forest-explained-a-visual-guide-with-code-examples-9f736a6e1b3c (дата звернення 02.12.2025)

8. Support Vector Machine (SVC) Algorithm: вебсайт. URL: https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/ (дата звернення 02.12.2025)

9. Gradient Boosting in ML: вебсайт. URL: https://www.geeksforgeeks.org/machine-learning/ml-gradient-boosting/ (дата звернення 02.12.2025)

10. Demystifying Maths of Gradient Boosting: вебсайт. URL: https://medium.com/data-science/demystifying-maths-of-gradient-boosting-bd5715e82b7c (дата звернення 02.12.2025)

11. The Mathematics Behind Support Vector Machine (SVC) Algorithm: вебсайт. URL: https://www.analyticsvidhya.com/blog/2020/10/the-mathematics-behind-svm/ (дата звернення 02.12.2025)

12. A Comprehensive Guide to Data Analysis using Pandas: вебсайт. URL: https://www.analyticsvidhya.com/blog/2021/05/a-comprehensive-guide-to-data-analysis-using-pandas-hands-on-data-analysis-on-imdb-movies-data/ (дата звернення 02.12.2025)

13. Python Data Analysis with Pandas and Matplotlib: вебсайт. URL: https://ourcodingclub.github.io/tutorials/pandas-python-intro/ (дата звернення 03.12.2025)

14. Pandas and Seaborn – A guide to handle & visualize data in Python: вебсайт. URL: https://tryolabs.com/blog/2017/03/16/pandas-seaborn-a-guide-to-handle-visualize-data-elegantly (дата звернення 03.12.2025)

15. Introduction to the Correlation Matrix: вебсайт. URL: https://builtin.com/data-science/correlation-matrix (дата звернення 03.12.2025)

16. REST API Principles: . URL: https://blog.dreamfactory.com/rest-apis-an-overview-of-basic-principles (дата звернення 03.12.2025)

17. Building RESTful APIs with FastApi: вебсайт. URL: https://www.geeksforgeeks.org/python/building-restful-apis-with-fastapi/ (дата звернення 03.12.2025)

18. Scalable APIs with FastApi and SQLAlchemy: вебсайт. URL: https://medium.com/@suganthi2496/fastapi-with-sqlalchemy-building-scalable-apis-with-a-database-backend-7ccc9aa659a1 (дата звернення 04.12.2025)

19. Handling database migrations with Alembic: вебсайт. URL: https://testdriven.io/blog/alembic-database-migrations/ (дата звернення 04.12.2025)

20. Comprehensive Guide to React Router: . URL: https://pieces.app/blog/react-router-v6-a-comprehensive-guide-to-page-routing-in-react (дата звернення 04.12.2025)

21. Managing state in React app via Redux: вебсайт. URL: https://medium.com/@info_52759/redux-a-state-management-guide-908a2339031c (05.12.2025)

22. Python Domain Driven Design: вебсайт. URL: https://www.cosmicpython.com/book/chapter_01_domain_model.html (дата звернення 04.12.2025)

23. Building Maintainable Python applications with Hexagonal Architecture and Domain-Driven-Design: . URL: https://dev.to/hieutran25/building-maintainable-python-applications-with-hexagonal-architecture-and-domain-driven-design-chp (дата звернення 05.12.2025)

24. Domain Driven Design (Examples with Python): вебсайт. URL: https://medium.com/@validate/domain-driven-design-examples-with-python-ce88740e3f26 (дата звернення 05.12.2025)

25. Unit of Work – Python DDD Patterns: вебсайт. URL: https://medium.com/technology-hits/unit-of-work-python-domain-driven-design-patterns-f07a675588ee (дата звернення 05.12.2025)