**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет**
**імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

**ДОПУЩЕНО ДО ЗАХИСТУ**
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

**КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА**

# МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ ВИЗНАЧЕННЯ КРЕДИТОСПРОМОЖНОСТІ КЛІЄНТІВ БАНКУ

Спеціальність 122 «Комп'ютерні науки»

**122 – КРБ – 401.22010115**

***Виконав студент 4-го курсу, групи 401***
***_____Д. В. Мурзакой***
«_17_» червня 2024 р.

***Керівник: канд. пед. наук, доцент***
***_____Н. М. Болюбаш***
«_17_» червня 2024 р.

**Миколаїв – 2024**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет ім. Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра інтелектуальних інформаційних систем**

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
*(шифр і назва)*
Галузь знань **12 «Інформаційні технології»**
*(шифр і назва)*

**ЗАТВЕРДЖУЮ**

Завідувач кафедри інтелектуальних інформаційних систем, д-р. техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

# З А В Д А Н Н Я

**на виконання кваліфікаційної роботи**

Видано студенту групи 401 факультету комп'ютерних наук Мурзакою Данилу Васильовичу.

1.Тема кваліфікаційної роботи «Мобільний застосунок для визначення кредитоспроможності клієнтів банку».

Керівник роботи Болюбаш Надія Миколаївна, канд. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2024 р. № _271_

2. Строк представлення кваліфікаційної роботи студентом «17» червня 2024 р.

3. Вхідні (початкові) дані до роботи: предметна сфера оцінки кредитоспроможності клієнтів банку та методів і моделей їх визначення, набір даних з інформацією про надання кредитів клієнтам банку.

Очікуваний результат: мобільний застосунок для оцінки кредитоспроможності клієнта банку.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз предметної сфери надання кредитів у банках та дослідження теоретичних засад виявлення кредитоспроможності клієнтів;

– обґрунтування вибору технологій і засобів розробки мобільного застосунку для визначення кредитоспроможності фізичних осіб;

– розробка та здійснення програмної реалізації мобільного застосунку для визначення кредитоспроможності клієнтів банку.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

6. Завдання до спеціальної частини: Охорона праці робочого місця фахівців з ІТ-технологій.

7. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис |
|---|---|---|
| Спеціальна частина з охорони праці | канд. техн. наук, доцент А. О. Алексєєва | |

Керівник роботи __канд. пед. наук, доцент Болюбаш Н. М._____
*(наук. ступінь, вчене звання, прізвище та ініціали)*

_____
*(підпис)*

Завдання прийнято до виконання ___Мурзакой Д. В._____
*(прізвище та ініціали)*

_____
*(підпис)*

Дата видачі завдання «__14__» ___січня__ 2024 р.

**КАЛЕНДАРНИЙ ПЛАН**
**виконання кваліфікаційної роботи**

Тема: Мобільний застосунок для визначення кредитоспроможності клієнтів банку

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1 | Визначення керівника і теми КРБ. Подання заяви на затвердження теми КРБ | 10.11.2023 | 15.11.2023 | Виконано |
| 2 | Отримання завдання на виконання КРБ | 10.01.2024 | 15.01.2024 | Виконано |
| 3 | Складання календарного плану | 16.01.2024 | 29.01.2024 | Виконано |
| 4 | Огляд літератури за темою дослідження. Аналіз сучасних методів оцінки кредитоспроможності клієнтів банку | 30.01.2024 | 17.02.2024 | Виконано |
| 5 | Вибір технологій та інструментальних засобів розробки системи | 18.02.2024 | 29.02.2024 | Виконано |
| 6 | Створення дизайну, проєктування та програмна реалізація, тестування | 1.03.2024 | 15.04.2024 | Виконано |
| 7 | Робота над розділами фахової частини БКР | 16.04.2024 | 31.04.2024 | Виконано |
| 8 | Проходження переддипломної практики, збір та аналіз матеріалів, остаточне оформлення розділів фахової частини БКР | 29.04.2024 | 12.05.2024 | Виконано |
| 9 | Розробка спеціальної частини з охорони праці | 13.05.2024 | 25.05.2024 | Виконано |
| 10 | Обговорення отриманих результатів з керівником та попередній захист КРБ | 27.05.2024 | 29.05.2024 | Виконано |
| 11 | Корегування роботи за результатами попереднього захисту | 30.05.2024 | 6.06.2024 | Виконано |
| 12 | Другий попередній захист КРБ | 10.06.2024 | 10.06.2024 | Виконано |
| 13 | Остаточне оформлення пояснювальної записки та слайдів доповіді до захисту | 11.06.2024 | 12.06.2024 | Виконано |
| 14 | Подання рецензенту та рецензування КРБ | 13.06.2024 | 13.06.2024 | Виконано |
| 15 | Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту | 17.06.2024 | 21.06.2024 | Виконано |
| 16 | Захист КРБ перед ЕК | 25.06.2024 | 25.06.2024 | Виконано |

Розробив студент _____Мурзакой Д. В._____ _____
*(прізвище та ініціали)* *(підпис)*

Керівник _____канд. пед. наук, доцент Болюбаш Н. М___ _____
*(наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

«_29_» _____01_____ 2024 р.

# АНОТАЦІЯ

**кваліфікаційної роботи бакалавра студента групи 401 ЧНУ ім. Петра Могили Мурзакоя Данила Васильовича**

**Тема: «Мобільний застосунок для визначення кредитоспроможності клієнтів банку»**


Кваліфікаційна робота бакалавра присвячена розробці та здійсненню програмної реалізації мобільного застосунку для визначення кредитоспроможності фізичних осіб – клієнтів банку. Це є актуальним, оскільки дослідження ефективних підходів до визначення кредитоспроможності клієнтів банку сприяє зменшенню ризиків у наданні кредитів в умовах високих темпів розвитку діджиталізації банківської сфери.

**Об'єкт роботи** – процес визначення кредитоспроможності клієнтів банку.

**Предмет роботи** – програмні засоби, методи та алгоритми визначення кредитоспроможності фізичних осіб – клієнтів банку.

**Мета роботи** – зменшення ризиків у наданні банківських кредитів шляхом розробки мобільного застосунку для визначення кредитоспроможності фізичних осіб на основі алгоритмів Data Mining.

Кваліфікаційна робота складається з фахової та спеціальної частини з охорони праці. Пояснювальна записка фахової частини складається зі вступу, трьох розділів, висновків та додатків. У першому розділі розкрито теоретичні засади визначення кредитоспроможності клієнтів банку. У другому розділі обґрунтовано вибір технологій і засобів розробки системи. У третьому розділі описано проєктування програмну реалізацію застосунку визначення кредитоспроможності клієнтів банку.

Кваліфікаційна робота бакалавра містить 73 сторінки (без додатків), 42 рисунки, 28 джерел та 1 додаток.

**Ключові слова**: кредитний ризик, кредитний скоринг, кредитний рейтинг, дерево рішень, ліс дерев, наївний байєсівський класифікатор.

# ABSTRACT

**for bachelor's qualification work**
**of a student of 401 group at Petro Mohyla Black Sea National University**
**Murzakoi Danylo**
**Theme: « Mobile application for determining the creditworthiness of bank clients »**

The bachelor's qualification work is devoted to the development and implementation of the software implementation of a mobile application for determining the creditworthiness of individuals - bank clients. This is relevant, because the study of effective approaches to determining the creditworthiness of bank clients contributes to reducing risks in granting loans in conditions of high rates of development of the digitalization of the banking sector.

**Object of work** – the process of determining the creditworthiness of bank clients.

**Subject of work** – software tools, methods and algorithms for determining the creditworthiness of individuals - bank clients.

**The purpose of this work** is to reducing risks in the provision of bank loans by developing a mobile application for determining the creditworthiness of individuals based on Data Mining algorithms.

The bachelor's qualification work consists of a professional and special part on labor protection. The explanatory note of the professional part consists of an introduction, three sections, conclusions and appendices. In the first chapter, the theoretical principles of determining the creditworthiness of bank clients are disclosed. In the second section, the choice of technologies and means of system development is substantiated. The third section describes the design and implementation of the application for determining the creditworthiness of bank clients.

The bachelor qualification work contains 73 pages (without appendices), 42 figures, 28 sources and 1 appendices.

**Keywords**: credit risk, credit scoring, credit rating, Decision Tree, Random Forest, Naive Bayes.

# ЗМІСТ

# ВСТУП

**Актуальність**. В умовах глобалізації та швидкого розвитку цифрових технологій, банки та фінансові установи стикаються з необхідністю оптимізації своїх процесів та покращення обслуговування клієнтів. Від швидкості та точності оцінки кредитоспроможності залежать не лише прибутки банку, але й його здатність мінімізувати ризики. Перспективним напрямком зменшення ризиків у наданні кредитів є розробка автоматизованих застосунків для визначення кредитоздатності клієнтів банку, що базуються на сучасних методах машинного навчання та алгоритмах Data Mining для надійного аналізу даних клієнтів при прийнятті кредитних рішень.

Стрімкий ріст банківського споживчого кредитування висуває високі вимоги до фінансової стійкості банків, одним із критичних факторів якого є вирішення проблеми неповернення банківських кредитів. Це обумовлює розвиток методів та підходів для визначення кредитоспроможності, які базуються на алгоритмах інтелектуального аналізу даних. Система банківського скорингу використовує різні моделі та спеціальні комп’ютерні програми, які дозволяють детально та швидко проаналізувати інформацію про позичальника, поставивши у відповідність кожному клієнту скоринг-бал, який автоматизує процес прийняття рішення щодо видачі кредиту. Впровадження мобільних рішень дозволяє банкам забезпечити зручність та доступність своїх послуг. Завдяки мобільним застосункам, клієнти можуть оперативно отримувати рішення щодо кредитних заявок, що значно скорочує час обробки та підвищує загальний рівень задоволеності клієнтів. Це особливо важливо в умовах зростаючої конкуренції на фінансовому ринку, де швидкість і якість обслуговування стають вирішальними факторами.

Це обумовило **мету роботи**, яка полягає у зменшенні ризиків у наданні банківських кредитів шляхом розробки мобільного застосунку для визначення кредитоспроможності фізичних осіб на основі алгоритмів Data Mining.

Відповідно до поставленої мети було сформульовано **завдання**:

– здійснити аналіз предметної сфери надання кредитів у банках та дослідити теоретичні засади виявлення кредитоспроможності клієнтів;

– обґрунтувати вибір технологій та засобів розробки мобільного застосунку для визначення кредитоспроможності фізичних осіб;

– розробка та здійснення програмної реалізації мобільного застосунку для визначення кредитоспроможності клієнтів банку.

**Об’єкт роботи** – процес визначення кредитоспроможності клієнтів банку.

**Предмет роботи** – програмні засоби, методи та алгоритми визначення кредитоспроможності фізичних осіб – клієнтів банку.

**Методологічною основою** дослідження є загальнонаукові та статистично-аналітичні методи, методи та алгоритми інтелектуального аналізу даних, які дозволили комплексно вивчити предмет та об’єкт дослідження, дослідити основні підходи до визначення кредитоздатності клієнтів банку.

**Структура кваліфікаційної роботи.** Відповідно до мети, завдань і предмета дослідження, бакалаврська робота містить основну та спеціальну частини. Основна частина роботи складається із вступу, трьох розділів, висновку, списку використаних джерел та __ додатків. Загальний обсяг роботи – __ сторінок, із них основного тексту основної частини – __ сторінок, спеціальної – __ сторінок. Кількість використаних джерел – __.

# 1 ТЕОРЕТИЧНІ ЗАСАДИ ВИЗНАЧЕННЯ КРЕДИТОСПРОМОЖНОСТІ КЛІЄНТІВ БАНКУ

## 1.1 Діяльність банків з надання кредитів

В умовах зростаючої конкуренції на банківському ринку для вітчизняних банків постає питання адаптації до нових умов та тенденцій ринку. Стрімкий розвиток сучасних технологій та збільшення інформаційних потоків, змушує українські банки запозичувати передовий досвід у всіх аспектах своєї діяльності. А збільшення прибутку від кредитних операцій безпосередньо пов’язане з якістю оцінки кредитного ризику.

Предметна сфера визначення кредитоспроможності клієнтів банку охоплює комплекс підходів, методів та інструментів, що застосовуються банківськими установами для аналізу та прогнозування можливостей потенційних позичальників своєчасно та в повному обсязі виконувати свої кредитні зобов'язання. Оцінка кредитоспроможності є ключовою діяльністю в роботі кожного банку, що дозволяє знижувати ризики неповернення кредитів і забезпечує стабільність фінансової системи в цілому.

Кредитоспроможність– це здатність організації або громадянина (юридична чи фізична особа) виплачувати позики відповідно до графіка та в повному обсязі [2]. Визначається на підставі безлічі факторів, у фізичних осіб та організацій вони різні. Аналіз багатьох аспектів у процесі визначення кредитоспроможності клієнтів банку є багатовимірним і комплексним, оскільки включає вивчення великої кількості змінних та факторів, які можуть впливати на здатність клієнта виконувати свої кредитні зобов'язання. До таких аспектів належать:

– історія попередніх позик: це включає аналіз історії платежів клієнта, наявність прострочень, раніше отримані та погашені кредити. Банки використовують цю інформацію для визначення надійності клієнта як позичальника. Історія попередніх позик часто впливає на кредитний рейтинг клієнта та може бути індикатором його поточної фінансової поведінки [3, 4];

– поведінка рахунків у банках: аналіз включає перевірку поточних балансів на рахунках, частоту та об'єми транзакцій, наявність і використання кредитних ліній та овердрафтів. Цей аналіз допомагає оцінити, наскільки раціонально клієнт управляє своїми фінансами, що є критичним фактором у визначенні його кредитоспроможності [3, 5];

– кредитні рейтинги: кредитний рейтинг клієнта, часто залежний від зовнішніх агенцій рейтингу, надає узагальнену оцінку кредитної історії та фінансової стабільності клієнта. Високий кредитний рейтинг може сприяти легшому доступу до більш вигідних кредитних умов [3, 6];

– особисті та демографічні дані клієнтів: ці дані включають вік, стать, освіту, сімейний стан, кількість дітей, професію, рівень доходу та інші соціально-економічні показники. Ця інформація допомагає банкам розуміти загальний життєвий контекст клієнта, його потенційну стабільність та спроможність підтримувати довгострокові фінансові зобов'язання [4].

Комбінований аналіз цих даних дозволяє банкам формувати глибоке та всебічне розуміння кредитного ризику, пов'язаного з кожним клієнтом. Сучасні технології, зокрема машинне навчання та великі дані, надають можливість автоматизувати частину цього аналізу, оптимізувати процеси прийняття рішень і забезпечити більш точне та ефективне кредитування.

З плином часу, у зв'язку з розвитком технологій, значно розширилися можливості аналізу даних, що дозволяє банкам використовувати більш складні і точні інструменти для оцінки кредитоспроможності. Наприклад, впровадження машинного навчання і штучного інтелекту в аналітичні системи дозволяє автоматизувати процеси збору та обробки інформації, а також забезпечує глибший і всебічний аналіз кредитного портфеля.

Також необхідно зазначити, що процес оцінки кредитоспроможності включає не тільки кількісний, але й якісний аналіз. Якісний аналіз може включати в себе розгляд таких факторів, як репутація клієнта, його соціальний статус та інші

некількісні параметри, що можуть вплинути на його здатність виконувати фінансові зобов'язання.

У контексті діджиталізації сучасного світу, роль мобільних технологій у фінансовій індустрії швидко зростає. Інтеграція методів оцінки кредитоспроможності в мобільні додатки відкриває нові можливості для банківських установ у підвищенні доступності та оперативності банківських послуг. Використання мобільних додатків для кредитного аналізу дозволяє клієнтам легко подавати заявки на кредити, отримувати швидкі рішення та управляти своїми кредитними лініями безпосередньо зі своїх смартфонів, що робить кредитний процес більш зручним та ефективним.

Така інтеграція також дозволяє банкам збирати та аналізувати велику кількість даних у реальному часі, що може включати геолокаційні дані, поведінкову інформацію та інші метрики, що збираються через мобільні пристрої. Це не тільки покращує точність кредитних оцінок, але й дозволяє банкам пропонувати персоналізовані фінансові продукти, відповідно до специфіки поведінки та потреб клієнтів.

Діджиталізація також сприяє більшій прозорості та взаємодії між клієнтами та банками. Мобільні додатки можуть надавати користувачам детальну інформацію про стан їхніх заявок, кредитний баланс та можливі наслідки для їхнього кредитного рейтингу в разі несплати. Це допомагає формувати більшу довіру та відкритість між банками та їхніми клієнтами, покращуючи загальне сприйняття кредитних продуктів і знижуючи фінансові ризики для обох сторін [5].

Загалом, інтеграція сучасних методів оцінки кредитоспроможності у мобільні додатки не тільки оптимізує роботу банків, але й відкриває нові горизонти для розвитку персоналізованих і інноваційних фінансових послуг, що відповідають вимогам сучасного цифрового суспільства.

Освітлюючи цю тему, можна зазначити, що предметна сфера визначення кредитоспроможності постійно розвивається, адаптується та вдосконалюється з урахуванням змін у економічному середовищі та технологічному прогресі, що

вимагає від фінансових інститутів постійного оновлення своїх методів і підходів для забезпечення ефективного управління ризиками та збільшення прибутковості.

Створення мобільного застосунку для визначення кредитоспроможності клієнтів, спеціалізованого на потреби працівників банку, може істотно вплинути на ефективність банківських операцій. Такий застосунок може не тільки прискорити процес оцінки кредитоспроможності, але й зменшити адміністративне навантаження на персонал, що дозволяє зосередити увагу на більш стратегічних задачах та підвищенні якості обслуговування клієнтів.

Перше та найважливіше, що пропонує мобільний додаток — це швидкість обробки даних. Використовуючи автоматизовані інструменти для збору та аналізу інформації про кредитоспроможність, банківські працівники можуть отримати потрібні висновки за лічені секунди. Це зменшує час, необхідний для розгляду кредитних заявок, і забезпечує можливість обробляти більшу кількість заявок протягом робочого дня. Крім того, автоматизація складних розрахунків і процедур оцінки знижує ймовірність людської помилки, що є критично важливим у фінансовій сфері. Це забезпечує більшу точність у прийнятті кредитних рішень і підвищує довіру клієнтів до банківських продуктів.

Інтеграція такого застосунку в повсякденні процедури банку також може допомогти знизити адміністративне навантаження на працівників. Завдяки мобільному доступу до системи оцінки кредитоспроможності, співробітники можуть ефективно керувати кредитними портфелями з будь-якої точки, де є доступ до інтернету, не обмежуючись робочим місцем у банку. Це особливо актуально для працівників, що працюють у польових умовах або віддалено. Наостанок, розробка мобільного додатку для працівників банку стимулює інноваційний підхід у всій організації, сприяючи культурі постійного покращення та використання сучасних технологій. Такий підхід не тільки підвищує ефективність роботи банку, але й забезпечує йому конкурентну перевагу на ринку.

## 1.2 Основні моделі та методи визначення кредитоспроможності позичальників

У сучасних економічних умовах більшість фізичних та юридичних осіб стикаються з проблемою нестачі власних коштів, тому залучення позикових коштів стає для них оптимальним виходом. Кредитоспроможність є основною умовою надання кредиту, саме вона показує фінансовий позичальника та характеризує його, як кредитоотримувача перед кредиторами. Надаючи позикові кошти, банки зацікавлені у своєчасному їх поверненні, тому вони приділяють велику увагу формуванню та розробці сучасних і точних методів оцінки кредитоспроможності [7, 8].

Кредитоспроможність являє собою комплексну характеристику, яку використовує для визначення доцільності взаємодії позичальника та банку під час проведення кредитної угоди.

Серед підходів до оцінки кредитоспроможності позичальників можна виділити експертну оцінку та рейтингові і скорингові алгоритми оцінки [7, 8].

*Експертна оцінка*. Фахівці займаються розрахунком показників позичальника, його особистісними якостями та характеристикою, аналізом його кредитної історії тощо. Виходить, по кожному, хто звернувся, приймається індивідуальне рішення, засноване не тільки на статистиці з його фінансів, а й на особистих враженнях від клієнта [8].

Існує загальна закономірність в оцінці результатів досліджень банками: чим вища сума позики та розмір щомісячної виплати, чим триваліший термін кредиту, тим ретельніше перевіряють громадянина на його здатність забезпечити погашення у строк. Також сучасні системи розрахунку кредитного ризику враховують макроекономічну ситуацію у світі тут і зараз. Наприклад, ризики дефолту, настання економічної кризи тощо. Для цього використовуються спеціальні комп'ютерні програми із вбудованими алгоритмами аналізу даних про клієнта [8].

*Рейтингова оцінка* дає змогу визначити фінансове становище клієнта за допомогою синтезованого показника – рейтингу, вираженого в балах, і віднести його до певного класу, що дає змогу зробити висновок про можливість надання кредиту. Загальний вид рейтингової оцінки є наступним:

$$K_0 = \sum_{i=1}^{n} A_i \cdot K_i , \tag{1.1}$$

де $K_0$ – інтегральний показник (рейтинг);

$A_i$ – вага $i$-го показника, $\sum_{i=1}^{n} A_i = 1$;

$K_i$ – значення $i$-го показника;

$n$ – число показників.

