МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ВІДТОКУ КЛІЄНТІВ»**

Виконав: студент 4 курсу, групи 6.1211-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
М.Ю. Цикало
(ініціали та прізвище)

Керівник професор кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Цикало Максиму Юрійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інформаційної системи прогнозування відтоку клієнтів

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз сучасних методів прогнозування відтоку клієнтів.
2. Проєктування інформаційної системи
3. Використання інформаційної системи.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.01.2025	
2.	Збір вихідних даних.	27.01.2025	
3.	Обробка методичних та теоретичних	21.02.2025	
	джерел.		
4.	Розробка першого та другого розділу.	15.04.2025	
5.	Розробка третього розділу.	19.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	19.06.2025	

Студент _____ М.Ю. Цикало
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інформаційної системи прогнозування відтоку клієнтів»: 86 с., 31 рис., 13 табл., 40 джерел, 1 додаток.

ВІДТІК КЛІЄНТІВ, ІНФОРМАЦІЙНА СИСТЕМА, КЛІЄНТСЬКИЙ ІНТЕРФЕЙС, МАШИННЕ НАВЧАННЯ, ОБРОБКА ДАНИХ, ПРОГНОЗУВАННЯ, УТРИМАННЯ КЛІЄНТІВ, FLASK, RANDOMFOREST, STREAMLIT, SQLITE, TELCOCUSTOMERCHURN.

Об’єкт дослідження – процеси прогнозування відтоку клієнтів у телекомунікаційній сфері.

Мета роботи: розробка інформаційної системи для прогнозування відтоку клієнтів телекомунікаційної компанії на основі методів машинного навчання з використанням сучасних фреймворків.

Методи дослідження: методи машинного навчання (RandomForest), аналіз даних, системне проєктування, тестування програмного забезпечення.

Розроблено інформаційну систему прогнозування відтоку клієнтів, яка включає серверну частину (Flask), клієнтський інтерфейс (Streamlit) і модуль генерації тестових даних (Python). Використано набір даних TelcoCustomerChurn для навчання моделі RandomForest, яка забезпечує точність прогнозів 80-85%. Система дозволяє ідентифікувати клієнтів із високим ризиком відтоку, надавати рекомендації щодо їх утримання та візуалізувати результати. Практична цінність полягає в оптимізації стратегій утримання клієнтів, а теоретична – у систематизації підходів до прогнозування відтоку. Система адаптивна для телекомунікаційних компаній і може бути застосована в інших галузях.

SUMMARY

Bachelor’s qualifying paper “Development of an Information System for Customer Churn Prediction”: 86 pages, 31 figures, 13 tables, 40 references, 1 supplement.

CLIENT INTERFACE, CUSTOMER CHURN, CUSTOMER RETENTION, DATA PROCESSING, FLASK, INFORMATION SYSTEM, MACHINE LEARNING, PREDICTION, RANDOMFOREST, STREAMLIT, SQLITE, TELCOCUSTOMERCHURN.

The object of the study is the processes of customer churn prediction in the telecommunications sector.

The aim of the study is to develop an information system for predicting customer churn in a telecommunications company based on machine learning methods using modern frameworks.

The research methods include machine learning techniques (such as Random Forest), data analysis, systems design, and software testing.

An information system for customer churn prediction has been developed, comprising a server-side component (Flask), a client interface (Streamlit), and a test data generation module (Python). The TelcoCustomerChurn dataset was used to train the RandomForest model, achieving a prediction accuracy of 80-85%. The system enables identification of customers at high risk of churn, provides retention recommendations, and visualizes results. Its practical value lies in optimizing customer retention strategies, while its theoretical value is in systematizing churn prediction approaches. The system is adaptable for telecommunications companies and applicable to other industries.

ЗМІСТ

Завдання на кваліфікаційну роботу ……..2
Реферат 4
Summary 5
Вступ..... 7
1 Аналіз сучасних методів прогнозування відтоку клієнтів..... 9
1.1 Огляд існуючих інформаційних систем та інструментів прогнозування 9
1.2 Аналіз методів машинного навчання для прогнозування відтоку..... 12
1.3 Постановка задачі 16
2 Проєктування інформаційної системи..... 20
2.1 Аналіз вимог до системи 20
2.2 Проєктування архітектури системи 24
2.3 Реалізація компонентів системи..... 31
2.1.1 Серверна частина (server.py)..... 32
2.1.2 Клієнтська частина (client.py)..... 34
2.1.3 Модуль генерації тестових даних (seed.py) 38
2.1.4 Особливості реалізації екранних форм..... 39
3 Використання інформаційної системи..... 41
3.1 Інтеграція системи в бізнес-процеси..... 41
3.2 Тестування системи 45
3.3 Аналіз результатів використання 55
Висновки 61
Перелік посилань..... 63
Додаток А Лістинг програми 67