Використання фактурного та кількісного аналізу, що умовно ділить позичальників на три групи за рейтингом: «позитивних», «нейтральних» та «негативних», або дві – «позитивні» та «негативні». За допомогою алгоритмів вираховується показник, який надають кожному клієнту, від нього залежить потрапляння в ту чи іншу групу. Підсумки такого аналізу підбиваються в балах, у сумі показників і так далі, у рейтингу.

*Скорингові алгоритми оцінки*. Модифікацією рейтингової оцінки є кредитний скоринг (англ. credit scoring), відмінність якого полягає у тому, що у формулу рейтингової оцінки замість $K_i$ (значення $i$-го показника) підставляють $B_i$ – часткову бальну оцінку $i$-го показника . При цьому для кожного показника визначають декілька інтервалів значень, а кожному інтервалу значень приписують певну кількість балів або визначають клас $(1, 2, 3, \ldots)$. Якщо отриманий заємником рейтинг (кредитний скоринг є нижчим за установлений фахівцями, у кредиті буде відмовлено, якщо більшим – кредит буде надано. Перевагами рейтингової та скорингової моделей є їх простота – достатньо розрахувати фінансові коефіцієнти та визначити клас заємника.

*Кредитний скоринг* – це система оцінки кредитоспроможності клієнтів. У її основу закладені методи статистичного аналізу даних про клієнта. Іноді кількість параметрів оцінки клієнта може сягати 1000. За кожен параметр клієнту нараховують певну кількість балів, які потім сумують. Чим вищий підсумковий скоринг-бал, тим більш благонадійним вважається клієнт. Відповідність рівня щомісячних доходів та наявних активів, що приносять регулярні дивіденди, показнику фінансового навантаження, що виникає під час оформлення позики.

Перед тим, як прийняти рішення щодо заявки, банки перевіряють документи, що надаються заявником як підтвердження власної спроможності. З їхньою допомогою проводиться розрахунок ризиків — оцінюється ймовірність і розміру збитків, що виникає за умови, що клієнт у певний момент часу не матиме можливості вносити регулярні платежі. Алгоритм передбачає використання базових показників скорингу – сукупності даних, до яких належать вік, трудовий стаж, тривалість роботи на поточному місці, середньомісячний дохід та сімейний статус.

Застосовувані формули зрештою дають результат, який можна порівняти з мінімально допустимим пороговим значенням. Оцінка впливає не тільки на сам факт схвалення чи відмови, а й на запропоновані умови: чим ближче потенційний клієнт до прохідного мінімуму, тим вищою буде процентна ставка, покликана компенсувати можливі збитки у разі порушення договору. Додаткові показники, що також враховуються при розгляді заяв — кредитна історія, наявність заборгованостей перед бюджетом, стаж обслуговування в кредитній організації тощо. Використання автоматизованих систем помітно спростило аналітичний цикл, завдяки чому сьогодні відповідь за попередніми заявками можна отримати протягом декількох хвилин в тому числі - при зверненні через мобільний додаток.

Використання скорингової моделі як одного з головних інструментів ризик-менеджменту кредитних операцій є одним із найбільш ефективних. У сучасній банківській практиці при побудові скоринг-систем найчастіше враховуються такі

характеристики клієнта: кількість дітей, сімейний стан, дохід, наявність телефону, термін співробітництва з банком.

При здійсненні оцінки фінансового стану позичальника фізичної особи враховуються [9]:

– соціальна стабільність клієнта – наявність нерухомості, цінних паперів, роботи, сімейний стан: наявність реальної застави; вік та здоров'я клієнта; загальний матеріальний стан клієнта, його прибутки та витрати;

– користування банківськими позиками в минулому та своєчасність погашення їх та відсотків за ними, а також користування іншими банківськими послугами;

– зв'язки клієнта з діловим світом тощо.

Крім заробітної плати на кредитоспроможність впливають стать і сімейний стан (розлучені жінки більш дисципліновані, ніж неодружені чоловіки), вік (банки не довіряють дуже молодим та особам занадто похилого віку). Для розробки доброї скорингової карти обирають ряд факторів, які найбільше впливають на поведінку позичальника в майбутньому. З цією метою роблять вибірку «добрих» і «поганих» позичальників (у тому числі і відмови) за певний попередній період. Далі здійснюють аналіз усіх даних позичальника з використанням статистичних методів із метою виявлення тих даних, які є найбільш частими характеристиками окремо «добрих» і «поганих» кредитних рахунків позичальників. Такий аналіз також виявляє ступінь кореляції і важливості даних із якістю кредитного рахунку та їх важливість у скоринговій моделі. Результати, отримані після проведеного аналізу, формують скорингову карту.

Для побудови скорингових моделей застосовуються різні класифікаційні методи, зокрема: статистичні методи, що ґрунтуються на дискримінаційному аналізі, лінійне програмування, нейронні мережі, генетичні алгоритми, метод найближчих сусідів, ліс дерев та інші.

*Дерево рішень* (англ. Decision Tree) – є алгоритмом, який використовується для розв'язання задачі класифікації. Модель представляє рішення у формі

деревоподібної структури, де кожен внутрішній вузол відповідає ознаці (або атрибуту), кожна гілка представляє рішення про правило, а кожен лист вузла представляє кінцевий результат (клас або прогнозоване значення). Модель вивчає прості правила визначення атрибутів для прогнозування цільової змінної [10].

У моделі дерева рішень певні ознаки, такі як сума кредиту, вік позичальника, тривалість кредиту та кількість поточних кредитів, можуть сильніше впливати на фінальну оцінку. Наприклад, якщо банк надає перевагу оцінці загальної суми наданого кредиту та його тривалості, то і вплив цих параметрів буде більшим ніж інші. Такий підхід до побудови моделі повністю залежить від вимог конкретного банку і не враховує гнучкість для інших сценаріїв. Це означає, що модель дерева рішень дуже відрізняється від банку до банку, оскільки кожен банк може мати свої власні критерії та вимоги щодо оцінки кредитоспроможності. Наприклад, деякі банки можуть більше враховувати історію кредитування, тоді як інші можуть надавати перевагу оцінці поточного фінансового стану або стабільності зайнятості позичальника. Ця залежність від індивідуальних вимог робить модель менш універсальною і менш адаптованою до різних умов та змін у політиці кредитування. Проте дерево рішень добре підходить для реалізації моделі кредитного скорингу, оскільки кожен вузол дерева буде перевіряти умови надання кредиту і у відповідності з ними здійснювати нарахування балів, коефіцієнт важливості є основою для оцінки кредитоспроможності клієнта.

*Ліс дерев* (англ. Random Forest) – це ансамблевий метод машинного навчання, який використовує множину дерев рішень для досягнення кращої точності та стабільності прогнозу. Кожне дерево будується на випадковій підвибірці даних та ознак, забезпечуючи декореляцію між окремими деревами, що допомагає зменшити перенавчання. Під час класифікації кожне дерево голосує, а остаточний прогноз формується за найпопулярнішим голосом серед дерев, що забезпечує більш точні та надійні результати [11].

*Наївний байєсівський класифікатор* (англ. Naive Bayes) – це досить ефективний статистичний метод класифікації, який базується на теоремі Байєса з

припущенням про незалежність ознак. Наївний Байєс аналізує кожну ознаку окремо і приймає рішення, виходячи з ваги ознак, що дозволяє легко адаптуватися до змін та швидко оновлювати класифікатор. Цей клас моделей добре себе зарекомендував у задачах класифікації з великою кількістю ознак, особливо коли дані чітко розділені або коли кількість вибірок у категоріях значно різниться [12].

В основі моделі Naive Bayes лежить розрахунок апостеріорної ймовірності події $y_j = c_r$ за умови, що має місце подія $E_j$, яку можна розрахувати за формулою Байєса [12]:

$$P(y_j = c_r \mid E_j) = \frac{p(E_j \mid y_j = c_r) \cdot P(y_j = c_r)}{P(E_j)}, \quad (1.2)$$

де $P(y_j = c_r)$ – апріорна ймовірність віднесення об'єкту $i_j$ до класу $c_r$;

$E_j$ – подія, що відповідає рівності кожної з змінних, які характеризують об'єкт $i_j$, певним значенням;

$P(E_j)$ – ймовірність події $E_j$, рівна:

$$P(E_j) = \sum_{r=1}^{l} P(E_j \mid y = c_r) \cdot P(y = c_r), \quad (1.3)$$

де $l$ – кількість можливих значень залежної змінної – класів.

Відповідно до Байєсівського класифікатора новий об'єкт $i_h$ буде належати до того класу $c_r$, для якого його апостеріорна ймовірність $P(y_h = c_r \mid E_h)$, розрахована за формулою Байєса, є максимальною.

Оскільки у нас бінарна класифікація, оцінку точності моделей розраховують за формулою [14]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (1.4)$$

де *TP* (англ. True Positive): кількість об'єктів, правильно класифікованих – віднесених класифікатором до цього класу;

*TN* (англ. True Negative): кількість об'єктів, правильно класифікованих – не віднесених класифікатором до цього класу;

*FP* (англ. False Positive): кількість об'єктів, неправильно класифікованих – віднесених класифікатором до цього класу;

*FN* (англ. False Negative): кількість об'єктів, не правильно класифікованих – не віднесених класифікатором до цього класу.

Використання Gaussian Naive Bayes у даному проекті обрано через його здатність ефективно обробляти неперервні дані, які часто зустрічаються в фінансових наборах даних. Модель не потребує налаштування великої кількості параметрів, що робить її використання простим і зручним.

У ході створення застосунку для оцінки кредитоспроможності клієнту вирішено зупинитися на реалізації саме цих моделей – дерево рішень, ліс дерев на наївний байєсівський класифікатор. Моделі були обрані на основі їхньої здатності обробляти великі обсяги даних, забезпечення гнучкості у налаштуванні параметрів та спроможності ефективно класифікувати індивідуальні ризики.

### 1.3 Постановка задачі

Розробка мобільного застосунку для оцінки кредитоспроможності клієнтів банку на основі впровадження ефективних моделей аналізу даних про клієнтів банку з вбудованими алгоритмами Data Mining є критично важливими для успішної кредитної діяльності банків.

**Об'єкт роботи** – процес визначення кредитоспроможності клієнтів банку.

**Предмет роботи** – програмні засоби, методи та алгоритми визначення кредитоспроможності фізичних осіб – клієнтів банку.

**Мета роботи** – зменшення ризиків у наданні банківських кредитів шляхом розробки мобільного застосунку для визначення кредитоспроможності фізичних осіб на основі алгоритмів Data Mining.

Для досягнення поставленої мети було поставлено такі **завдання**:

1) здійснити аналіз предметної сфери надання кредитів у банках та дослідити теоретичні засади виявлення кредитоспроможності клієнтів;

2) обґрунтувати вибір технологій та засобів розробки мобільного застосунку для визначення кредитоспроможності фізичних осіб;

3) розробка та здійснення програмної реалізації мобільного застосунку для визначення кредитоспроможності клієнтів банку.

Розроблюваний мобільний застосунок для визначення кредитоспроможності повинен мати наступні функціональні можливості: аналізувати введені дані клієнта. Користувач застосунок (працівник банку) буде вводити у відповідні поля інформацію про дані клієнта (вік, стать, сума кредиту, тривалість займу, наявність інших кредитів). Отримані дані повинні оброблятися попередньо створеними та навченими моделями, та видавати результат відповідно до кожного з використаних методів оцінки кредитоспроможності. Ці вимоги є основою для подальшого проєктування та реалізації системи.

## Висновки до розділу 1

У даному розділі було розглянуто теоретичні засади визначення кредитоспроможності клієнтів банку. Основна увага була приділена аналізу предметної сфери, ролі діджиталізації у процесі оцінки кредитоспроможності, а також оптимізації банківських процесів з використанням сучасних технологій. Визначено, що кредитоспроможність клієнтів охоплює комплексну оцінку різноманітних аспектів, включаючи історію попередніх позик, поведінку рахунків, кредитні рейтинги та особистісні дані.

Було встановлено, що діджиталізація сприяє інтеграції процесів оцінки кредитоспроможності у мобільні додатки, що дозволяє покращити оперативність та доступність банківських послуг. Мобільні технології дозволяють збирати велику кількість даних у реальному часі, що підвищує точність кредитних оцінок і дає змогу банкам надавати персоналізовані фінансові послуги.

Також розглянуто важливість оптимізації банківських процесів, яка включає автоматизацію збору та аналізу даних, що зменшує людську помилку та підвищує ефективність банківських операцій. Розробка мобільних додатків для працівників банку сприяє більшій оперативності та ефективності прийняття кредитних рішень.

Цей розділ закладає основу для подальшого дослідження та розробки мобільного застосунку, який забезпечить точне та швидке визначення кредитоспроможності клієнтів, враховуючи сучасні тенденції та виклики в фінансовій індустрії.

# 2 ТЕХНОЛОГІЇ ТА ЗАСОБИ РОЗРОБКИ МОБІЛЬНОГО ЗАСТОСУНКУ

## 2.1 Середовище розробки Android Studio

Для створення програми для оцінки кредитоспроможності клієнтів на ОС Android необхідно вибрати відповідне середовище розробки та мову програмування (рис. 2.1, рис. 2.2). В наш час існує кілька популярних варіантів, таких як Android Studio, IntelliJ IDEA, Eclipse, AIDE і різні фреймворки у Visual Studio. Обравши Android Studio для розробки, отримуємо доступ до потужного та універсального інструменту, який допомагає створювати високоякісні додатки для Android. Його інтуїтивно зрозумілий інтерфейс, надійні функції та широкий набір інструментів роблять процес розробки мобільних додатків простішим, ніж коли-небудь.

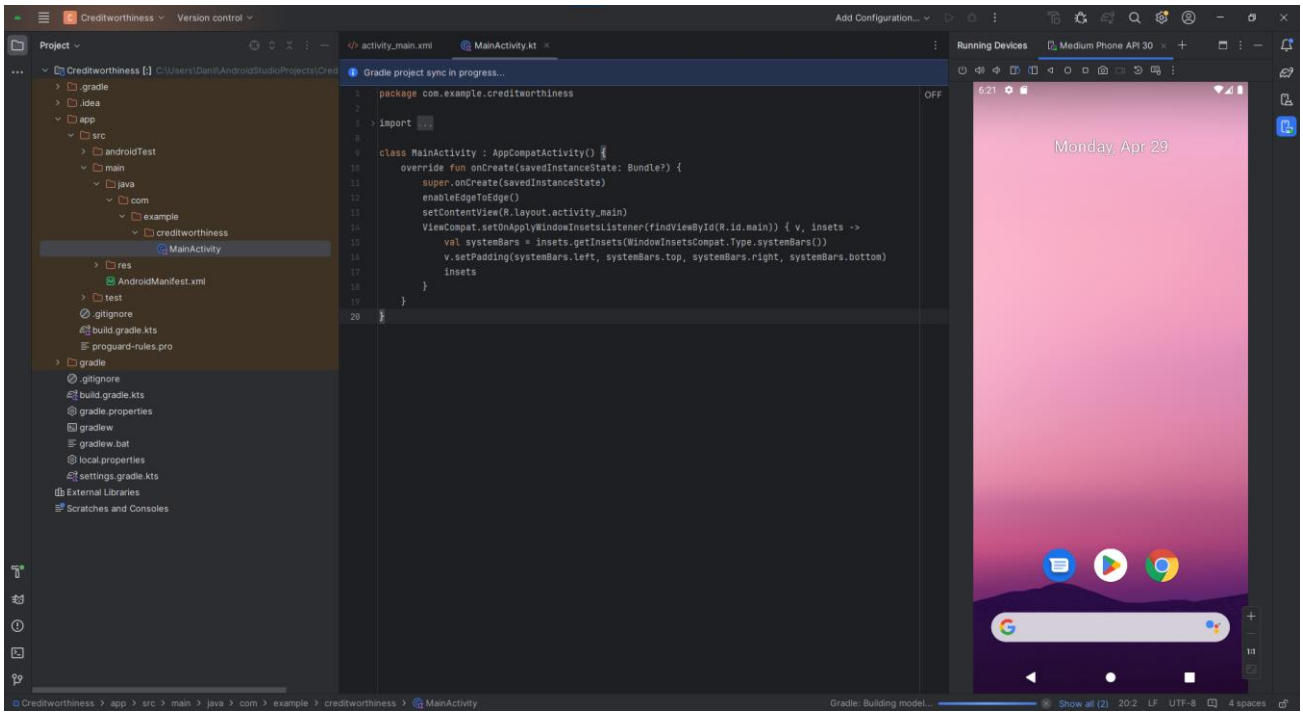



Рисунок 2.1 – Середовище розробки Android Studio

Android Studio — це офіційне інтегроване середовище розробки (IDE) для операційної системи Android від Google, створене на базі програмного забезпечення IntelliJ IDEA від JetBrains і спеціально призначене для розробки під

Android. Його можна завантажити для операційних систем Windows, macOS та Linux. Воно прийшло на зміну інструментам розробки Android в Eclipse (E-ADT) як основне середовище для розробки нативних додатків Android. Android Studio ліцензоване під Apache License, але включає деякі оновлення SDK, які мають невільну ліцензію, тому не є повністю відкритим кодом [13].

Рисунок 2.2 – Логотип середовища розробки Android Studio

Android Studio було анонсовано 16 травня 2013 року на конференції Google I/O. Початковий доступ до попереднього перегляду було відкрито з версії 0.1 у травні 2013 року, потім перейшло у бета-версію з версії 0.8, яка була випущена у червні 2014 року. Перша стабільна версія була випущена у грудні 2014 року, починаючи з версії 1.0. Наприкінці 2015 року Google припинила підтримку Eclipse ADT, зробивши Android Studio єдиним офіційно підтримуваним IDE для розробки додатків під Android [15].

Android Studio пропонує велику кількість функцій, які підвищують продуктивність під час розробки додатків для Android, наприклад [14]:

– гнучка система збірки на базі Gradle;

– швидкий та функціонально насичений емулятор;

– єдине середовище для розробки під всі пристрої Android;

– Live Edit для оновлення композитів в емуляторах та фізичних пристроях у реальному часі;

– шаблони коду та інтеграція з GitHub для створення поширених функцій додатків та імпорту прикладів коду;

– розширені інструменти та фреймворки для тестування;

– інструменти Lint для виявлення проблем із продуктивністю, зручністю використання, сумісністю версій та іншими проблемами;

– підтримка C++ та NDK;

– вбудована підтримка Google Cloud Platform, що спрощує інтеграцію Google Cloud Messaging та App Engine.

Перед розробкою мобільного додатку для визначення кредитоспроможності клієнтів банку на платформі Android, необхідно провести глибоке вивчення технічних аспектів, що включають в себе:

– вибір мови програмування та фреймворку: Аналіз різних мов програмування для мобільних додатків, таких як Java та Kotlin, та вибір оптимального фреймворку для розробки на платформі Android, такого як Android Studio;

– освоєння інструментів розробки: Ознайомлення з інтегрованою середою розробки Android Studio, а також іншими інструментами розробки, які допоможуть у створенні функціонального та ефективного мобільного додатку;

– оптимізація для мобільних пристроїв: Вивчення технік оптимізації додатків для мобільних пристроїв, таких як управління пам'яттю, енергоефективність, адаптивний дизайн тощо, для забезпечення високої продуктивності та зручного користування;

– інтеграція з базами даних: Вивчення методів підключення до баз даних на платформі Android, таких як SQLite з використанням фреймворку Room, та розробка структури баз даних для зберігання та обробки фінансових даних клієнтів.

– забезпечення безпеки даних: Вивчення методів захисту конфіденційності та безпеки даних в мобільних додатках, таких як шифрування даних, автентифікація користувачів, захист від вразливостей тощо.

Вивчення та реалізація цих технічних аспектів є важливим етапом перед розробкою мобільного застосунку для визначення кредитоспроможності клієнтів банку на платформі Android. Впевнене засвоєння цих знань дозволить створити

функціональний, ефективний та безпечний додаток, який задовольнить потреби користувачів та вимоги банку.

Переваги використання Android Studio очевидні. По-перше, з його допомогою можна швидко та легко розробляти додатки для Android, навіть без попереднього досвіду програмування. Крім того, витончений дизайн та інтуїтивно зрозумілий інтерфейс спрощують навігацію в процесі створення додатків, не потребуючи додаткового навчання чи допомоги фахівця з дизайну [14].

Android Studio має багато плюсів, зокрема простоту використання. Навіть розробники без значного досвіду можуть швидко освоїти його, слідуючи початковим інструкціям, а досвідчені фахівці оцінять гнучкість та розширюваність його функцій. Він включає вбудовану підтримку версій і розгалуження, комплексний інструмент аналізу коду та різноманітні рефакторинги та перевірки коду, що допомагають виявляти та виправляти помилки.

Почати роботу з Android Studio легко. Спочатку необхідно встановити його на комп'ютері, а потім слідувати початковим інструкціям для освоєння основ розробки Android. Після отримання базового розуміння мови програмування та основ розробки Android можна приступити до роботи над проєктом.

Обране середовище розробки підтримує роботу з двома мовами програмування для Android додатків: Java та Kotlin. Ці дві мови добре відомі в останні роки і користуються значною популярністю. Кожна з них має свої переваги і недоліки, які варто врахувати перед вибором. Java славиться своєю надійністю, у той час як Kotlin відомий своєю компактністю. Однак Java може бути повільнішою за новіші мови, такі як Python або JavaScript, а Kotlin страждає від обмеженої підтримки на різних платформах.

Java є однією з найпопулярніших і широко використовуваних мов програмування в світі (рис. 2.3). Вона має довгу історію, яка починається ще з 1995 року. Популярність Java пояснюється її широкою екосистемою бібліотек і інструментів, що робить її дуже універсальною. Ще однією причиною її популярності є продуктивність та надійність [15].

Рисунок 2.3 – Логотип мови програмування Java

Kotlin, з іншого боку, є новою мовою програмування, яка була створена у 2017 році (рис. 2.4). Вона призначена для того, щоб зробити програмування на Java більш компактним і зрозумілим для читача. Kotlin також має багатоплатформену сумісність і працює на платформах Java і Android. Крім того, він оптимізований для швидкості, що робить його привабливим вибором для розробки мобільних додатків. Для розробки застосунку було обрано Kotlin як мову програмування. Це зумовлено безліччю переваг [16]:

– компактність: Kotlin відомий своєю конкретністю, що робить його привабливим для багатьох розробників;

– надійність в роботі з нульовими значеннями: Kotlin робить код більш безпечним, уникаючи помилок з нульовими вказівниками;

– функції розширення: Kotlin дозволяє змінювати функціональність існуючих класів без їх успадкування, що полегшує розширення функцій програми;

– відкритий вихідний код: Kotlin є відкритим для розробників, що сприяє виявленню та виправленню помилок швидше;

– легкість вивчення: Багато вважають Kotlin досить простим для вивчення, що робить його привабливим для початківців.


Рисунок 2.4 – Логотип мови програмування Kotlin

Для побудови інтерфейсу програми буде використовуватися мова розмітки XML. Це дозволяє створювати легко читабельні файли, що мають простий текстовий формат і легко адаптуються до різних платформ (рис. 2.5).

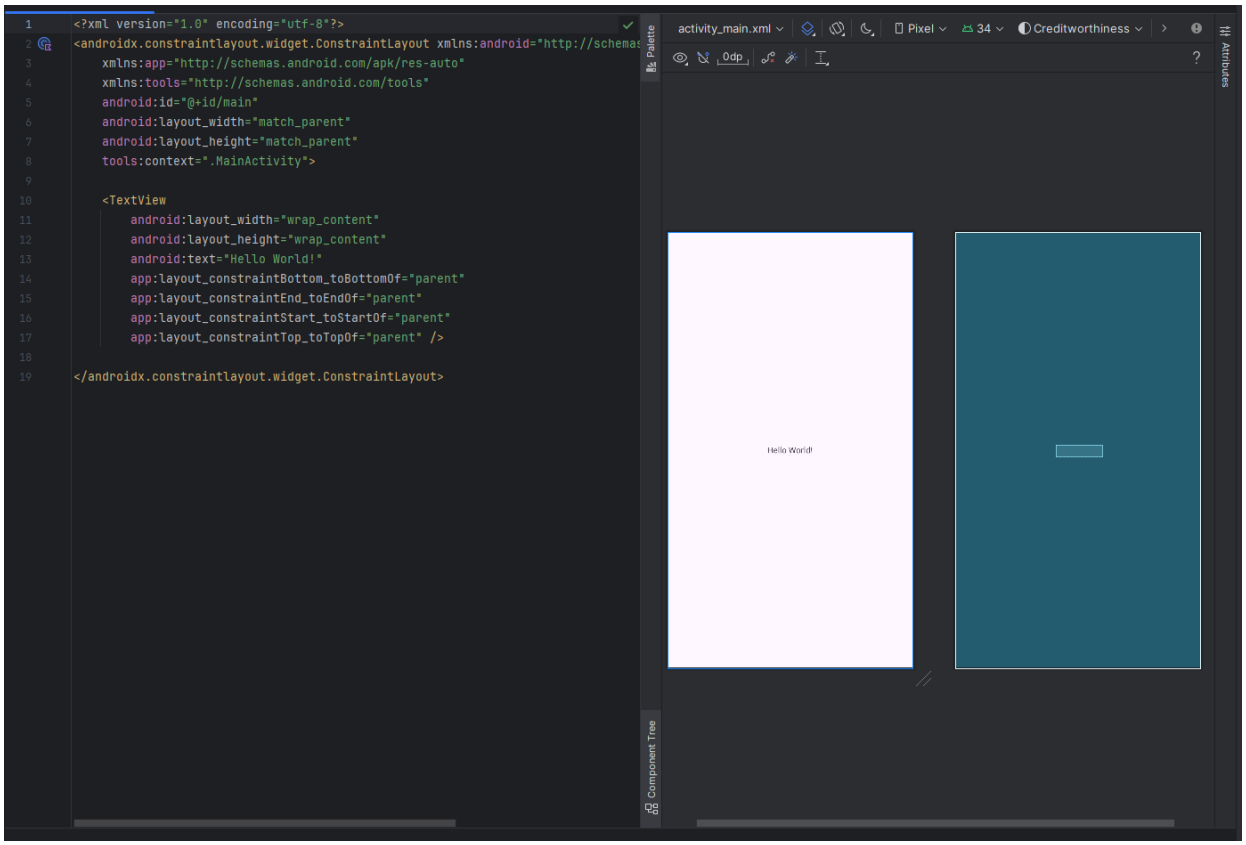



Рисунок 2.5 – Інтерфейс редагування екрану додатка використовуючи XML

Підсумовуючи, Android Studio виступає як ключовий інструмент для розробки мобільних додатків на платформі Android, що надає розробникам все необхідне для створення, тестування та оптимізації додатків. Це середовище розробки не тільки забезпечує широкі можливості для реалізації технічно складних проектів, але й відкриває простір для креативності та інновацій. Особлива увага до деталей, таких як інтеграція з мовами програмування Java та Kotlin, а також використання XML для розмітки інтерфейсу, підкреслює здатність Android Studio адаптуватися до вимог та вподобань розробників. Враховуючи ці переваги, Android Studio заслужено вважається оптимальним вибором для будь-якого проекту

розробки мобільних додатків, особливо коли мова йде про розробку функціональних і безпечних додатків для оцінки кредитоспроможності.

## 2.2 Середовище розробки PyCharm

PyCharm — це інтегроване середовище розробки (IDE) від компанії JetBrains, створене спеціально для мови програмування Python. PyCharm пропонує широкий спектр функціональності, який робить його популярним серед професіоналів та початківців у галузі Python-розробки [17].

PyCharm був випущений на ринок інтегрованих середовищ розробки (IDE), орієнтованих на Python, для конкуренції з такими продуктами, як PyDev (для Eclipse) або більш загально орієнтований Komodo IDE від ActiveState. Цей продукт був розроблений для того, щоб забезпечити розробників могутнім інструментом, який би інтегрував в себе всі необхідні можливості для ефективної роботи з Python, включно з редактором коду, інструментами для налагодження та автоматизації рутинних операцій [17].

22 жовтня 2013 року PyCharm став відкритим програмним забезпеченням. Варіант відкритого коду був випущений під назвою Community Edition, у той час як комерційна версія, Professional Edition, містить закриті модулі. Community Edition пропонує великий спектр функцій, таких як підтримка різноманітних фреймворків, інтеграція з системами контролю версій та багато іншого, що робить його привабливим вибором для студентів, хобістів та професійних розробників, які шукають доступне рішення. З іншого боку, Professional Edition включає розширені можливості, такі як підтримка веб-розробки, робота з базами даних та інші спеціалізовані інструменти для професійної роботи, що вимагають комерційної ліцензії [17].

Нижче більш детально охарактеризовано основні особливості PyCharm [18, 19].

1. Інтелектуальне редагування коду:

– автодоповнення коду: PyCharm аналізує контекст вашого коду та пропонує можливі варіанти доповнення;

– перевірка коду на льоту: IDE перевіряє код на наявність помилок і попереджає про них у реальному часі;

– рефакторинг: PyCharm надає потужні інструменти для реорганізації коду, включаючи перейменування, екстракцію методу, зміну сигнатури методу та багато іншого.

2. Підтримка веб-розробки:

– підтримка різноманітних фреймворків: Django, Flask, Pyramid та інші;

– робота з базами даних: PyCharm має вбудовані інструменти для роботи з SQL і базами даних, такими як MySQL, PostgreSQL, SQLite.

3. Віртуальні середовища: інтеграція з virtualenv, venv та Docker: PyCharm дозволяє легко управляти різними ізольованими середовищами для розробки;

4. Інтеграція з системами контролю версій: підтримка Git, Mercurial, SVN та інших: IDE надає інструменти для управління версіями та інтеграції з репозитаріями;

5. Налаштованість:

– плагіни: користувачі можуть розширювати функціональність PyCharm за допомогою плагінів, доступних через JetBrains Plugin Repository;

– теми: Можна налаштувати вигляд ідентифікатора відповідно до своїх переваг.

6. Вбудований термінал: користувачі можуть запускати команди shell безпосередньо з IDE, що полегшує роботу з системою та інструментами командного рядка.

7. Професійна і користувальницька версії:

– професійна версія пропонує додаткові функції, зокрема підтримку веб-фреймворків та роботу з базами даних;

– користувальницька версія є безкоштовною і включає базові можливості IDE для розробки на Python.

Charm забезпечує потужні інструменти для розробки на Python, що робить його одним з найпопулярніших виборів серед розробників (рис. 2.6).

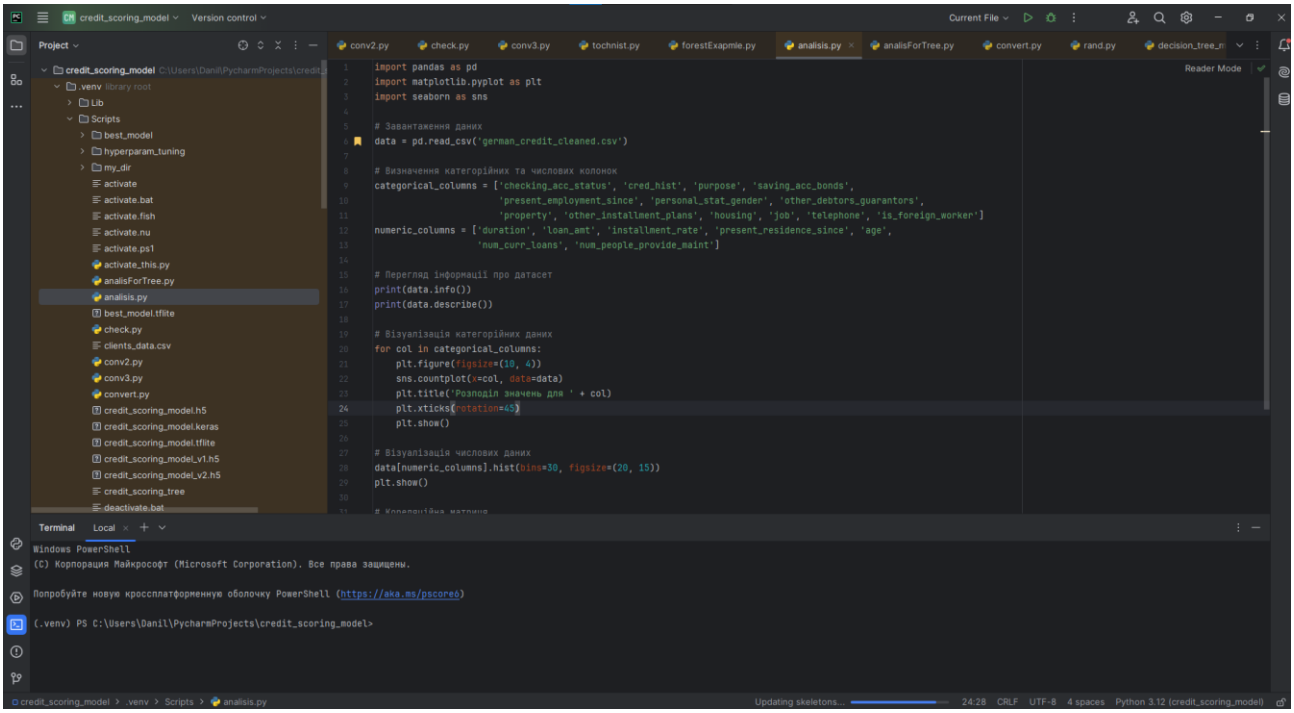



Рисунок 2.6 – Інтерфейс середовища розробки PyCharm

У підсумку, PyCharm становить собою виняткове інтегроване середовище розробки для Python, що забезпечує комплексний набір інструментів, які оптимізують процес розробки та підвищують ефективність програмування. Від високоякісного інтелектуального редагування коду до глибокої підтримки веб-розробки та роботи з базами даних, PyCharm відповідає потребам як новачків, так і досвідчених розробників. Вибір між користувальницькою та професійною версіями дозволяє користувачам адаптувати інструмент під власні потреби та бюджет, забезпечуючи гнучкість у виборі оптимальних можливостей для конкретного проєкту. Така багатофункціональність і налаштованість роблять

PyCharm ідеальним вибором для розробки на Python, посилюючи його позиції на ринку інтегрованих середовищ розробки.

### 2.3 Scikit-learn

Scikit-learn – це потужна бібліотека машинного навчання для мови програмування Python, яка використовується у широкому спектрі застосувань від наукових досліджень до промислових систем. Бібліотека забезпечує простий та зручний інтерфейс для виконання багатьох задач машинного навчання та обробки даних [20].

Основні характеристики Scikit-learn [21]:

– широкий спектр алгоритмів: Scikit-learn включає підтримку багатьох алгоритмів машинного навчання, включаючи класифікацію, регресію, кластеризацію та зниження розмірності. Крім того, бібліотека надає інструменти для модельного вибору та оцінки, обробки даних, а також вбудовані набори даних;

– підтримка обробки даних: Перед застосуванням моделей машинного навчання дані часто потрібно підготувати. Scikit-learn пропонує широкий набір інструментів для масштабування, перетворення та обробки даних, таких як нормалізація, кодування категорійних змінних та автоматичне видалення непотрібних даних;

– модульність: Scikit-learn розроблений з дотриманням принципів модульності. Це означає, що різні моделі та алгоритми можуть легко заміщуватися або налаштовуватися залежно від специфічних потреб проекту;

– спільнота і підтримка: Однією з ключових переваг Scikit-learn є активна спільнота розробників та користувачів, які постійно доповнюють бібліотеку новими алгоритмами та функціями. Багата документація, численні зразки коду допомагають швидко освоїти ці інструменти.

Основні алгоритми в Scikit-learn [21]:

– Random Forest: це ансамблевий метод, який використовує множину рішучих дерев для покращення точності та стабільності прогнозування. Random Forest добре справляється з великими наборами даних і може обробляти як категорійні, так і числові дані;

– Naive Bayes: це сімейство простих ймовірнісних класифікаторів, заснованих на застосуванні теореми Байєса з наївними припущеннями про незалежність між ознаками. Naive Bayes широко використовується для задач класифікації тексту та з його допомогою можна швидко робити прогнози на великих наборах даних;

– Support Vector Machines (SVM): це потужний класифікатор, який знаходить гіперплощину або набір гіперплощин у високо вимірному просторі, який може бути використаний для класифікації або регресії. SVM ефективний у випадках, де кількість ознак значно перевищує кількість зразків;

– K-Nearest Neighbors (KNN): цей алгоритм використовується для класифікації та регресії. Він працює шляхом знаходження K найближчих зразків до даної точки даних та використання їхніх властивостей для визначення виводу. KNN є дуже інтуїтивно зрозумілим та простим у використанні;

– Gradient Boosting Machines (GBM): це ансамблевий алгоритм, який побудований на засадах посилення, де нові моделі додаються для коригування помилок попередніх моделей. GBM часто використовується для підвищення продуктивності моделей машинного навчання завдяки своїй здатності оптимізувати різноманітні функції втрат;

– Principal Component Analysis (PCA): використовується для зниження розмірності даних шляхом перетворення великої кількості ознак у меншу кількість не корельованих ознак, званих головними компонентами. Цей метод допомагає у візуалізації даних, спрощенні моделей та зменшенні часу обчислень.

Ці алгоритми в Scikit-learn покривають широкий спектр можливих застосувань і допомагають аналітикам та розробникам ефективно розв'язувати

задачі аналізу даних. Scikit-learn забезпечує єдиний інтерфейс до цих алгоритмів, що робить перехід між ними гладким та зручним.

Прикладні сфери застосування Scikit-learn:

– фінансовий аналіз: використовується для кредитного скорингу, виявлення шахрайства та прогнозування фінансових ринків;

– біомедичні дослідження: для аналізу даних про здоров'я та передбачення медичних станів;

– маркетинговий аналіз: для сегментації клієнтів, аналізу ефективності реклами та прогнозування поведінки споживачів.

Scikit-learn залишається однією з найпопулярніших бібліотек для машинного навчання завдяки своїй гнучкості, потужності та доступності, роблячи її відмінним вибором для академічних досліджень та комерційних застосувань.

### 2.4 Бібліотеки Python: NumPy, Pandas, Chaquopy, Matplotlib

Використання різноманітних бібліотек Python стає ключовим для забезпечення аналізу та визначення кредитного скорингу з високою точністю та швидкістю. Охарактеризуємо найбільш важливі з них.

NumPy (скорочено від "Numerical Python") — це популярна бібліотека для роботи з числовими даними в мові програмування Python. Вона надає потужні інструменти для обробки та аналізу великих масивів даних, що робить її надзвичайно корисною для наукових обчислень, машинного навчання та аналізу даних . Основні можливості та переваги NumPy включають [22, 23]:

– масиви (ndarray): Основним об'єктом у NumPy є багатовимірний масив (ndarray), який дозволяє ефективно зберігати та обробляти великі обсяги даних. Масиви NumPy більш продуктивні та ефективні у використанні пам'яті, ніж стандартні списки Python;

– математичні операції: NumPy забезпечує широкий набір функцій для виконання математичних операцій на масивах, включаючи арифметичні операції,

лінійну алгебру, статистику та інші наукові обчислення. Всі операції оптимізовані для швидкого виконання;

– універсальні функції (ufuncs): NumPy містить багато універсальних функцій, які дозволяють виконувати елементні операції над масивами. Це включає такі функції, як синус, косинус, експоненціальна функція, логарифм та багато інших;

– інтеграція з іншими бібліотеками: NumPy добре інтегрується з іншими бібліотеками Python, такими як SciPy, pandas, matplotlib та іншими, що робить її основою для більш складних наукових і аналітичних пакетів;

– висока продуктивність: NumPy написана на мові C, що забезпечує високу продуктивність і дозволяє обробляти великі обсяги даних значно швидше, ніж за допомогою чистого Python;

– підтримка широкого спектру функцій: NumPy підтримує роботу з різними типами даних, включаючи цілі числа, числа з плаваючою комою, комплексні числа, а також надає можливість створення масивів із зазначеним типом даних.

NumPy є важливою частиною екосистеми Python для наукових обчислень і аналізу даних, тому її використання відкриває широкі можливості для роботи з даними та машинного навчання.

Pandas – це потужна і гнучка бібліотека для обробки та аналізу даних у Python. Вона спеціально розроблена для маніпулювання і аналізу структурованих даних, таких як таблиці або часові ряди. Pandas надає високопродуктивні, прості у використанні структури даних і інструменти для обробки числових таблиць та часових рядів. Основні можливості та переваги Pandas включають [24, 25]:

– DataFrame та Series: Основні структури даних у Pandas — це DataFrame та Series. DataFrame є двовимірною таблицею даних з індексами та стовпцями, а Series — одновимірним масивом даних з індексами;

– імпорт та експорт даних: Pandas дозволяє легко імпортувати та експортувати дані з різних джерел, включаючи CSV, Excel, SQL, JSON, HTML, HDF5 та інші формати;

– маніпуляція даними: Pandas надає широкий набір функцій для маніпуляції даними, включаючи фільтрацію, сортування, групування, злиття (merge) та з'єднання (join) таблиць, обробку відсутніх даних та багато іншого;

– часові ряди: Pandas має потужні інструменти для роботи з часовими рядами, включаючи генерацію діапазонів дат, зміщення (shifting), агрегування та ресемплінг (resampling) даних;

– операції з індексами: Pandas забезпечує гнучке індексування, що дозволяє легко отримувати доступ до підмножин даних, змінювати індекси та виконувати операції з багаторівневими (multi-index) даними;

– інтеграція з іншими бібліотеками: Pandas добре інтегрується з іншими бібліотеками Python для наукових обчислень, такими як NumPy, SciPy, matplotlib та інші;

– підтримка складних операцій: Pandas підтримує виконання складних операцій з даними, таких як розрахунок зведених таблиць (pivot tables), виконання віконних (rolling) та ковзних (expanding) операцій.