ВСТУП

У сучасних умовах високої конкуренції на ринку телекомунікаційних послуг збереження клієнтської бази є одним із ключових факторів успіху компаній. Відтік клієнтів (churn) становить серйозну проблему, оскільки втрата клієнтів призводить до зменшення доходів і зростання витрат на залучення нових користувачів. У зв'язку з цим актуальність розробки інформаційних систем для прогнозування відтоку клієнтів набуває особливого значення. Такі системи дозволяють своєчасно виявляти клієнтів із високим ризиком відтоку, що дає змогу підприємствам застосовувати превентивні заходи, спрямовані на їх утримання.

Метою даної роботи є розробка інформаційної системи прогнозування відтоку клієнтів телекомунікаційної компанії на основі методів машинного навчання.

Для досягнення цієї мети було поставлено такі завдання:

– створення серверної частини для обробки даних і навчання моделі;
– розробка інтуїтивно зрозумілого клієнтського інтерфейсу для введення даних і відображення результатів прогнозування;
– генерація тестових даних для оцінки ефективності системи.

Об'єктом дослідження є процеси прогнозування відтоку клієнтів у телекомунікаційній сфері, а предметом – інформаційна система, що включає серверну (Flask), клієнтську (Streamlit) частини та модуль генерації даних (Python).

Історично проблема відтоку клієнтів досліджується з моменту зростання конкуренції в телекомунікаційній галузі наприкінці XX століття. З появою методів машинного навчання, зокрема алгоритмів класифікації, таких як Random Forest, з'явилися нові можливості для точного прогнозування поведінки клієнтів. У даній роботі використано набір даних Telco Customer Churn від IBM, який містить інформацію про 7043 клієнтів і 21 атрибут, що дозволяє моделювати поведінку користувачів із різними характеристиками.

Розроблена система складається з трьох основних компонентів: серверної частини (server.py), яка відповідає за ініціалізацію бази даних SQLite, завантаження та обробку даних, навчання моделі Random Forest і виконання прогнозів; клієнтської частини (client.py), реалізованої за допомогою Streamlit, що забезпечує зручний інтерфейс для введення даних клієнтів, відображення прогнозів і історії аналізу; а також модуля генерації тестових даних (seed.py), який створює профілі клієнтів із різними рівнями ризику відтоку. Система дозволяє прогнозувати ймовірність відтоку з точністю 80-85%, що підтверджується оцінкою моделі на тестових даних.

Теоретичне значення роботи полягає в систематизації підходів до прогнозування відтоку клієнтів із використанням сучасних технологій машинного навчання та аналізу даних. Практична цінність системи полягає в її здатності надавати телекомунікаційним компаніям інструмент для ідентифікації клієнтів із високим ризиком відтоку, пропонуючи персоналізовані рекомендації щодо їх утримання.

Новизна роботи полягає в інтеграції серверної та клієнтської частин із використанням сучасних фреймворків (Flask, Streamlit), що забезпечує простоту використання та високу ефективність системи. Отримані результати можуть бути застосовані в телекомунікаційних компаніях для оптимізації стратегій утримання клієнтів, а також адаптовані для інших галузей, де актуальна проблема відтоку.

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ ПРОГНОЗУВАННЯ ВІДТОКУ КЛІЄНТІВ

1.1 Огляд існуючих інформаційних систем та інструментів прогнозування

Прогнозування відтоку клієнтів (customer churn prediction) є однією з ключових задач сучасних інформаційних систем, спрямованих на підвищення рівня утримання клієнтів у різних галузях, зокрема телекомунікаціях, банківській сфері, електронній комерції та сфері послуг. Відтік клієнтів, тобто припинення співпраці з компанією, може мати значний негативний вплив на її дохідність і репутацію. За оцінками, залучення нового клієнта обходиться в 5-7 разів дорожче, ніж утримання існуючого [1]. У цьому контексті інформаційні системи, що використовують методи машинного навчання, дозволяють не лише ідентифікувати клієнтів із високим ризиком відтоку, але й пропонувати персоналізовані стратегії їх утримання.