Бібліотека Pandas допомагає на всіх етапах підготовки даних: від завантаження даних до їх попередньої обробки, що включає фільтрацію, трансформацію та масштабування даних для подальшого використання в моделі машинного навчання.

Chaquopy – це плагін для Android Studio, який дозволяє розробникам інтегрувати код Python у додатки Android, написані на Java або Kotlin. Він забезпечує зручний спосіб використання Python-екосистеми, включно з бібліотеками та інструментами, без необхідності повного переписування існуючого коду Android. Ось основні особливості Chaquopy:

– інтеграція Python і Java/Kotlin: Chaquopy дозволяє викликати Python-код безпосередньо з Java або Kotlin та навпаки. Це дає змогу розробникам використовувати бібліотеки Python, такі як NumPy, Pandas, SciPy, а також машинне навчання та обробку даних;

– використання популярних Python-бібліотек: З Chaquopy можна легко інтегрувати велику кількість доступних Python-бібліотек. Розробники можуть додавати їх через pip, що значно розширює можливості Android-додатків;

– компіляція та виконання: Python-код, що використовується у додатку, компілюється та оптимізується під час збірки додатка, що забезпечує високу продуктивність на рівні з використанням Java або Kotlin;

– легкість використання: Chaquopy надає плагін для Android Studio, який спрощує налаштування та роботу з Python. Розробники можуть відслідковувати залежності та управляти ними через звичайний файл “build.gradle” Android проекту;

– ліцензія та підтримка: Chaquopy доступний як у вільній, так і у комерційній версії, залежно від потреб проекту. Розробники можуть вибрати платну версію для отримання додаткової підтримки та функцій.

Matplotlib — це популярна бібліотека для створення статичних, анімаційних та інтерактивних візуалізацій у Python. Вона забезпечує широкий спектр інструментів для побудови графіків, діаграм та інших візуальних представлень даних. Matplotlib є дуже гнучкою бібліотекою, яка дозволяє створювати як прості, так і складні візуалізації . Основні можливості та переваги Matplotlib включають [26, 27]:

– різноманітні типи графіків: Matplotlib підтримує створення різних типів графіків, таких як лінійні графіки, гістограми, стовпчикові діаграми, кругові діаграми, коробчаті графіки, діаграми розсіювання та багато інших;

– висока налаштовуваність: Користувачі можуть налаштовувати практично всі аспекти графіків, включаючи кольори, шрифти, мітки осей, легенди, лінії сітки та інші елементи;

– інтерактивні графіки: Використовуючи інструменти, такі як matplotlib.pyplot, користувачі можуть створювати інтерактивні графіки, які можна

масштабувати, панорамувати та зберігати в різних форматах (PNG, PDF, SVG тощо);

– анімації: Matplotlib дозволяє створювати анімації, які можуть бути корисними для демонстрації змін даних з часом;

– вбудована підтримка числових бібліотек: Matplotlib добре інтегрується з такими бібліотеками, як NumPy і Pandas, що робить її особливо зручною для візуалізації даних з цих джерел;

– компонування графіків: Користувачі можуть легко створювати складні компоновки з кількома підграфіками, що дозволяє порівнювати різні набори даних або аспекти даних на одному полотні.

Matplotlib є невід'ємною частиною інструментарію, оскільки вона дозволяє ефективно візуалізувати дані та робити їх зрозумілими для аналізу обробки даних.

### 2.5 Фреймворки Android SDK та Jetpack

Android SDK (Software Development Kit) – це набір інструментів і бібліотек, який використовується для розробки додатків для операційної системи Android [28]. SDK містить все необхідне для створення, налагодження, тестування та розгортання Android-додатків. Основні компоненти та можливості Android SDK.

Нижче опишемо більш детально основні компоненти Android SDK.

1. Комплект інструментів для розробки:

– компілятор: Використовується для компіляції коду (Java, Kotlin) у байт-код, який потім виконується на віртуальній машині Dalvik або ART (Android Runtime);

– інструменти командного рядка: Набір інструментів для роботи з проектами з командного рядка, включаючи adb (Android Debug Bridge), avdmanager (інструмент для управління віртуальними пристроями Android), sdkmanager та інші.

2. Android Emulator: віртуальний пристрій, який дозволяє тестувати додатки без необхідності використовувати фізичний пристрій. Емулятор підтримує різні конфігурації пристроїв і версії Android.

3. Інструменти налагодження та профілювання:

– Logcat: інструмент для перегляду системних логів, що допомагає виявляти і виправляти помилки в коді;

– Android Profiler: інструмент для моніторингу продуктивності додатків, включаючи використання CPU, пам'яті та мережевих ресурсів.

Основні можливості та функціональність Android SDK:

– розробка інтерфейсів користувача: XML Layouts: Використання XML для створення макетів інтерфейсів користувача; View і ViewGroup: Основні компоненти для побудови користувацького інтерфейсу, такі як TextView, Button, ImageView, LinearLayout, RelativeLayout та інші;

– робота з даними: SharedPreferences: Для зберігання простих даних у форматі ключ-значення; SharedPreferences: Для зберігання простих даних у форматі ключ-значення; SQLite: Легка база даних, вбудована в Android, для зберігання структурованих даних; Room: Частина Android Jetpack, яка забезпечує абстракцію над SQLite для спрощення роботи з базою даних;

– мережеві запити: HTTP бібліотеки: Використання бібліотек, таких як Retrofit або OkHttp, для виконання мережевих запитів і взаємодії з API;

– фонова робота: Services: Компоненти для виконання довготривалих операцій у фоновому режимі; WorkManager: Бібліотека для керування фоновими завданнями з гарантованим виконанням.

Android SDK є основним інструментом для подальшої розробки застосунку Android, забезпечуючи всі необхідні засоби використання створених моделей.

Jetpack — це набір бібліотек і інструментів від Google, розроблених для спрощення та покращення процесу розробки додатків для операційної системи Android. Jetpack об'єднує найкращі практики і забезпечує вирішення загальних проблем, з якими стикаються розробники під час створення додатків.

Використання Jetpack дозволяє зосередитися на логіці додатку, залишаючи на бібліотеки обробку стандартних завдань.

Нижче описамо більш детально основні компоненти Jetpack.

1. Foundation:

– AppCompat: Забезпечує сумісність з попередніми версіями Android;

– Android KTX: Набір Kotlin-розширень, які роблять код більш лаконічним та зручним для читання.

2. Architecture:

– DataBinding: Зв'язування UI-компонентів з джерелами даних у вашій програм;

– Lifecycles: Управління життєвим циклом активностей та фрагментів;

– LiveData: Реактивний компонент для спостереження за даними в межах певного життєвого циклу;

– Navigation: Бібліотека для навігації між фрагментами та активностями;

– Paging: Завантаження і відображення великих наборів даних по сторінках;

– Room: Легка ORM для SQLite бази даних;

– ViewModel: Компонент, який зберігає та керує UI-даними під час змін конфігурації.

3. Behavior:

– DownloadManager: Спрощує керування завантаженнями;

– Media & Playback: Інструменти для роботи з мультимедіа;

– Permissions: Спрощує роботу з запитами дозволів;

– Sharing: Інструменти для спільного використання контенту;

– Slices: Інтерактивні фрагменти UI для додатків.

4. UI:

– Animation & Transitions: Бібліотеки для анімацій та переходів;

– Emoji: Підтримка сучасних емодзі;

– Fragment: Розширення для роботи з фрагментами;

– Layout: Бібліотеки для управління макетами;

– Palette: Отримання кольорів з зображень;

– RecyclerView: Високопродуктивний компонент для відображення списків.

**Висновки до розділу 2**

У другому розділі було розглянуто ключові технології та середовища розробки, які є фундаментальними для успішного вирішення поставленої задачі розробки мобільного додатку для оцінки кредитоспроможності клієнтів. Значну увагу приділено Android Studio, як офіційному інтегрованому середовищу розробки для платформи Android, що забезпечує всі необхідні інструменти та бібліотеки для ефективної розробки мобільних застоунків. Окрім цього, вибір мови програмування Kotlin, як основної для розробки, зумовлений її сучасними можливостями, що забезпечують безпеку, компактність коду та високу продуктивність.

Поряд з мобільною розробкою, важливу роль відіграє використання PyCharm для розробки серверної частини на Python. Це середовище розробки інтегрує безліч функцій для ефективної роботи з кодом, включаючи інтелектуальне редагування, підтримку множини фреймворків і бібліотек, що робить його ідеальним вибором для складних завдань обробки даних та машинного навчання.

Програмна реалізація застосунку виконана на мові Kotlin у середовищі Android Studio, що забезпечило високу адаптивність і легкість використання рішення. Інтеграція з Python через бібліотеку Chaquopy дозволила використати наявні Python-сценарії та бібліотеки безпосередньо в мобільному додатку.

Окрім технічних аспектів, особливу увагу було приділено безпеці обробки даних, що важливо для дотримання стандартів конфіденційності у банківській сфері. Введення даних клієнтами через зручний і інтуїтивно зрозумілий інтерфейс,

а також швидка обробка та візуалізація результатів забезпечили високу ефективність застосунку.

У цілому, досліджені технології та середовища розробки формують міцну основу для подальшої роботи над проєктом, забезпечуючи необхідні інструменти та ресурси для ефективного рішення поставлених задач. Вибір правильних інструментів та технологій є ключовим для успішної реалізації проекту, дозволяючи забезпечити високу продуктивність, безпеку та зручність використання розроблюваних рішень.

# 3 РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ВИЗНАЧЕННЯ КРЕДИТОСПРОМОЖНОСТІ КЛІЄНТІВ БАНКУ

## 3.1 Характеристика та аналіз набору даних

Аналіз даних відіграє ключову роль у проєктуванні та розробці ефективних моделей машинного навчання, які здатні передбачати фінансову надійність клієнтів на основі їх кредитних історій та інших відповідних характеристик. У цьому параграфі представлено аналіз та опис даних, які використовуються при розробці Android-застосунку для оцінки кредитоспроможності клієнтів. Обраний датасет наданий професором Гофманом університету Стретклайд, який знаходиться у Шотландії. Цей набір даних класифікує людей, описаних за допомогою набору атрибутів, як осіб із хорошою або поганою кредитоспроможністю.

У цьому датасет представлено набір атрибутів, які дають змогу класифікувати людей за набором на хороших і поганих стосовно кредитних ризиків. Типи ознак є категоріальними та числовими, кількість ознак, за якими проводиться класифікація – 20, один цільовий атрибут – має значення: низька чи висока кредитоспроможність.

Охарактеризуємо нижче більш детально атрибути набору даних, який було використано для створення та навчання класифікаційних моделей.

1. Атрибут checking_acc_status (категоріальний): статус існуючого розрахункового рахунку:

- below_0: … < 0 DM;
- below_200: 0 <= … < 200 DM;
- above_200: … >= 200 DM;
- no_checking_acc: рахунок відсутній.

2. Атрибут duration (числовий): погоджена тривалість позики в місяцях.

3. Атрибут cred_hist (категоріальний): статус кредитної історії:

– no_loan_or_paid_duly_other: кредитів не бралось, або всі кредити сплачені належним чином;

– paid_duly_this_bank: всі кредити в цьому банку сплачені належним чином;

– curr_loans_paid_duly: поточні кредити сплачені належним чином до цього часу;

– delay_in_past: затримки у погашенні у минулому;

– risky_acc_or_curr_loan_other: критичний рахунок, або інші існуючі кредити.

4. Атрибут purpose (категоріальний): мета запиту позики:

– car_new: автомобіль (новий);
– car_used: автомобіль (б/у);
– furniture_equipment: меблі або обладнання;
– radio_tv: радіо або телебачення;
– domestic_appliance: побутова техніка;
– repairs: ремонти;
– education: освіта;
– retraining: перепідготовка;
– business: бізнес;
– others: інше.

5. Атрибут loan_amt (числовий): сума кредиту.

6. Атрибут saving_acc_bonds (категоріальний): рахунок заощаджень або облігації:

– below_100: … < 100 DM;
– below_500: 100 <= … < 500 DM;
– below_1000: 500 <= … < 1000 DM;
– above_1000: .. >= 1000 DM;
– unknown_no_saving_acc: невідомо або відсутній рахунок заощаджень.

7. Атрибут present_employment_since (категоріальний): поточне місце роботи та тривалість у роках:

– unemployed: безробітний;
– below_1y: … < 1 рік;
– below_4y: 1 <= … < 4 роки;
– below_7y: 4 <= … < 7 років;
– above_7y: .. >= 7 років.

8. Атрибут installment_rate (числовий): ставка внесків у відсотках від дохідного залишку.

9. Атрибут personal_stat_gender (категоріальний): особистий статус та стать:

– male_divorced_separated;
– female_divorced_separated_married;
– male_single;
– male_married_widowed;
– female_single.

10. Атрибут other_debtors_guarantors (категоріальний): інші позичальникиа або гаранти: співпозичальник, гарант, жоден.

11. Атрибут present_residence_since (числовий): теперішнє проживання з.

12. Атрибут property (категоріальний): власність:

– real_estate;
– life_insurance_or_agreements: якщо не нерухомість: договір про накопичення або страхування життя;
– car_or_other: якщо не інше: автомобіль або інше, не в атрибуті 6;
– unknown_or_no_property: невідомо або відсутня власність.

13. Атрибут age (числовий): вік у роках.

14. Атрибут other_installment_plans (категоріальний): інші плани погашення: банк, магазин, жоден.

15. Атрибут housing (категоріальний): житло: оренда, власне, безкоштовне.

16. Атрибут num_curr_loans (числовий): кількість існуючих кредитів у цьому банку.

17. Атрибут job (категоріальний): робота:

– unemployed_non_resident: безробітний або некваліфікований - не резидент.
– unskilled_resident: некваліфікований – резидент.
– skilled_official: кваліфікований працівник або службовець.
– management_or_self_emp: управління, самозайнятість висококваліфікований працівник або службовець

18. Атрибут num_people_provide_maint (числовий): кількість осіб, які мають надавати утримання.

19. Атрибут telephone (категоріальний): номер телефону.

20. Атрибут is_foreign_worker (категоріальний): вказує, чи є особа іноземним працівником.

21. Атрибут target (категоріальний): є цільовим, приймає 2 значення – good: добра кредитоспроможність, bad: погана кредитоспроможність.

Для кращого сприйняття та роботи з даними було проведено статистичний аналіз набору даних: побудовано гістограми числових та категоріальних змінних та кореляційну матрицю числових змінних (рис. 3.1).

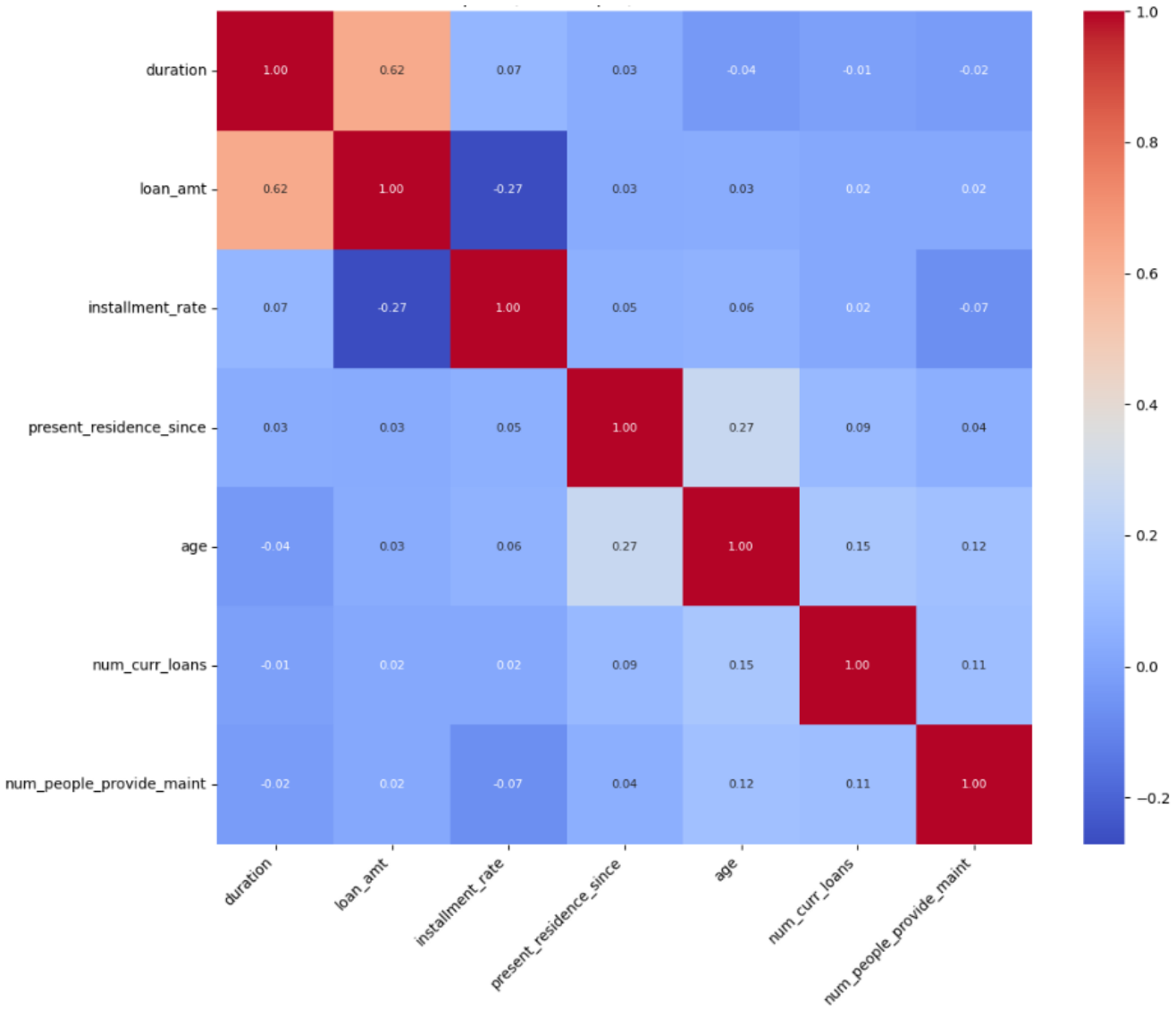



Рисунок 3.1 – Кореляційна матриця числових змінних

Кореляційна матриця допомагає ідентифікувати ключові взаємозв’язки між числовими змінними у наборі даних, які можуть впливати на рішення про кредитування. Розуміння цих залежностей є критично важливим для побудови ефективних моделей кредитного скорингу, оскільки воно дозволяє зосередитися на найбільш значимих атрибутах та зменшити ризик помилок у оцінці кредитоспроможності.

Кореляційний аналіз показав наявність значущих взаємозв'язків між деякими з ключових атрибутів, що може допомогти у вдосконаленні алгоритмів машинного навчання для більш точного прогнозування кредитної оцінки.

На рисунку 3.2 представлено гістограми числових змінних набору даних. на рисунках 3.3 та 3.4 – гістограми категоріальних змінних набору даних.