Сучасні інформаційні системи для прогнозування відтоку клієнтів базуються на обробці великих обсягів даних (Big Data) і застосовують різноманітні алгоритми машинного навчання, такі як логістична регресія, дерева рішень, випадкові ліси (Random Forest), градієнтний бустинг (Gradient Boosting) та нейронні мережі. Ці системи інтегрують дані про поведінку клієнтів, демографічні характеристики, історію транзакцій і взаємодії з компанією для створення прогнозних моделей [2]. Наприклад, телекомунікаційні компанії, як показано в коді проєкту, використовують набори даних, що включають інформацію про тип контракту, тривалість співпраці, спосіб оплати та додаткові послуги, для прогнозування ймовірності відтоку [3].

Одним із найпоширеніших інструментів для створення таких систем є платформи машинного навчання, такі як TensorFlow, PyTorch, Scikit-learn та H2O.ai. Зокрема, Scikit-learn, використаний у коді проєкту, є популярною

бібліотекою для Python, яка надає широкий набір інструментів для обробки даних, навчання моделей і оцінки їх ефективності [4]. У коді застосовано алгоритм Random Forest, який характеризується високою точністю та стійкістю до перенавчання завдяки ансамблевому підходу, що комбінує результати кількох дерев рішень [5]. Цей алгоритм ефективно працює з категоріальними та числовими даними, що робить його придатним для аналізу складних наборів даних, таких як Telco Customer Churn, використаний у проєкті.

Серед комерційних інформаційних систем, що застосовуються для прогнозування відтоку, варто виділити Salesforce Customer 360, Microsoft Dynamics 365 Customer Insights та IBM Watson. Salesforce Customer 360 інтегрує дані з різних джерел, таких як CRM-системи, соціальні мережі та історія покупок, для створення єдиного профілю клієнта, що дозволяє прогнозувати відтік із використанням вбудованих моделей машинного навчання [6]. Microsoft Dynamics 365 використовує Azure Machine Learning для аналізу клієнтських даних і генерації персоналізованих рекомендацій щодо утримання клієнтів [7]. IBM Watson, у свою чергу, пропонує інструменти для прогнозування відтоку на основі штучного інтелекту, які дозволяють аналізувати неструктуровані дані, такі як текстові відгуки клієнтів [8].

Крім комерційних рішень, значну роль відіграють системи з відкритим кодом, такі як Apache Spark MLlib та KNIME. Apache Spark MLlib підтримує обробку великих даних і пропонує масштабовані алгоритми для прогнозування відтоку, що особливо важливо для компаній із великою клієнтською базою [9]. KNIME, як платформа з графічним інтерфейсом, дозволяє створювати складні аналітичні пайплайни без глибоких знань програмування, що робить її доступною для бізнес-аналітиків [10]. Ці платформи часто інтегруються з базами даних, такими як SQLite, використана в коді проєкту, для зберігання та обробки клієнтських даних.

Важливим показником сучасних систем є їх здатність до інтерактивної візуалізації та інтеграції з інтерфейсами користувача. У розробленому проєкті використано Streamlit, бібліотеку для створення вебдодатків із інтерактивними

формами та візуалізаціями, що дозволяє менеджерам і аналітикам легко взаємодіяти з системою [11]. Streamlit забезпечує швидке створення інтуїтивно зрозумілого інтерфейсу, де користувачі можуть вводити дані клієнтів, отримувати прогнози та переглядати історію аналізу. Такі інструменти підвищують доступність прогнозних систем для нефахівців у галузі даних.

Попри широкий вибір інструментів, сучасні системи стикаються з викликами, такими як якість даних, потреба в їх попередній обробці та проблема незбалансованих даних, коли кількість клієнтів, що залишаються, значно перевищує кількість тих, що йдуть. У коді проєкту ці проблеми вирішуються шляхом використання LabelEncoder для кодування категоріальних змінних і заповнення пропущених значень у TotalCharges, що є стандартною практикою в підготовці даних [4]. Крім того, для оцінки моделей часто застосовують метрики, такі як точність (accuracy), AUC-ROC або F1-score, що дозволяють оцінити якість прогнозування.

У телекомунікаційній галузі, як показано в проєкті, особлива увага приділяється факторам, що впливають на відтік, таким як тип контракту, термін співпраці та спосіб оплати. Дослідження показують, що клієнти з короткостроковими контрактами та високими щомісячними платежами мають вищий ризик відтоку [3]. Системи прогнозування, подібні до реалізованої в коді, дозволяють не лише ідентифікувати таких клієнтів, але й надавати рекомендації щодо їх утримання, наприклад, пропозицію знижок або зміни тарифного плану.