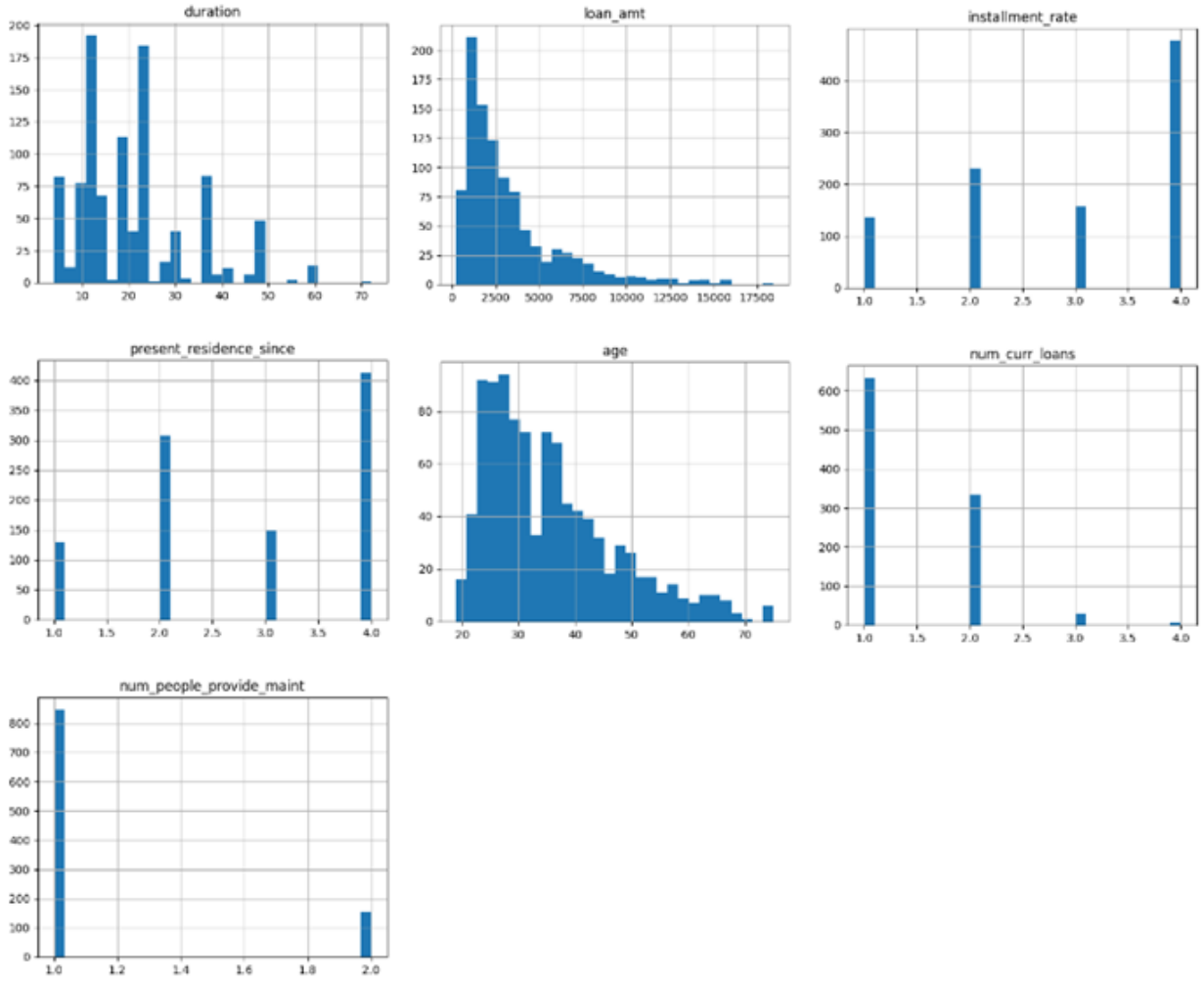



Рисунок 3.2 – Гістограми числових змінних

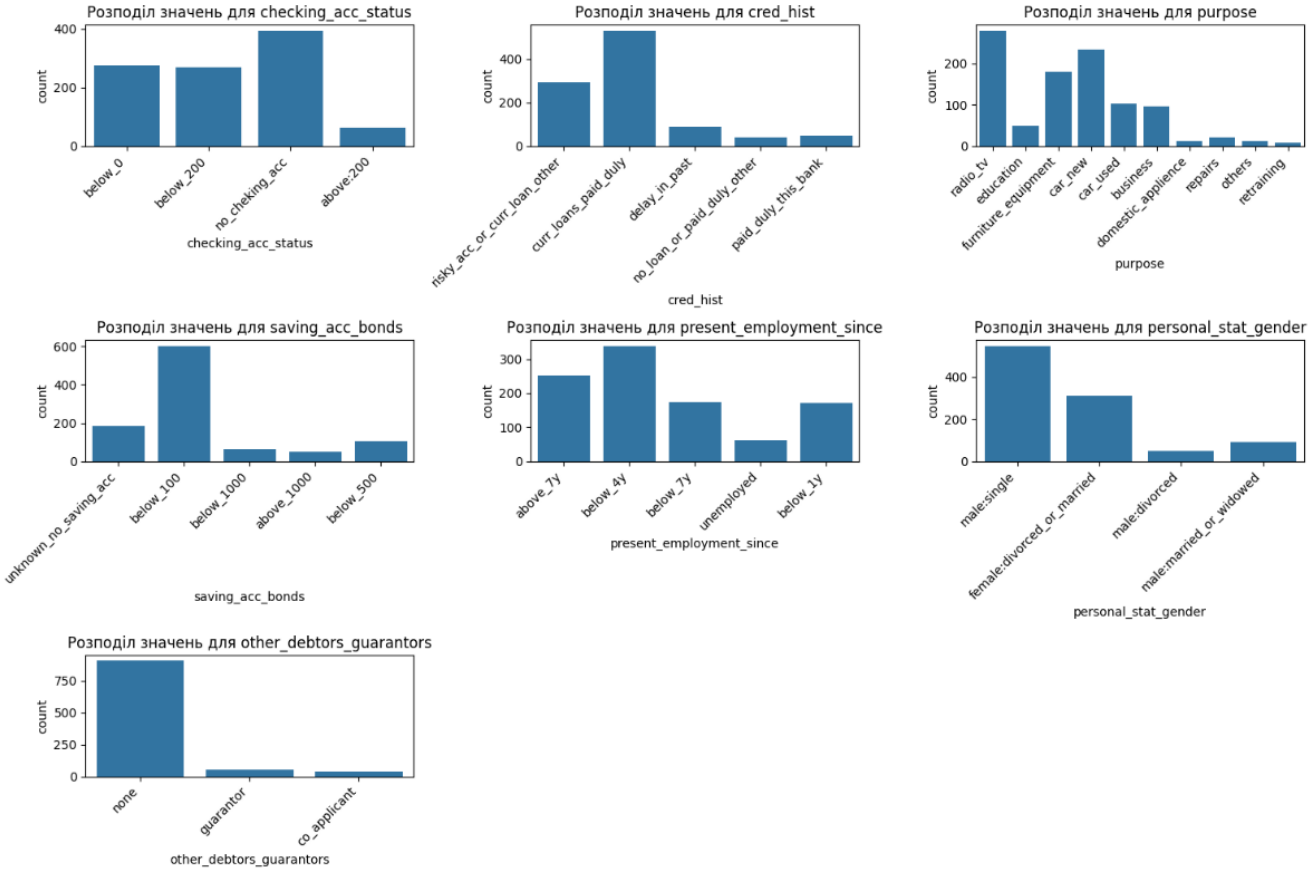



Рисунок 3.3 – Гістограми категоріальних змінних (частина 1)

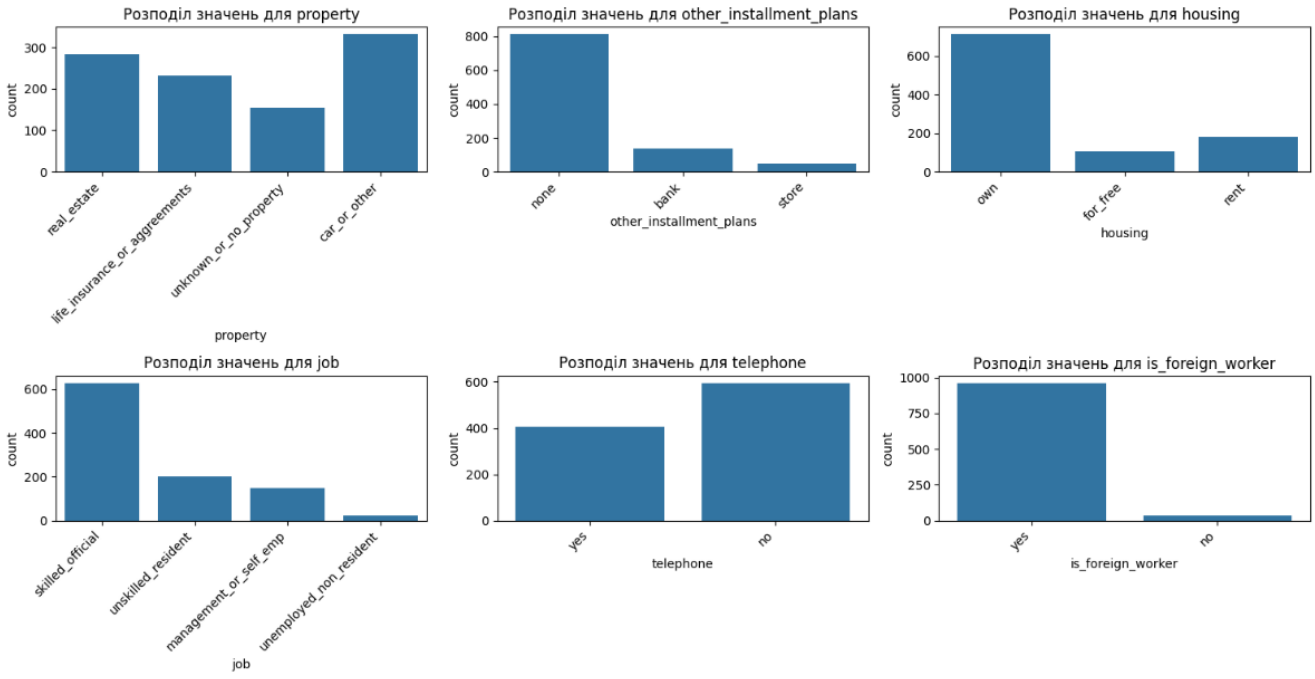



Рисунок 3.4 – Гістограми категоріальних змінних (частина 2)

Проведений аналіз набору даних підтвердив його придатність для подальшого використання у розробці андроїд-застосунку для оцінки кредитоспроможності клієнтів. Зібрані дані добре структуровані та містять різноманітні категоріальні та числові атрибути, які дозволяють виконати комплексне моделювання кредитних ризиків. Гістограми допомогли візуалізувати розподіл даних по категоріальних та числових змінних, що є важливим для розуміння структури даних і забезпечення балансу класів в навчальному процесі.

Здійснений аналіз свідчить, що датасет містить всю необхідну інформацію для виявлення факторів, які впливають на кредитоспроможність, та розробки надійних моделей для її оцінки.

### 3.2 Створення та навчання моделей

У застосунку реалізовано три ключові моделі класифікації, які використовуються для оцінки кредитоспроможності клієнтів в андроїд застосунку. Це моделі Random Forest, Naive Bayes та Decision Tree. Модель Decision Tree було розроблено для реалізації моделі кредитного скорингу. Моделі Random Forest і Naive Bayes було навчено з використанням датасет, описаного у попередньому параграфі. Кожна з цих моделей має свої унікальні характеристики і підходи до вирішення задачі класифікації, які дозволяють з різних ракурсів аналізувати поведінку позичальників. Важливим аспектом є також оцінка точності та перевірка надмірного навчання моделей, що забезпечує високий рівень довіри до результатів прогнозування.

1. ***Дерево рішень*** (англ. Decision Tree) є досить розповсюдженою моделлю машинного навчання для класифікаційних задач. Для підвищення ефективності моделі та зниження навантаження на процес прийняття рішень, модель дерева рішень використовує фільтрацію очевидно негативних результатів на початкових етапах (рис. 3.5). Це означає, що деякі параметри, які явно не відповідають критеріям кредитоспроможності, відразу ж призводять до відмови.

У моделі дерева рішень деякі параметри мають більші вагові коефіцієнти ніж інші, залежно від вимог конкретного банку. Це означає, що певні ознаки, такі як сума кредиту, вік позичальника, тривалість кредиту та кількість поточних кредитів, можуть сильніше впливати на фінальну оцінку.

```
if (loanAmt > 10000) return "Відмова: занадто великий розмір кредиту"
if (age < 18 || age > 80) return "Відмова: невідповідний вік"
if (duration > 60) return "Відмова: занадто довга тривалість кредиту"
if (numCurrLoans > 3) return "Відмова: занадто багато поточних кредитів"
```

Рисунок 3.5 – Відсіювання очевидно негативних результатів

На рисунку 3.6 наведено приклад, де використовуються прості правила на основі умов щодо основних атрибутів клієнта, таких як сума кредиту, вік, тривалість кредиту та кількість поточних кредитів.

```
when {
    loanAmt > 5000 -> score -= 200
    loanAmt > 3000 -> score -= 100
    loanAmt < 1000 -> score += 100
    loanAmt < 500 -> score += 200
}

when {
    duration > 36 -> score -= 100
    duration > 24 -> score -= 50
    duration < 12 -> score += 50
    duration < 6 -> score += 100
}

when {
    age < 25 -> score -= 50
    age > 60 -> score -= 20
    age in 30 ≤ .. ≤ 50 -> score += 30
}
```

Рисунок 3.6 – Різний вплив на оцінку кредитного рейтингу в залежності від вимог до параметрів

Додаткові умови включають такі параметри, як статус розрахункового рахунку, історія кредитування, мета позики, рахунок заощаджень, поточне місце роботи, особистий статус та стать, поручителі, наявність нерухомості, інші плани погашення, тип житла, кількість осіб на утриманні, наявність телефону та статус іноземного працівника.

Опишемо основні показники та бальні оцінки моделі кредитного скорингу, для реалізації якої було застосовано алгоритм дерева рішень, здійснивши покроковий опис роботи алгоритму.

1. Ініціалізація: витягуються значення характеристик клієнта: сума кредиту (loanAmt), вік (age), тривалість кредиту (duration), кількість поточних кредитів (numCurrLoans).

2. Перевірка базових умов, які можуть одразу призвести до відмови:

– сума кредиту більше 10000: «Відмова: занадто великий розмір кредиту»;

– вік менше 18 або більше 80 років: «Відмова: невідповідний вік»;

– тривалість кредиту більше 60 місяців: «Відмова: занадто довга тривалість кредиту»;

– кількість поточних кредитів більше 3: «Відмова: занадто багато поточних кредитів».

3. Ініціалізація початкового кредитного рейтингу: початковий рейтинг встановлюється на 500.

4. Вплив суми кредиту на рейтинг:

– якщо сума кредиту більше 5000, рейтинг зменшується на 200;

– якщо сума кредиту більше 3000, але менше або дорівнює 5000, рейтинг зменшується на 100;

– якщо сума кредиту менше 1000, рейтинг збільшується на 100;

– якщо сума кредиту менше 500, рейтинг збільшується на 200.

4. Вплив тривалості кредиту на рейтинг:

– якщо тривалість кредиту більше 36 місяців, рейтинг зменшується на 100;

– якщо тривалість кредиту більше 24 місяців, але менше або дорівнює 36, рейтинг зменшується на 50;

– якщо тривалість кредиту менше 12 місяців, рейтинг збільшується на 50;

– якщо тривалість кредиту менше 6 місяців, рейтинг збільшується на 100.

5. Вплив віку клієнта на рейтинг:

– якщо вік менше 25 років, рейтинг зменшується на 50;

– якщо вік більше 60 років, рейтинг зменшується на 20;

– якщо вік в діапазоні від 30 до 50 років включно, рейтинг збільшується на 30.

6. Вплив статусу рахунку на рейтинг:

– «нижче 0» – рейтинг зменшується на 100;

– «нижче 200» – рейтинг зменшується на 50;

– «без рахунку» – рейтинг зменшується на 20;

– «понад 200» – рейтинг збільшується на 50.

7. Вплив кредитної історії на рейтинг:

– «ризиковий рахунок або інший кредит» – рейтинг зменшується на 100;

– «затримки в минулому» – рейтинг зменшується на 50;

– «поточні кредити виплачуються вчасно» – рейтинг збільшується на 20;

– «немає кредиту або інший кредит виплачується вчасно» – рейтинг збільшується на 50;

– «кредит виплачується вчасно в цьому банку» - рейтинг збільшується на 100.

8. Вплив мети кредиту на рейтинг:

– «радіо/телевізор», «побутова техніка» – рейтинг зменшується на 30;

– «новий автомобіль», «вживаний автомобіль», «меблі/обладнання» – рейтинг збільшується на 20;

– «освіта», «бізнес» – рейтинг збільшується на 50.

9. Вплив заощаджень на рейтинг:

– «невідомо/немає заощаджень» – рейтинг зменшується на 50;

– «нижче 100» – рейтинг зменшується на 20;
– «нижче 1000» – рейтинг збільшується на 20;
– «понад 1000» – рейтинг збільшується на 50.

10. Вплив часу поточного працевлаштування на рейтинг:
– «безробітний» – рейтинг зменшується на 50;
– «нижче 1 року» – рейтинг зменшується на 50;
– «нижче 4 років» – рейтинг зменшується на 20;
– «нижче 7 років» – рейтинг збільшується на 20;
– «понад 7 років» – рейтинг збільшується на 100.

11. Вплив особистого статусу та статі на рейтинг:
– «жінка: не одружена» – рейтинг зменшується на 20;
– «жінка: одружена» – рейтинг збільшується на 20;
– «чоловік: не одружений» – рейтинг зменшується на 20;
– «чоловік: одружений» – рейтинг збільшується на 20.

12. Вплив інших позичальників на рейтинг:
– «поручитель» – рейтинг збільшується на 20;
– «співпозичальник» – рейтинг зменшується на 20;

13. Вплив наявності майна на рейтинг:
– «нерухомість» – рейтинг збільшується на 50;
– «автомобіль або інше» – рейтинг збільшується на 20.

14. Вплив інших кредитних планів на рейтинг:
– «банк» – рейтинг зменшується на 20;
– «магазин» – рейтинг зменшується на 10.

15. Вплив типу житла на рейтинг:
– «оренда» – рейтинг зменшується на 20;
– «власне житло» – рейтинг збільшується на 20.

16. Вплив кількості поточних кредитів на рейтинг:
– кількість поточних кредитів більше 2 – рейтинг зменшується на 30;
– один поточний кредит – рейтинг збільшується на 20.

17. Вплив зайнятості на рейтинг:

– «безробітний нерезидент» – рейтинг зменшується на 100;

– «некваліфікований резидент» – рейтинг зменшується на 50;

– «кваліфікований службовець» – рейтинг збільшується на 50;

– «управління або самозайнятість» – рейтинг збільшується на 50.

18. Вплив кількості осіб, яких клієнт утримує, на рейтинг: якщо кількість більше 1, рейтинг зменшується на 20.

19. Вплив наявності телефону на рейтинг: «так» – рейтинг збільшується на 20.

20. Вплив статусу іноземного працівника на рейтинг: «так» – рейтинг зменшується на 50.

21. Обмеження діапазону рейтингу:

– якщо рейтинг більше 1000, він встановлюється на 1000;

– якщо рейтинг менше 1, він встановлюється на 1.

22. Повернення рейтингу: рейтинг конвертується в строковий формат і повертається.

Модель використовує ці умови для розрахунку кінцевого результату, який визначає кредитоспроможність клієнта. Незважаючи на простоту та прямолінійність, такий підхід може бути ефективним для початкових етапів оцінки кредитоспроможності, але потребує подальшої оптимізації та адаптації до специфічних вимог кожного банку.

Дерево рішення, побудоване для реалізації цієї моделі, має велике розгалуження та глибину. Через велику кількість параметрів повністю зобразити дерево складно, корінь дерева та початкові розгалуження зображено на рисунку 3.7.

2. ***Random Forest*** – це модель, яка включає велику кількість дерев рішень, кожне з яких тренується на випадковій підвибірці ознак та даних. Використання багатьох дерев дозволяє зменшити ризик перенавчання та підвищити точність прогнозування порівняно з одиночним деревом рішень. Модель видає прогноз шляхом агрегації відповідей від усіх дерев (голосуванням для класифікації).

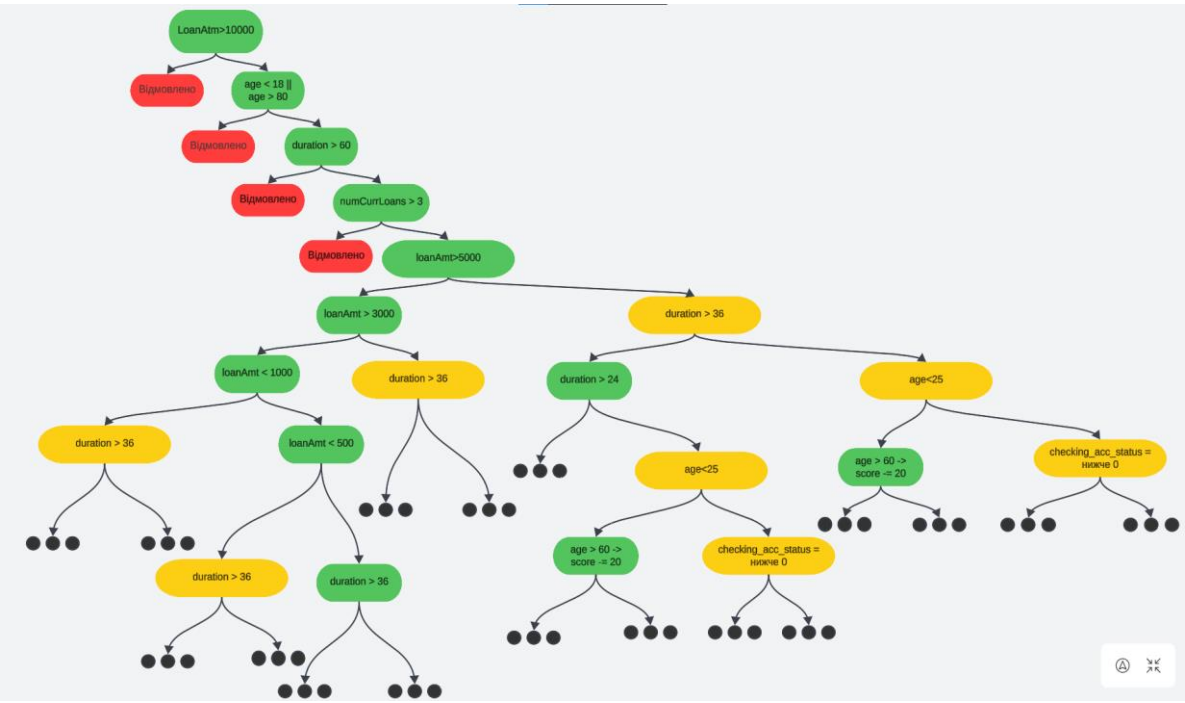



Рисунок 3.7 – Корінь та початкові розгалуження дерева рішень

Модель Random Forest була навчена з використанням 100 дерев, що є стандартним вибором для балансу між продуктивністю та точністю. Використання n_estimators = 100 забезпечує достатню кількість дерев для ефективного голосування при класифікації. Інший важливий параметр — random_state = 42 у початковому RandomForest та random_state = 42 у фінальній моделі, що забезпечує відтворюваність результатів (рис. 3.8).