Сучасні інформаційні системи прогнозування відтоку клієнтів, як продемонстровано в реалізованому проєкті, ефективно поєднують алгоритми машинного навчання, обробку великих даних та інтуїтивні інтерфейси для вирішення бізнес-завдань. Використання бібліотеки Scikit-learn з алгоритмом Random Forest забезпечує високу точність прогнозів, а інтеграція з SQLite та Streamlit дозволяє створювати зручні та масштабовані рішення. Порівняно з комерційними платформами, такими як Salesforce чи IBM Watson, реалізована система є економічно вигідною альтернативою з відкритим кодом, що зберігає функціональність для аналізу даних і генерації рекомендацій. Водночас вона

ефективно вирішує типові виклики, такі як обробка незбалансованих даних і кодування категоріальних змінних, що підтверджує її практичну цінність для телекомунікаційної галузі.

1.2 Аналіз методів машинного навчання для прогнозування відтоку

Прогнозування відтоку клієнтів є важливою задачею для компаній, що прагнуть зберегти свою клієнтську базу та оптимізувати бізнес-процеси. У телекомунікаційній сфері, де конкуренція висока, а втрата клієнтів може суттєво вплинути на прибутковість, методи машинного навчання (ML) стали незамінними інструментами для аналізу поведінки клієнтів і передбачення їх відходу. Ці методи дозволяють виявляти приховані закономірності в даних, оцінювати ймовірність відтоку та пропонувати персоналізовані стратегії утримання. Доцільно розглянути основні методи машинного навчання, які застосовуються для прогнозування відтоку клієнтів (рис. 1.1), з акцентом на їх особливості, переваги та обмеження, а також їх релевантність до реалізації проєкту. Аналіз зосереджується на порівнянні їх ефективності, переваг та обмежень, а також на обґрунтуванні вибору методу Random Forest для прогнозування відтоку в телекомунікаційному датасеті.



Рисунок 1.1 – Основні методи машинного навчання для прогнозування відтоку клієнтів

Логістична регресія є одним із базових методів машинного навчання для задач бінарної класифікації, таких як прогнозування відтоку клієнтів (відтік/залишення). Цей метод оцінює ймовірність належності об’єкта до певного класу на основі лінійної комбінації ознак з подальшим застосуванням сигмоїдної функції [12]. У контексті прогнозування відтоку логістична регресія ефективна для інтерпретації впливу окремих ознак, таких як тривалість співпраці (tenure) або щомісячні витрати, на ймовірність відтоку. У коді прєкту(server.py) логістична регресія не використовується, оскільки вона менш ефективна для даних із нелінійними залежностями та великою кількістю категоріальних змінних, що характерно для телекомунікаційних датасетів. Однак цей метод залишається популярним у задачах, де потрібна простота та інтерпретованість результатів.

Дерева рішень є ще одним поширеним методом для прогнозування відтоку. Вони будують ієрархічну структуру правил, які розбивають дані на основі значень ознак, що дозволяє класифікувати клієнтів як тих, хто ймовірно залишить компанію, або тих, хто залишиться [13]. Перевагою дерев рішень є їх інтуїтивність та здатність обробляти як числові, так і категоріальні дані. Проте вони схильні до перенавчання, особливо на невеликих наборах даних. У реалізованому проєкті (server.py) використано ансамблевий метод на основі дерев рішень – Random Forest, що свідчить про перевагу більш складних підходів для підвищення точності прогнозів.

Ансамблеві методи, такі як Random Forest та Gradient Boosting, є одними з найефективніших для прогнозування відтоку клієнтів завдяки їх здатності комбінувати результати кількох базових моделей (дерев рішень) для підвищення точності та стійкості прогнозів. Random Forest, який використовується у коді проєкту(server.py), створює множину дерев рішень, кожне з яких навчається на випадковій підмножині даних і ознак, а остаточне рішення приймається шляхом голосування [5]. У коді модель RandomForestClassifier налаштована з параметрами n_estimators=100 та max_depth=10, що забезпечує баланс між точністю та обчислювальною ефективністю. Gradient Boosting, у свою чергу, будує дерева послідовно,

оптимізуючи функцію втрат, що дозволяє досягати високої точності на складних наборах даних [14]. Хоча Gradient Boosting не використовується у проєкті, він є популярною альтернативою Random Forest, особливо для датасетів із нерівномірним розподілом класів.