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_selected = RandomForestClassifier(random_state=42)
```

Рисунок 3.8 – Налаштування моделі Random Forest

Перед тренуванням моделі дані були підготовлені та масштабовані. Використовувалася техніка SMOTE для балансування класів, забезпечуючи однакову представленість категорій good та bad у тренувальному наборі. Для вибору найбільш значущих ознак використовувалась модель Random Forest з

подальшим застосуванням SelectFromModel, що дозволило сконцентрувати тренування на найбільш інформативних атрибутах (рис. 3.9, рис. 3.10).

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
from sklearn.feature_selection import SelectFromModel

data = pd.read_csv('german_credit_cleaned.csv')
categorical_columns = ['checking_acc_status', 'cred_hist', 'purpose', 'saving_acc_bonds',
                       'present_employment_since', 'personal_stat_gender', 'other_debtors_guarantors',
                       'property', 'other_installment_plans', 'housing', 'job', 'telephone',
                       'is_foreign_worker', 'target']
label_encoders = {}
for column in categorical_columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le

joblib.dump(label_encoders, filename: 'label_encoders.joblib')

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

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(*arrays: X_resampled, y_resampled, test_size=0.3, random_state=42)
```

Рисунок 3.9 – Вміст файлу тренування та виводу результатів моделі Random Forest (частина 1)

Для кращого розуміння, які атрибути найбільше впливають на рішення моделі, була проведена візуалізація важливості ознак. Це дозволило ідентифікувати ключові параметри, які використовувались моделлю для класифікації кредитних ризиків.

На рисунку 3.11 представлено важливість кожної ознаки для моделі Random Forest. Ми бачимо, що найбільший внесок у прогнозування роблять наступні ознаки: checking_acc_status, loan_amt, duration, cred_hist, age. Цей графік допомагає зрозуміти, які ознаки найбільше впливають на рішення моделі, і може бути використаний для подальшої оптимізації та інтерпретації моделі.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

joblib.dump(scaler, filename: 'scaler.joblib')

rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train_scaled, y_train)

selector = SelectFromModel(rf, prefit=True)
X_train_selected = selector.transform(X_train_scaled)
X_test_selected = selector.transform(X_test_scaled)

joblib.dump(selector, filename: 'selector.joblib')

rf_selected = RandomForestClassifier(random_state=42)
rf_selected.fit(X_train_selected, y_train)
y_pred_rf = rf_selected.predict(X_test_selected)
print('Accuracy for RandomForest:', accuracy_score(y_test, y_pred_rf))

joblib.dump(rf_selected, filename: 'random_forest_model.joblib')

feature_importances = rf.feature_importances_
sns.barplot(x=feature_importances, y=X.columns)
plt.title('Feature Importances for RandomForest')
plt.show()

cm_rf = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm_rf, annot=True, fmt='d')
plt.title('Confusion Matrix for RandomForest')
plt.show()
```

Рисунок 3.10 – Вміст файлу тренування та виводу результатів моделі Random Forest (частина 2)

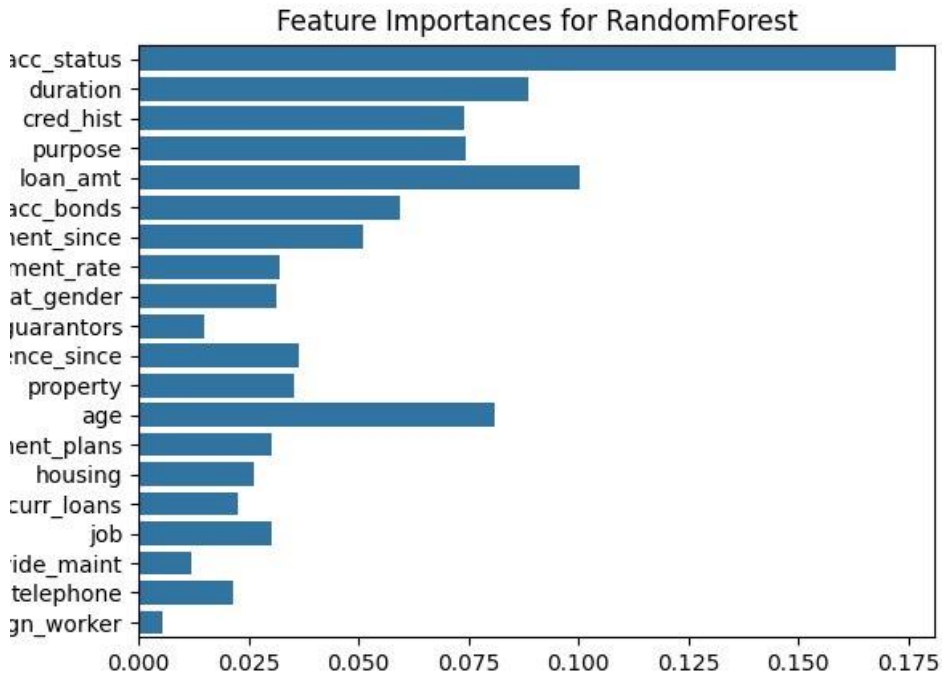



Рисунок 3.11 – Діаграма важливості ознак для Random Forest

На тепловій карті Confusion Matrix for RandomForest відображено результати класифікації моделі Random Forest (рис. 3.12). В Confusion Matrix – матриці помилок, представлені наступні дані:

– кількість правильних класифікацій для класу 0 становить TP =181;
– кількість неправильних класифікацій для класу 0 становить FN =14;
– кількість правильних класифікацій для класу 1 становить TN =184;
– кількість неправильних класифікацій для класу 1 становить FP =24.

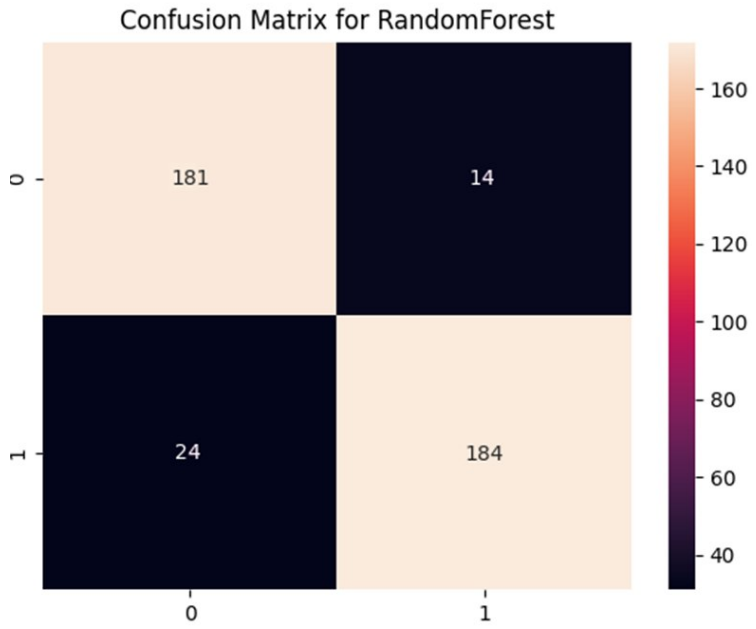



Рисунок 3.12 – Confusion Matrix for RandomForest

Ця матриця показує, що модель має високу точність класифікації, але також існують помилки, які варто враховувати. Зокрема, модель добре класифікує позитивні випадки, що важливо для зниження ризиків у кредитуванні.

Оцінка моделі показала високу точність класифікації, що склала 90.57%:

```
Accuracy for RandomForest: 90.5766666666666667
```

Такий результат демонструє ефективність Random Forest у задачах кредитного скорингу.

3. ***Naive Bayes*** є ймовірнісною моделлю, яка базується на застосуванні теореми Байєса з припущенням про незалежність ознак між собою.

Для зрозуміння впливу кожної ознаки на класифікаційне рішення було використано візуалізацію, яка показує вплив ознак на модель. Візуалізація допомагає ідентифікувати, які ознаки мають найбільше значення для рішень моделі (рис. 3.13).

```
data = pd.read_csv('german_credit_cleaned.csv')
categorical_columns = ['checking_acc_status', 'cred_hist', 'purpose', 'saving_acc_bonds',
                       'present_employment_since', 'personal_stat_gender', 'other_debtors_guarantors',
                       'property', 'other_installment_plans', 'housing', 'job', 'telephone',
                       'is_foreign_worker', 'target']
label_encoders = {}
for column in categorical_columns:
    le = LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le
X = data.drop(labels: 'target', axis=1)
y = data['target']

smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X, y)

X_train, X_test, y_train, y_test = train_test_split(*arrays: X_resampled, y_resampled, test_size=0.3, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

nb = GaussianNB()
nb.fit(X_train_scaled, y_train)

feature_effect = abs(nb.theta_[0] - nb.theta_[1]) / (nb.var_[0] + nb.var_[1])
sns.barplot(x=feature_effect, y=X.columns)
plt.title('Feature Effects for GaussianNB')
plt.show()

y_pred_nb = nb.predict(X_test_scaled)
print('Accuracy for GaussianNB:', accuracy_score(y_test, y_pred_nb))
cm_nb = confusion_matrix(y_test, y_pred_nb)
sns.heatmap(cm_nb, annot=True, fmt='d')
plt.title('Confusion Matrix for GaussianNB')
plt.show()

joblib.dump(nb, filename: 'gaussian_nb_model.joblib')
```

Рисунок 3.13 – Вміст файлу тренування та виводу результатів моделі Naive Bayes

На рисунку 3.14 представлено важливість кожної ознаки для моделі Random Forest. Видно, що найбільший внесок у прогнозування робить параметр acc_status. Інші параметри дають більш-менш рівний вклад.

Ця діаграма допомагає зрозуміти, які ознаки найбільше впливають на рішення моделі, і може бути використаний для подальшої оптимізації та інтерпретації моделі.

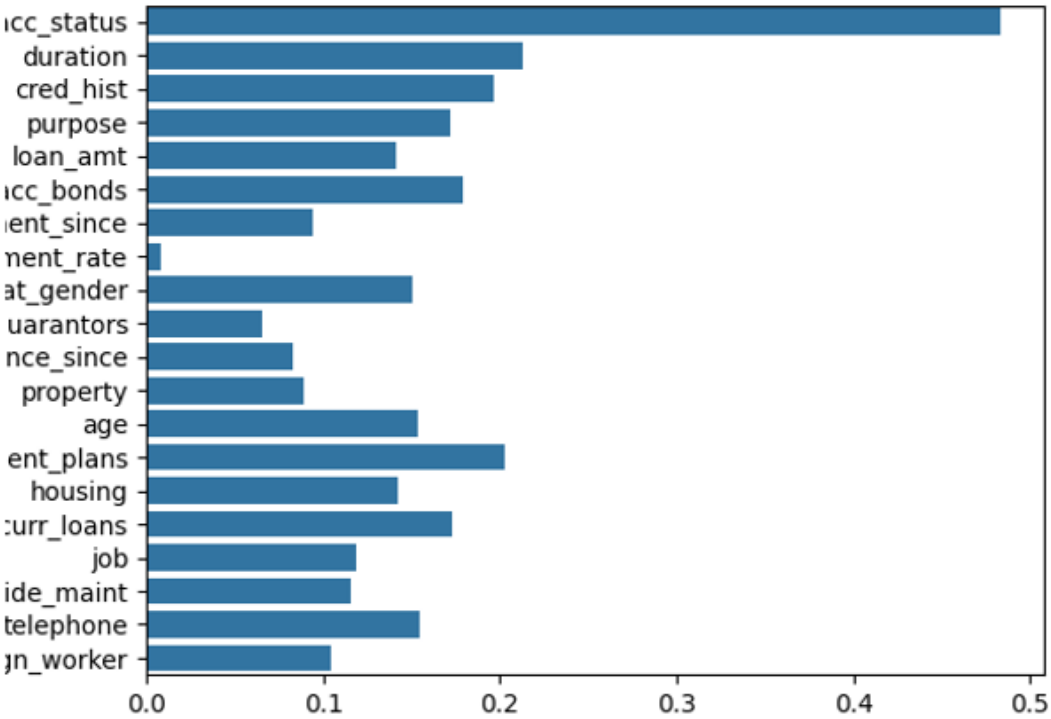



Рисунок 3.14 – Діаграма важливості ознак для моделі Naive Bayes

На рисунку 3.15 представлено матрицю помилок для моделі Naive Bayes, яка використовується для оцінки точності класифікації моделі. Матриця помилок показує, скільки зразків було правильно та неправильно класифіковано. Ось як можна інтерпретувати дані в матриці, прийнявши, клас 0 – це «хороший» кредитний ризик, а клас 1 – «поганий» кредитний ризик:

– ліва верхня клітинка: кількість зразків, які були правильно класифіковані як клас «0» TP =168;

– права верхня клітинка: кількість зразків, які належать до класу «0», але були помилково класифіковані як клас «1» FN =19;

– ліва нижня клітинка: кількість зразків, які належать до класу «1», але були помилково класифіковані як клас «0» FP =31;

– права нижня клітинка: кількість зразків, які були правильно класифіковані як клас «1» TN =171.

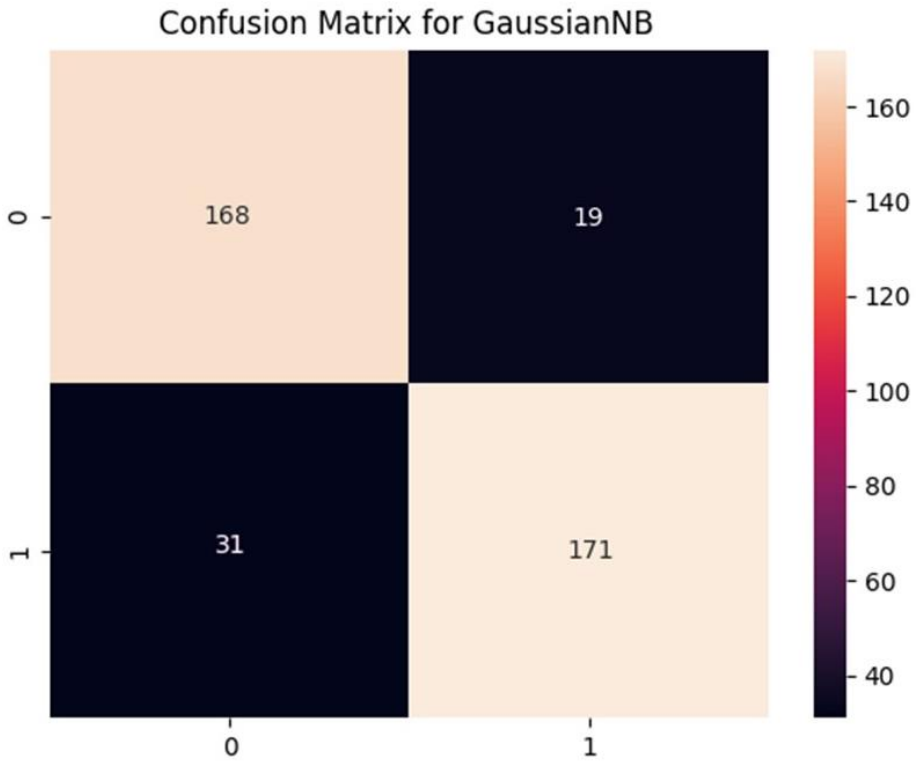



Рисунок 3.15 – Матриця помилок для моделі Naive Bayes

Модель Naive Bayes показала, що точність, розрахована за формулою 3.1 становить 87.19%, що є досить високим результатом для простих ймовірнісних моделей: `Accuracy for GaussianNB: 87.1919047619047619`.

Ця модель добре підходить для задач, де важлива швидкість обчислень і відносна простота реалізації.

### 3.3 Програмна реалізація системи

Інтерфейс системи відіграє ключову роль у забезпеченні зручного та інтуїтивного взаємодії користувачів з додатком, дозволяючи легко вводити необхідні дані та отримувати результати оцінки кредитного ризику. Розглянемо створення інтерфейсу та функціональність системи, а також інтеграцію і

використання класифікаційних моделей для оцінки кредитоспроможності клієнтів у мобільному застосунку.

***Створення Activity та Fragment.*** Activity та Fragment є основними концепціями в архітектурі Android-застосунків, які дозволяють ефективно організувати взаємодію з користувачем та керування інтерфейсом. Для реалізації застосунку створено основне Activity для подальшої навігації через фрагменти та основний фрагмент PredictFragment у якому і реалізована логіка обробки навчених моделей та вивід результатів на екран (рис. 3.16).

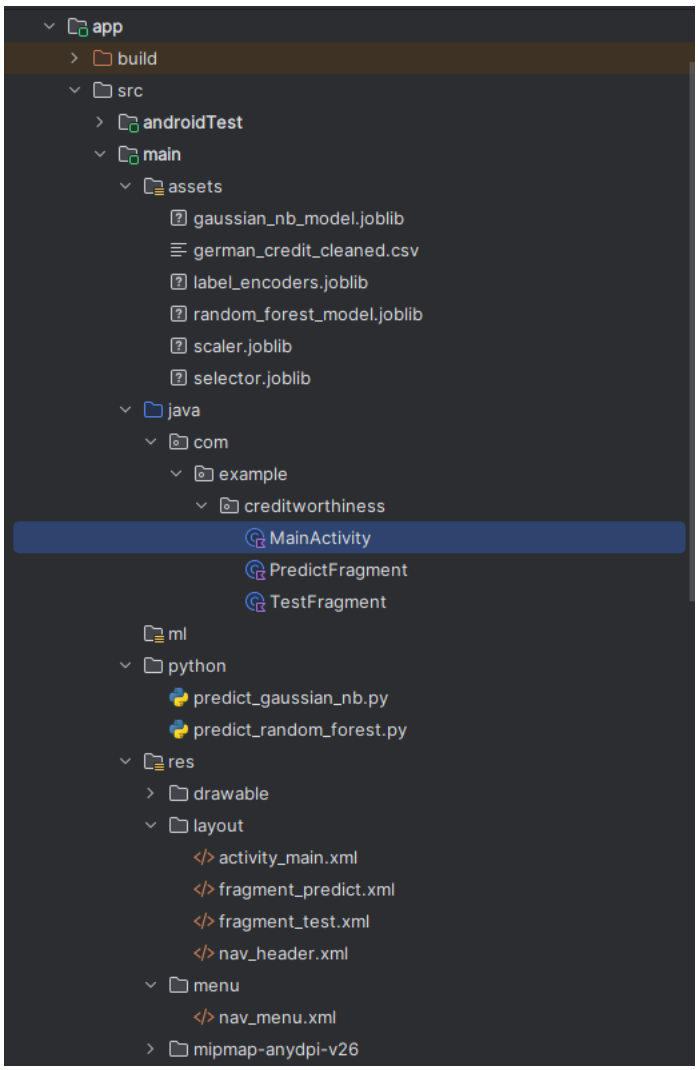



Рисунок 3.16 – Activity у структурі файлів проекту

Activity є однією з основних складових будь-якого Android-застосунку. Вона відповідає за створення вікна з користувацьким інтерфейсом, де користувач може взаємодіяти із застосунком. Головна Activity (MainActivity) буде містити основний інтерфейс із навігаційним меню, що дозволяє користувачеві перемикатися між різними частинами застосунку прогнозування кредитоспроможності (рис. 3.17). Навігаційне меню – це стандартна функція багатьох сучасних застосунків, яка дозволяє легко додавати різні фрагменти до майбутнього проекту, якщо будуть потрібні нові функції (налаштування, реєстрація тощо).

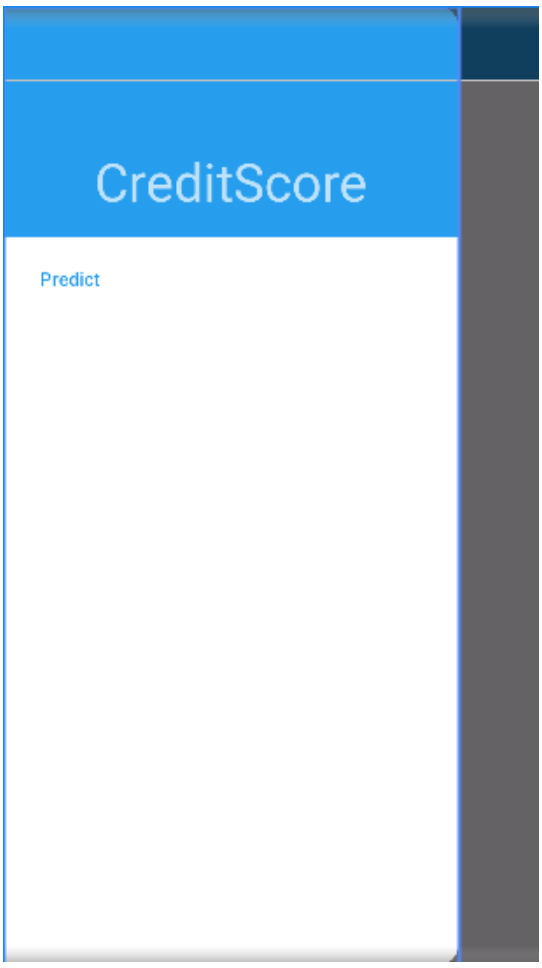



Рисунок 3.17 – Навігаційне меню

Завдяки правильному використанню Activity та Fragment, мобільний застосунок набуває модульності та зручності у використанні, що дозволяє

користувачам ефективно взаємодіяти з системою для оцінки кредитоспроможності клієнтів банку.

Важливу роль у визначенні вигляду та взаємодії користувацького інтерфейсу відіграють XML файли, забезпечуючи чітке розділення логіки програми від її візуального представлення.

***Основний макет*** (activity_main.xml). Файл activity_main.xml визначає основний макет головної Activity (MainActivity). Він містить такі елементи:

– DrawerLayout: основний контейнер, який дозволяє створювати навігаційне меню, що висувається зліва;

– LinearLayout: вертикальний лінійний макет, який містить Toolbar і контейнер для фрагментів;

– Toolbar: інструментальна панель, що розташована у верхній частині екрана і використовується для навігації та інших дій;

– FrameLayout: контейнер для фрагментів, який змінюється в залежності від вибраного пункту меню;

– NavigationView: навігаційне меню, що дозволяє користувачеві перемикатися між різними розділами застосунку.

***Макет для введення даних клієнта*** (fragment_predict.xml). Цей файл визначає макет для фрагмента, який відповідає за введення даних клієнта і відображення результатів прогнозування (рис. 3.18). Він містить такі елементи:

– ScrollView: контейнер для прокрутки, що дозволяє переглядати вміст, який перевищує розмір екрана;

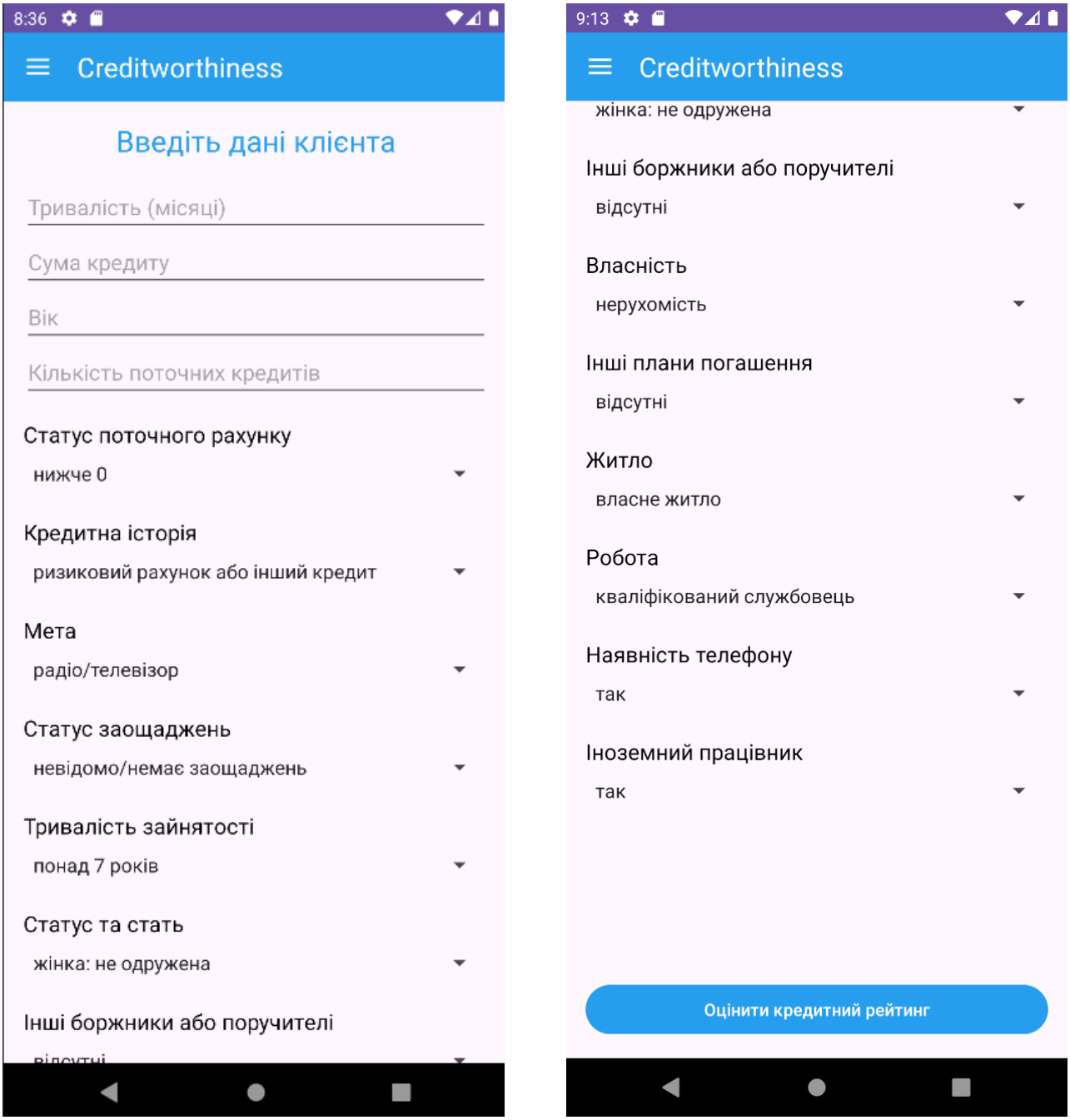
– LinearLayout: вертикальний лінійний макет, що містить всі інші елементи інтерфейсу;

– TextView: текстові поля, що використовуються для відображення заголовків і результатів;

– EditText: поля для введення числових даних клієнта (наприклад, тривалість кредиту, сума кредиту, вік);

– Spinner: випадаючі списки для введення категорійних даних (наприклад, статус рахунку, кредитна історія, мета кредиту);

– Button: кнопка для відправки введених даних та запуску процесу прогнозування.




*а*) частина 1 *б*) частина 2

Рисунок 3.18 – Фрагмент для введення даних клієнта

***Використання навчених моделей у Android-застосунку***. Розглянемо інтеграцію навчених моделей Random Forest та Naive Bayes у мобільний застосунок для оцінки кредитоспроможності клієнтів банку. Ці моделі були навчені на основі історичних даних і збережені у вигляді файлів для подальшого використання у застосунку. Для цього в проєкт було додано кілька необхідних файлів, а також створено два Python-скрипти для обробки даних клієнта. Було додано наступні файли:

– gaussian_nb_model.joblib – файл, що містить навчену модель Naive Bayes;

– label_encoders.joblib – файл, що містить енкодери для перетворення категорійних змінних у числові значення;

– random_forest_model.joblib – файл, що містить навчену модель Random Forest;

– scaler.joblib – файл, що містить об'єкт StandardScaler для нормалізації даних;

– selector.joblib – файл, що містить об'єкт для вибору найважливіших ознак (feature selector).

***Python-скрипти для обробки даних клієнта***. Для інтеграції моделей у мобільний застосунок були створені два Python-скрипти: predict_random_forest.py та predict_gaussian_nb.py. Ці скрипти обробляють введені дані клієнта, застосовуючи відповідні моделі для прогнозування кредитоспроможності.

Скрипт predict_random_forest.py завантажує необхідні файли, кодує введені дані клієнта, нормалізує їх, вибирає найважливіші ознаки та застосовує модель Random Forest для прогнозування (рис. 3.19). Основні етапи роботи скрипта:

– завантаження навчених моделей та енкодерів;

– кодування категорійних даних клієнта;

– нормалізація даних за допомогою scaler;

– вибір найважливіших ознак за допомогою selector;

– прогнозування кредитоспроможності за допомогою моделі Random Forest.