Нейронні мережі, зокрема глибокі нейронні мережі (DNN), застосовуються для прогнозування відтоку у випадках, коли дані мають високу розмірність або складні нелінійні залежності [15]. Вони здатні автоматично виділяти ознаки з необроблених даних, що зменшує потребу в ручній обробці даних. Однак нейронні мережі вимагають значних обчислювальних ресурсів і великих обсягів даних для ефективного навчання, що робить їх менш практичними для проєктів із обмеженими ресурсами, таких як представлений у коді. У даному проєкті перевага віддана Random Forest через його простоту в реалізації та високу точність на телекомунікаційних даних.

Метод опорних векторів (SVM) використовується для знаходження гіперплощини, яка найкраще розділяє класи у просторі ознак [16]. SVM ефективний для даних із чіткими межами між класами, але його продуктивність знижується при великій кількості ознак або шуму в даних. У телекомунікаційних датасетах, таких як використаний у проєкті (Telco Customer Churn), SVM рідко застосовується через складність налаштування гіперпараметрів та обчислювальну складність. Random Forest, реалізований у коді, є кращим вибором завдяки швидшій обробці даних і стійкості до шуму.

Ефективність методів машинного навчання значною мірою залежить від якості підготовки даних. У коді проєкту (server.py) використано методи попередньої обробки, такі як кодування категоріальних змінних за допомогою LabelEncoder та заповнення пропущених значень у змінній TotalCharges. Ці кроки є стандартними для підготовки даних до навчання моделей ML [17]. Наприклад, категоріальні змінні, такі як тип контракту чи спосіб оплати, кодуються для забезпечення сумісності з алгоритмами, що працюють із числовими даними. Крім того, у коді реалізовано збереження енкодерів

(ENCODERS_PATH) та назв колонок (COLUMNS_PATH) для повторного використання при прогнозуванні, що підвищує ефективність роботи системи.

Для оцінки моделей прогнозування відтоку зазвичай використовують метрики, такі як точність (accuracy), повнота (recall), точність передбачення (precision) та F1-оцінка. У коді проєкту модель Random Forest оцінюється за допомогою метрики точності (accuracy), яка досягає 80-85%, що є типовим показником для телекомунікаційних датасетів [18]. Порівняно з іншими методами, Random Forest має перевагу у стійкості до перенавчання та здатності обробляти категоріальні дані без необхідності складного налаштування гіперпараметрів. Однак для підвищення точності можна розглядати комбінацію методів, наприклад, ансамбль із Gradient Boosting або використання нейронних мереж для великих датасетів.

У розробленому проєкті вибір Random Forest є виправданим з кількох причин. По-перше, телекомунікаційний датасет (Telco Customer Churn) містить комбінацію числових і категоріальних ознак, таких як тип контракту, щомісячні витрати та тривалість співпраці, які добре обробляються Random Forest [19]. По-друге, алгоритм забезпечує високу точність без значних обчислювальних витрат, що важливо для систем реального часу, як у представленій реалізації (client.py). По-третє, Random Forest дозволяє оцінити важливість ознак, що може бути використано для надання рекомендацій щодо утримання клієнтів, як це реалізовано у рекомендаціях для високого, середнього та низького рівня ризику відтоку.

Незважаючи на ефективність методів машинного навчання, вони мають певні обмеження. Наприклад, Random Forest може бути менш ефективним при обробці незбалансованих даних, коли кількість клієнтів, що залишаються, значно перевищує кількість тих, що йдуть [20]. У коді проєкту (seed.py) створено тестові дані з різними профілями клієнтів (HIGH_, LOYAL_, MED_*), що дозволяє моделювати різні сценарії, але для реальних даних може знадобитися додаткове балансування класів. У майбутньому доцільно розглянути використання ансамблевих методів із адаптивним бустингом

(наприклад, AdaBoost) або нейронних мереж для підвищення точності на великих і складних наборах даних.