```python
import joblib
import pandas as pd

scaler = joblib.load('scaler.joblib')
selector = joblib.load('selector.joblib')
rf_model = joblib.load('random_forest_model.joblib')
label_encoders = joblib.load('label_encoders.joblib')

def predict_random_forest(client_data):
    new_sample_encoded = []
    for key, value in client_data.items():
        if key in label_encoders:
            new_sample_encoded.append(label_encoders[key].transform([value])[0])
        else:
            new_sample_encoded.append(value)

    feature_names = [
        'checking_acc_status', 'duration', 'cred_hist', 'purpose', 'loan_amt',
        'saving_acc_bonds', 'present_employment_since', 'installment_rate',
        'personal_stat_gender', 'other_debtors_guarantors', 'present_residence_since',
        'property', 'age', 'other_installment_plans', 'housing', 'num_curr_loans',
        'job', 'num_people_provide_maint', 'telephone', 'is_foreign_worker'
    ]

    new_sample_df = pd.DataFrame([new_sample_encoded], columns=feature_names)

    new_sample_scaled = scaler.transform(new_sample_df)
    new_sample_selected = selector.transform(new_sample_scaled)

    rf_prediction = rf_model.predict(new_sample_selected)[0]

    return int_(rf_prediction)

```

Рисунок 3.19 – Скрипт “predict_random_forest.py”

Скрипт predict_gaussian_nb.py працює аналогічно до predict_random_forest.py, але використовує модель Naive Bayes для прогнозування (рис. 3.20). Основні етапи роботи скрипта:

– завантаження навчених моделей та енкодерів;

– кодування категорійних даних клієнта;

– нормалізація даних за допомогою scaler;

– вибір найважливіших ознак за допомогою selector;

– прогнозування кредитоспроможності за допомогою моделі Naive Bayes.

```python
import joblib
import pandas as pd

scaler = joblib.load('scaler.joblib')
selector = joblib.load('selector.joblib')
rf_model = joblib.load('random_forest_model.joblib')
label_encoders = joblib.load('label_encoders.joblib')

def predict_random_forest(client_data):
    new_sample_encoded = []
    for key, value in client_data.items():
        if key in label_encoders:
            new_sample_encoded.append(label_encoders[key].transform([value])[0])
        else:
            new_sample_encoded.append(value)

    feature_names = [
        'checking_acc_status', 'duration', 'cred_hist', 'purpose', 'loan_amt',
        'saving_acc_bonds', 'present_employment_since', 'installment_rate',
        'personal_stat_gender', 'other_debtors_guarantors', 'present_residence_since'
        'property', 'age', 'other_installment_plans', 'housing', 'num_curr_loans',
        'job', 'num_people_provide_maint', 'telephone', 'is_foreign_worker'
    ]

    new_sample_df = pd.DataFrame([new_sample_encoded], columns=feature_names)

    new_sample_scaled = scaler.transform(new_sample_df)
    new_sample_selected = selector.transform(new_sample_scaled)

    rf_prediction = rf_model.predict(new_sample_selected)[0]

    return int_(rf_prediction)
```

Рисунок 3.20 – Скрипт “predict_random_forest.py”

Обидва скрипти повертають результат прогнозування у вигляді числового значення, яке інтерпретується як позитивне або негативне рішення щодо надання кредиту.

***Інтеграція у Android-застосунок***. У мобільному застосунку виклик цих Python-скриптів здійснюється за допомогою бібліотеки Chaquopy, яка дозволяє виконувати Python-код у середовищі Android (рис. 3.21). Після введення даних клієнта у відповідні поля інтерфейсу, застосунок відправляє ці дані до скриптів для обробки та отримання прогнозу. основні етапи інтеграції:

– введення даних клієнта у форму;

– валідація введених даних;

– виклик відповідного Python-скрипта для прогнозування;

– отримання та відображення результату прогнозування у інтерфейсі застосунку.

```
val pythonFileRF = python.getModule( name: "predict_random_forest")
val rfResult = pythonFileRF.callAttr( key: "predict_random_forest", clientData).toInt()
val pythonFileNB = python.getModule( name: "predict_gaussian_nb")
val nbResult = pythonFileNB.callAttr( key: "predict_gaussian_nb", clientData).toInt()
```

Рисунок 3.21 – Використання скриптів в PredictFragment

Моделі Random Forest, Naive Bayes та Decision Tree були інтегровані в систему з метою забезпечення високої точності прогнозування та надійності рішень. Кожна модель має свої унікальні особливості та переваги, які враховуються під час їх використання в додатку. В параграфі детально описано, як ці моделі взаємодіють з інтерфейсом системи, як користувач може вводити дані, та як відбувається обробка та відображення результатів оцінки кредитоспроможності.

### 3.4 Визначення кредитоспроможності клієнтів

Для отримання результату потрібно ввести дані клієнту , заповнивши числові та категоріальні поля зображені на рисунках 3.22 та 3.23.

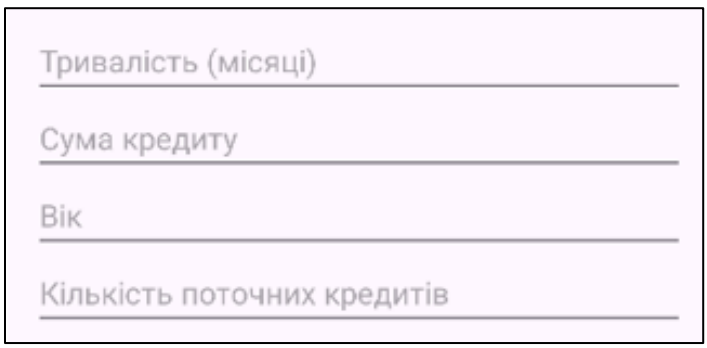
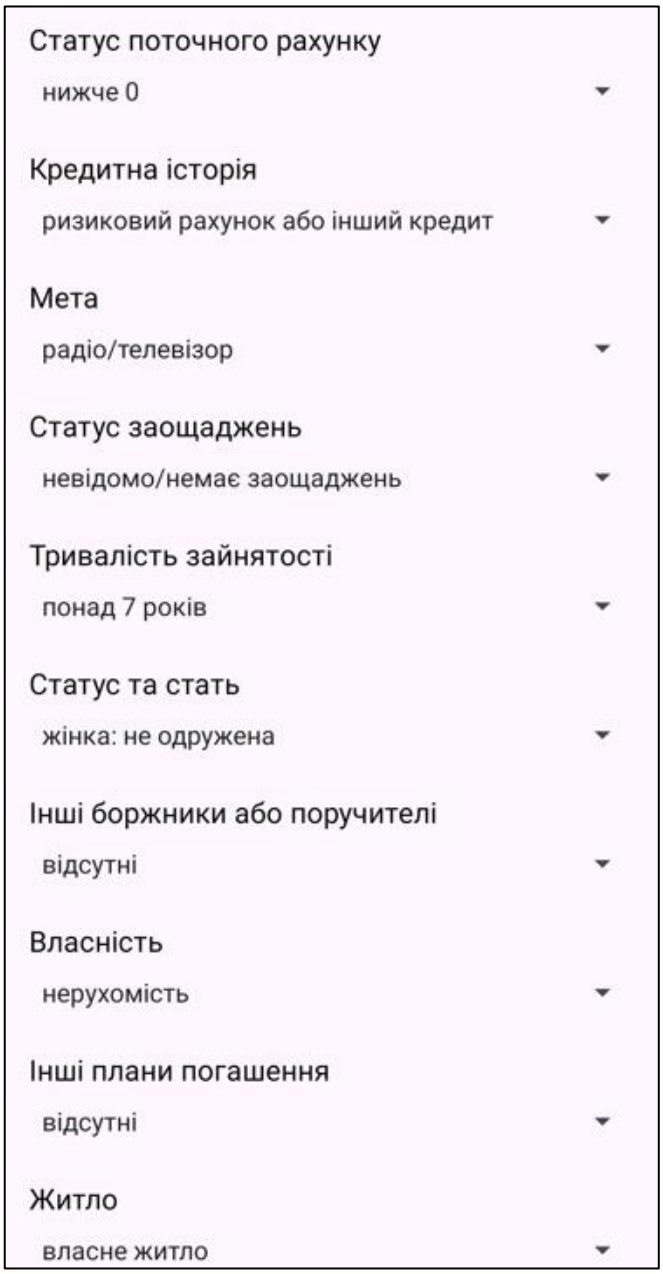



Рисунок 3.22 – Поля для введення числових даних




3.23 – Поля для вибору категоріальних даних

Після введення даних клієнта вони будуть оброблені моделями, які були описані в попередніх параграфах даного розділу.

Після обробки виведеться результат за відповідних до кожної моделі. Якщо модель оцінила клієнта як “гарного”, то результат рішення про надання кредиту буде позитивним, якщо оцінка даних клієнту “негативна”, то рішення буде негативним.

Для демонстрації різних результатів застосунку та його роботи введемо трьох клієнтів з різними даними.

У першому прикладі обрано основні характеристики які найбільш сильно можуть позитивно вплинути на подальшу оцінку, а саме (рис. 3.24):

– тривалість кредиту: 10 місяців;
– сума кредиту: 1200 німецьких марок;
– кількість поточних кредитів: 0 кредитів;
– вік: 35 років;
– кредитна історія: кредит виплачується вчасно в цьому банку;
– робота: кваліфікований службовець.

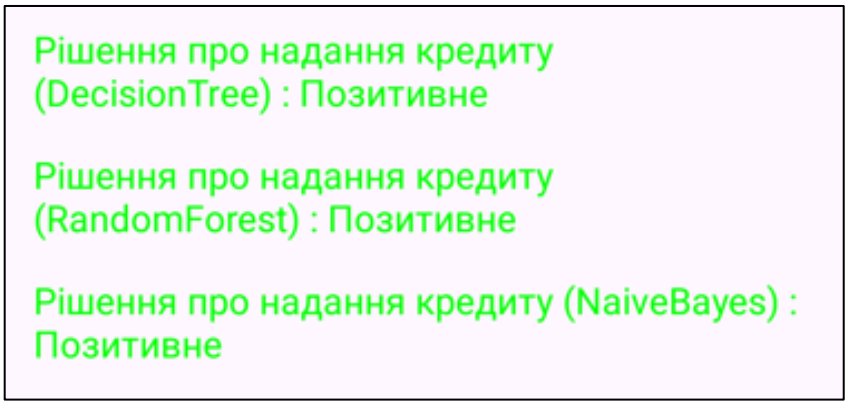



Рисунок 3.24 – Приклад результату з позитивним рішенням

У другому прикладі обрано основні характеристики які найбільш сильно можуть негативно вплинути на подальшу оцінку, а саме (рис. 3.25):

– тривалість кредиту: 34 місяці;
– сума кредиту: 6000 німецьких марок;
– кількість поточних кредитів: 2 кредитів;
– вік: 18 років;
– кредитна історія: ризиковий рахунок або інший кредит;
– робота: безробітний.

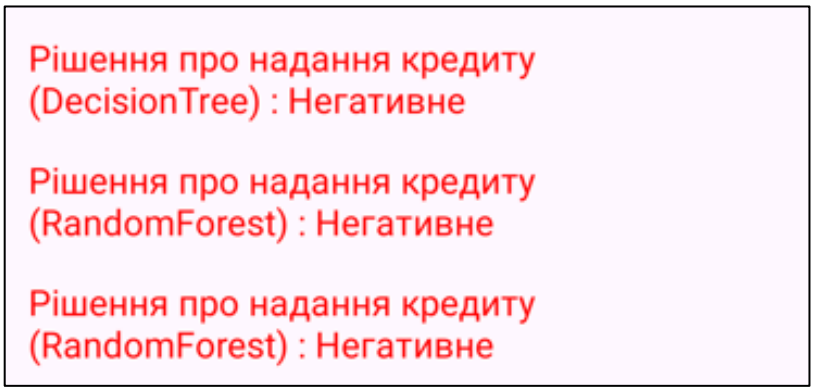



Рисунок 3.25 – Приклад результату з негативним рішенням

У третьому прикладі обрано основні характеристики які можна віднести до неоднозначного клієнту, а саме (рис. 3.26):

- тривалість кредиту: 24 місяці;
- сума кредиту: 2500 німецьких марок;
- кількість поточних кредитів: 1 кредитів;
- вік: 25 років;
- кредитна історія: затримки у минулому;
- робота: управління або самозайнятість;

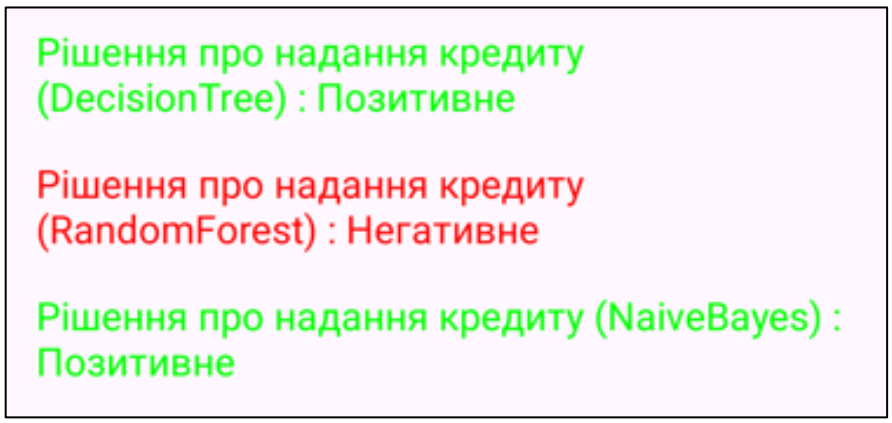



Рисунок 3.26 – Приклад результату з неоднозначним рішенням

При визначенні кредитоспроможності клієнта застосунок виводить результат, отриманий за трьома реалізованими моделями. А працівник банку, який

результат переглядає, приймає остаточне рішення відповідно до отриманих результатів.

## Висновки до розділу 3

У третьому розділі було здійснено аналіз та характеристику набору даних, який використовувався для навчання класифікаційних моделей Random Forest та Naive Bayes. У застосунку для оцінки кредитоспроможності було створено три моделі Random Forest, Naive Bayes та Decision Tree. Кожна модель була адаптована до специфіки даних і потреб користувачів, із зосередженням на забезпеченні високої точності прогнозування та здатності ефективно обробляти реальні сценарії використання. Моделі Random Forest та Naive Bayes інтегровані безпосередньо в мобільний застосунок, що дозволяє проводити аналіз на пристрої користувача без потреби в постійному з'єднанні з сервером.

Описано програмну розробку та основні компоненти користувальницького інтерфейсу мобільного застосунку визначення кредитоспроможності клієнтів, включаючи форми для введення даних, елементи відображення результатів та інше. Використання навчених моделей Random Forest та Naive Bayes у мобільному застосунку забезпечує швидке та точне оцінювання кредитоспроможності клієнтів на основі введених даних.

# ВИСНОВКИ

Розробка мобільного застосунку для визначення кредитоспроможності клієнтів банку викликала значний інтерес через зростання важливості цифрових рішень у фінансовій індустрії. В умовах сучасного банківського сервісу, де швидкість та точність обробки даних грають критичну роль, впровадження технологічно вдосконалених рішень може значно покращити процеси оцінювання кредитоспроможності та зменшити ризики.

Проведене дослідження дозволяє зробити наступні висновки. Аналіз теоретичних засад визначення кредитоспроможності клієнтів банку – предметної сфери, ролі діджиталізації у процесі оцінки кредитоспроможності, а також оптимізації банківських процесів з використанням сучасних технологій дозволив установити, що кредитоспроможність клієнтів охоплює комплексну оцінку різноманітних аспектів, включаючи історію попередніх позик, поведінку рахунків, кредитні рейтинги та особистісні дані.

Установлено, що діджиталізація сприяє інтеграції процесів оцінки кредитоспроможності у мобільні додатки, що дозволяє покращити оперативність та доступність банківських послуг. Мобільні технології дозволяють збирати велику кількість даних у реальному часі, що підвищує точність кредитних оцінок і дає змогу банкам надавати персоналізовані фінансові послуги. Розглянуто важливість оптимізації банківських процесів, яка включає автоматизацію збору та аналізу даних, що зменшує людську помилку та підвищує ефективність банківських операцій. Розробка мобільних додатків для працівників банку сприяє більшій оперативності та ефективності прийняття кредитних рішень.

Здійснений аналіз підходів до визначення кредитоспроможності клієнтів банку дозволив обґрунтувати вибір моделей Random Forest, Naive Bayes та Decision Tree.

Програмна реалізація застосунку виконана на мові Kotlin у середовищі Android Studio, що забезпечило високу адаптивність і легкість використання

рішення. Інтеграція з Python через бібліотеку Chaquopy дозволила використати наявні Python-сценарії та бібліотеки безпосередньо в мобільному додатку.

У застосунку для оцінки кредитоспроможності реалізовано використання трьох моделей Random Forest, Naive Bayes та Decision Tree. Кожна модель була адаптована до специфіки даних і потреб користувачів, із зосередженням на забезпеченні високої точності прогнозування та здатності ефективно обробляти реальні сценарії використання. Моделі Random Forest та Naive Bayes інтегровані безпосередньо в мобільний застосунок, що дозволяє проводити аналіз на пристрої користувача без потреби в постійному з'єднанні з сервером. Використання навчених моделей у мобільному застосунку забезпечує швидке та точне оцінювання кредитоспроможності клієнтів на основі введених даних.

Окрім технічних аспектів, особливу увагу було приділено безпеці обробки даних, що важливо для дотримання стандартів конфіденційності у банківській сфері. Введення даних клієнтами через зручний і інтуїтивно зрозумілий інтерфейс, а також швидка обробка та візуалізація результатів забезпечили високу ефективність застосунку.

У результаті, мобільний застосунок демонструє здатність значно спростити процеси визначення кредитоспроможності, забезпечуючи банківським працівникам потужний інструмент для швидкого та ефективного прийняття рішень. Це відкриває нові перспективи для оптимізації банківських операцій та підвищення рівня клієнтського обслуговування, одночасно підвищуючи рівень довіри та задоволеності клієнтів.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мармоза А.Т. Теорія статистики : підручник. К.: Центр учбової літератури, 2013. 592 с.

2. Кредитоспроможність / А. Г. Чубенко, М. В. Лошицький, Д. М. Павлов, С. С. Бичкова, О. С. Юнін. – Київ : Ваіте, 2018. – 376 с.

3. Creditworthiness: How to Check and Improve It. Investopedia: вебсайт. URL: https://www.investopedia.com/terms/c/creditworthiness.asp (дата звернення: 10.05.2024).

4. Bank Credit Analysis. Wall Street Oasis: вебсайт. URL: https://www.wallstreetoasis.com/resources/skills/finance/bank-credit-analysis (дата звернення: 11.05.2024).

5. Understanding Creditworthiness Assessment: How Banks Evaluate Your Borrowing Potential. Enrichest: вебсайт. URL: https://www.enrichest.com/creditworthiness-assessment (дата звернення: 11.05.2024).

6. Credit Analysis. Wall Street Prep: вебсайт. URL: https://www.wallstreetprep.com/knowledge/credit-analysis/ (дата звернення: 20.05.2024).

7. Designing next-generation credit-decisioning models. McKinsey: вебсайт. URL: https://www.mckinsey.com/business-functions/risk-and-resilience/our-insights/designing-next-generation-credit-decisioning-models (дата звернення: 24.05.2024).

8. Methodology and Models for Individuals' Creditworthiness Management Using Digital Footprint Data and Machine Learning Methods. MDPI. URL: https://www.mdpi.com/2076-3417/9/15/1820 (дата звернення: 24.05.2024).

9. Оцінювання кредитоспроможності фізичних осіб. URL: https://pidru4niki.com/1494051145721/bankivska_sprava/otsinyuvannya_kreditospromozhnosti_fizichnih_osib (дата звернення: 25.05.2024).

10. Decision tree : вебсайт. URL: https://en.wikipedia.org/wiki/Decision_tree (дата звернення: 26.05.2024).

11. Random forest : вебсайт. URL: https://en.wikipedia.org/wiki/Random_forest (дата звернення: 30.05.2024).

12. Naive Bayes classifier : вебсайт. URL: https://en.wikipedia.org/wiki/Naive_Bayes_classifier (дата звернення: 30.05.2024).

13. Android Studio : вебсайт. URL: https://en.wikipedia.org/wiki/Android_Studio (дата звернення: 30.05.2024).

14. Android Studio Reviews - Pros and Cons : вебсайт. URL: https://www.trustradius.com/products/android-studio/reviews?qs=pros-and-cons#pricing (дата звернення: 31.05.2024).

15. Java (programming language) : вебсайт. URL: https://en.wikipedia.org/wiki/Java_%28programming_language%29 (дата звернення: 1.06.2024).

16. Pros and Cons of Kotlin for Android App Development. URL: https://medium.com/quick-code/pros-and-cons-of-kotlin-for-android-app-development-c4b0f95c1324 (дата звернення: 1.06.2024).

17. PyCharm: вебсайт. URL: https://en.wikipedia.org/wiki/PyCharm (дата звернення: 2.06.2024).

18. PyCharm Official Site: вебсайт. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 2.06.2024).

19. What is PyCharm? : вебсайт. URL: https://hackr.io/blog/what-is-pycharm (дата звернення: 3.06.2024).

20. Scikit-learn : вебсайт. URL: https://en.wikipedia.org/wiki/Scikit-learn (дата звернення: 3.06.2024).

21. Scikit-learn Official Site : вебсайт. URL: https://scikit-learn.org/stable/index.html (дата звернення: 16.06.2024).

22. NumPy : вебсайт. URL: https://en.wikipedia.org/wiki/NumPy (дата звернення: 3.06.2024).

23. What is NumPy? : вебсайт. URL: https://numpy.org/doc/1.21/user/whatisnumpy.html (дата звернення: 3.06.2024).

24. Pandas (software) : вебсайт. URL: https://en.wikipedia.org/wiki/Pandas_(software) (дата звернення: 3.06.2024).

25. Python Pandas Tutorial: A Complete Introduction : вебсайт. URL: https://www.learndatasci.com/tutorials/python-pandas-tutorial-complete-introduction-for-beginners/ (дата звернення: 3.06.2024).

26. Matplotlib : вебсайт. URL: https://en.wikipedia.org/wiki/Matplotlib (дата звернення: 3.06.2024).

27. Matplotlib in Python : вебсайт. URL: https://www.datacamp.com/tutorial/matplotlib-tutorial-python (дата звернення: 4.06.2024).

28. Android App Components Fundamentals : вебсайт. URL: https://developer.android.com/guide/components/fundamentals (дата звернення: 5.06.2024).

# ДОДАТОК А

## Код мобільного застосунку

Predict fragment:

```
package com.example.creditworthiness

import android.graphics.Color
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ArrayAdapter
import android.widget.Button
import android.widget.EditText
import android.widget.Spinner
import android.widget.TextView
import androidx.fragment.app.Fragment
import com.chaquo.python.Python
import kotlin.random.Random

class PredictFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_predict, container, false)

        setupSpinners(view)

        val btnSubmit: Button = view.findViewById(R.id.btn_submit)
        val resultDecisionTreeTextView: TextView = view.findViewById(R.id.result_decision_tree_text)
        val resultNaiveBayesTextView: TextView = view.findViewById(R.id.result_naive_bayes_text)
        val resultRandomForestTextView: TextView = view.findViewById(R.id.result_random_forest_text)

        btnSubmit.setOnClickListener {
            if (validateInputs(view)) {
                val clientData = getClientData(view)
                val creditScore = evaluateCreditScore(clientData)
                val python = Python.getInstance()

                val pythonFileRF = python.getModule("predict_random_forest")
                val rfResult = pythonFileRF.callAttr("predict_random_forest", clientData).toInt()
                val pythonFileNB = python.getModule("predict_gaussian_nb")
                val nbResult = pythonFileNB.callAttr("predict_gaussian_nb", clientData).toInt()

                val rfDecision = if (rfResult == 1) "Рішення про надання кредиту (RandomForest) : Позитивне" else
"Рішення про надання кредиту (RandomForest) : Негативне"
                val nbDecision = if (nbResult == 1) "Рішення про надання кредиту (NaiveBayes) : Позитивне" else "Рішення
про надання кредиту (NaiveBayes) : Негативне"

                resultRandomForestTextView.text = rfDecision
                resultNaiveBayesTextView.text = nbDecision

                if (rfResult == 1) { resultRandomForestTextView.setTextColor(Color.GREEN) }
                else { resultRandomForestTextView.setTextColor(Color.RED) }
                if (nbResult == 1) { resultNaiveBayesTextView.setTextColor(Color.GREEN) }
                else { resultNaiveBayesTextView.setTextColor(Color.RED) }
```

```
            if (Integer.valueOf(creditScore) >= 500) { resultDecisionTreeTextView.text = "Рішення про надання кредиту
(DecisionTree) : Позитивне"
                resultDecisionTreeTextView.setTextColor(Color.GREEN) }
            else { resultDecisionTreeTextView.text = "Рішення про надання кредиту (DecisionTree) : Негативне"
                resultDecisionTreeTextView.setTextColor(Color.RED) }

        } else {
            resultDecisionTreeTextView.text = "Будь ласка, заповніть усі поля правильно."
        }
    }


    return view
}

private fun setupSpinners(view: View) {
    setupSpinner(view, R.id.spinner_checking_acc_status, R.array.checking_acc_status_options)
    setupSpinner(view, R.id.spinner_cred_hist, R.array.cred_hist_options)
    setupSpinner(view, R.id.spinner_purpose, R.array.purpose_options)
    setupSpinner(view, R.id.spinner_saving_acc_bonds, R.array.saving_acc_bonds_options)
    setupSpinner(view, R.id.spinner_present_employment_since, R.array.present_employment_since_options)
    setupSpinner(view, R.id.spinner_personal_stat_gender, R.array.personal_stat_gender_options)
    setupSpinner(view, R.id.spinner_other_debtors_guarantors, R.array.other_debtors_guarantors_options)
    setupSpinner(view, R.id.spinner_property, R.array.property_options)
    setupSpinner(view, R.id.spinner_other_installment_plans, R.array.other_installment_plans_options)
    setupSpinner(view, R.id.spinner_housing, R.array.housing_options)
    setupSpinner(view, R.id.spinner_job, R.array.job_options)
    setupSpinner(view, R.id.spinner_telephone, R.array.telephone_options)
    setupSpinner(view, R.id.spinner_is_foreign_worker, R.array.is_foreign_worker_options)
}

private fun validateInputs(view: View): Boolean {
    val spinnersAreValid = listOf(
        R.id.spinner_checking_acc_status, R.id.spinner_cred_hist, R.id.spinner_purpose,
        R.id.spinner_saving_acc_bonds, R.id.spinner_present_employment_since,
        R.id.spinner_personal_stat_gender, R.id.spinner_other_debtors_guarantors,
        R.id.spinner_property, R.id.spinner_other_installment_plans,
        R.id.spinner_housing, R.id.spinner_job, R.id.spinner_telephone,
        R.id.spinner_is_foreign_worker
    ).all {
        val spinner: Spinner = view.findViewById(it)
        spinner.selectedItem != null
    }

    val inputsAreValid = listOf(
        R.id.input_duration, R.id.input_loan_amt, R.id.input_age,
        R.id.input_num_curr_loans
    ).all {
        val editText: EditText = view.findViewById(it)
        editText.text.isNotEmpty() && editText.text.toString().toIntOrNull() != null
    }

    return spinnersAreValid && inputsAreValid
}

private fun setupSpinner(view: View, spinnerId: Int, arrayId: Int) {
    val spinner: Spinner = view.findViewById(spinnerId)
    ArrayAdapter.createFromResource(
        requireContext(),
```

```
            arrayId,
            android.R.layout.simple_spinner_item
        ).also { adapter ->
            adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)
            spinner.adapter = adapter
        }
    }

    private fun getClientData(view: View): Map<String, Any?> {
        return mapOf(
            "checking_acc_status" to (view.findViewById<Spinner>(R.id.spinner_checking_acc_status).selectedItem as?
String),
            "duration" to (view.findViewById<EditText>(R.id.input_duration).text.toString().toIntOrNull()),
            "cred_hist" to (view.findViewById<Spinner>(R.id.spinner_cred_hist).selectedItem as? String),
            "purpose" to (view.findViewById<Spinner>(R.id.spinner_purpose).selectedItem as? String),
            "loan_amt" to (view.findViewById<EditText>(R.id.input_loan_amt).text.toString().toIntOrNull()),
            "saving_acc_bonds" to (view.findViewById<Spinner>(R.id.spinner_saving_acc_bonds).selectedItem as? String),
            "present_employment_since" to
(view.findViewById<Spinner>(R.id.spinner_present_employment_since).selectedItem as? String),
            "personal_stat_gender" to (view.findViewById<Spinner>(R.id.spinner_personal_stat_gender).selectedItem as?
String),
            "other_debtors_guarantors" to
(view.findViewById<Spinner>(R.id.spinner_other_debtors_guarantors).selectedItem as? String),
            "property" to (view.findViewById<Spinner>(R.id.spinner_property).selectedItem as? String),
            "age" to (view.findViewById<EditText>(R.id.input_age).text.toString().toIntOrNull()),
            "other_installment_plans" to (view.findViewById<Spinner>(R.id.spinner_other_installment_plans).selectedItem
as? String),
            "housing" to (view.findViewById<Spinner>(R.id.spinner_housing).selectedItem as? String),
            "num_curr_loans" to (view.findViewById<EditText>(R.id.input_num_curr_loans).text.toString().toIntOrNull()),
            "job" to (view.findViewById<Spinner>(R.id.spinner_job).selectedItem as? String),
            "telephone" to (view.findViewById<Spinner>(R.id.spinner_telephone).selectedItem as? String),
            "is_foreign_worker" to (view.findViewById<Spinner>(R.id.spinner_is_foreign_worker).selectedItem as? String)
        )
    }

    private fun evaluateCreditScore(client: Map<String, Any?>): String {
        val loanAmt = client["loan_amt"] as? Int ?: 0
        val age = client["age"] as? Int ?: 0
        val duration = client["duration"] as? Int ?: 0
        val numCurrLoans = client["num_curr_loans"] as? Int ?: 0

        if (loanAmt > 10000) return "Відмова: занадто великий розмір кредиту"
        if (age < 18 || age > 80) return "Відмова: невідповідний вік"
        if (duration > 60) return "Відмова: занадто довга тривалість кредиту"
        if (numCurrLoans > 3) return "Відмова: занадто багато поточних кредитів"

        var score = 500

        when {
            loanAmt > 5000 -> score -= 200
            loanAmt > 3000 -> score -= 100
            loanAmt < 1000 -> score += 100
            loanAmt < 500 -> score += 200
        }

        when {
            duration > 36 -> score -= 100
            duration > 24 -> score -= 50
            duration < 12 -> score += 50
            duration < 6 -> score += 100
```

```
}

when {
    age < 25 -> score -= 50
    age > 60 -> score -= 20
    age in 30..50 -> score += 30
}

when (client["checking_acc_status"]) {
    "нижче 0" -> score -= 100
    "нижче 200" -> score -= 50
    "без рахунку" -> score -= 20
    "понад 200" -> score += 50
}

when (client["cred_hist"]) {
    "ризиковий рахунок або інший кредит" -> score -= 100
    "затримки в минулому" -> score -= 50
    "поточні кредити виплачуються вчасно" -> score += 20
    "немає кредиту або інший кредит виплачується вчасно" -> score += 50
    "кредит виплачується вчасно в цьому банку" -> score += 100
}

when (client["purpose"]) {
    "радіо/телевізор", "побутова техніка" -> score -= 30
    "новий автомобіль", "вживаний автомобіль", "меблі/обладнання" -> score += 20
    "освіта", "бізнес" -> score += 50
}

when (client["saving_acc_bonds"]) {
    "невідомо/немає заощаджень" -> score -= 50
    "нижче 100" -> score -= 20
    "нижче 1000" -> score += 20
    "понад 1000" -> score += 50
}

when (client["present_employment_since"]) {
    "безробітний" -> score -= 50
    "нижче 1 року" -> score -= 50
    "нижче 4 років" -> score -= 20
    "нижче 7 років" -> score += 20
    "понад 7 років" -> score += 100
}

when (client["personal_stat_gender"]) {
    "жінка: не одружена" -> score -= 20
    "жінка: одружена" -> score += 20
    "чоловік: не одружений" -> score -= 20
    "чоловік: одружений" -> score += 20
}

when (client["other_debtors_guarantors"]) {
    "поручитель" -> score += 20
    "співпозичальник" -> score -= 20
}

when (client["property"]) {
    "нерухомість" -> score += 50
    "автомобіль або інше" -> score += 20
}
```

```
        when (client["other_installment_plans"]) {
            "банк" -> score -= 20
            "магазин" -> score -= 10
        }

        when (client["housing"]) {
            "оренда" -> score -= 20
            "власне житло" -> score += 20
        }

        when {
            numCurrLoans > 2 -> score -= 30
            numCurrLoans == 1 -> score += 20
        }

        when (client["job"]) {
            "безробітний нерезидент" -> score -= 100
            "некваліфікований резидент" -> score -= 50
            "кваліфікований службовець" -> score += 50
            "управління або самозайнятість" -> score += 50
        }

        when {
            client["num_people_provide_maint"] as? Int ?: 0 > 1 -> score -= 20
        }

        when (client["telephone"]) {
            "так" -> score += 20
        }

        when (client["is_foreign_worker"]) {
            "так" -> score -= 50
        }

        if (score > 1000) score = 1000
        if (score < 1) score = 1

        return score.toString()
    }
}
```