У результаті аналізу методів машинного навчання для прогнозування відтоку клієнтів встановлено, що Random Forest, використаний у проєкті, є оптимальним вибором для телекомунікаційних даних завдяки високій точності (80-85%), стійкості до шуму та здатності обробляти категоріальні ознаки. Порівняно з логістичною регресією, яка забезпечує інтерпретованість, але обмежена для нелінійних залежностей, та нейронними мережами, що потребують значних ресурсів, Random Forest пропонує баланс між ефективністю та простотою реалізації. Обмеженням методу є потенційна чутливість до незбалансованих даних, що вимагає додаткової обробки в реальних сценаріях. Подальший розвиток системи може включати інтеграцію Gradient Boosting або методів балансування класів для підвищення точності прогнозів.

1.3 Постановка задачі

Прогнозування відтоку клієнтів (customer churn prediction) є однією з ключових задач у сучасному бізнесі, зокрема в телекомунікаційній сфері, де конкуренція змушує компанії активно працювати над утриманням клієнтів. Відтік клієнтів, тобто припинення співпраці з компанією, може суттєво впливати на фінансові показники, оскільки залучення нових клієнтів зазвичай є більш витратним, ніж збереження існуючих [21]. У контексті розробки інформаційної системи, представленої в коді проєкту, задача прогнозування відтоку клієнтів полягає у створенні ефективного інструменту, який дозволяє передбачати ймовірність відмови клієнтів від послуг на основі їх даних, а також надавати рекомендації для запобігання відтоку.

Задача прогнозування відтоку клієнтів належить до категорії задач бінарної класифікації в машинному навчанні, де необхідно визначити, чи залишить клієнт компанію (клас 1 – відтік, клас 0 – залишиться) на основі

набору характеристик (атрибутів), таких як демографічні дані, тип контракту, спосіб оплати, тривалість співпраці та обсяг споживаних послуг. У коді проєкту для вирішення цієї задачі використовується датасет Telco Customer Churn від IBM, який містить інформацію про 7043 клієнтів із 21 атрибутом, включаючи цільову змінну – факт відтоку (Churn) [22]. Мета полягає в тому, щоб на основі історичних даних навчити модель машинного навчання, яка здатна точно передбачати ймовірність відтоку для нових клієнтів і зберігати результати прогнозів у базі даних для подальшого аналізу.

Для ефективного вирішення задачі необхідно врахувати кілька важливих питань, що показані в таблиці 1.1.

Таблиця 1.1 – Важливі показники для ефективного вирішення задачі

№	Показники	Пояснення
1	2	3
1	Якість даних	Дані про клієнтів часто містять пропуски, неконсистентні значення або потребують попередньої обробки. Наприклад, у коді server.py реалізовано обробку пропусків у колонці TotalCharges шляхом заміни їх значеннями з колонки MonthlyCharges, а також кодування категоріальних змінних за допомогою LabelEncoder [23]. Це дозволяє забезпечити коректну роботу моделі машинного навчання, яка чутлива до формату вхідних даних.
2	Вибір моделі	У реалізації проєкту використовується алгоритм Random Forest, який є ансамблевим методом на основі дерев рішень і демонструє високу точність у задачах класифікації завдяки здатності обробляти нелінійні залежності та взаємодії між змінними [24]. Вибір цього алгоритму обґрунтований його стійкістю до шуму в даних та відносною простотою інтерпретації.
3	Інтерфейс користувача	Для практичного використання прогнозів розроблено вебінтерфейс на основі бібліотеки Streamlit, який дозволяє вводити дані клієнтів, отримувати прогнози та переглядати історію аналізу [25]. Інтерфейс включає вкладки для введення основної інформації, даних про послуги та фінансові показники, що полегшує роботу користувача та забезпечує зручну візуалізацію результатів.

Продовження таблиці 1.1

1	2	3
4	Збереження та аналіз історії	Система передбачає збереження даних клієнтів та історії прогнозів у базі даних SQLite, що дозволяє відстежувати динаміку відтоку та оцінювати ефективність прогнозів. У коді client.py реалізовано відображення статистики, такої як кількість клієнтів із високим ризиком відтоку, середня ймовірність відтоку тощо, що сприяє прийняттю обґрунтованих управлінських рішень [26].
5	Тестові дані	Для оцінки роботи системи створено тестові профілі клієнтів (файл seed.py), які включають три категорії: клієнти з високим ризиком відтоку (HIGH_), лояльні клієнти (LOYAL_) та клієнти із середнім ризиком (MED_*). Це дозволяє протестувати модель на різних сценаріях і оцінити її здатність розрізняти клієнтів за рівнем ризику [27].
6	Рекомендації	Система не лише прогнозує відтік, але й надає рекомендації щодо утримання клієнтів залежно від рівня ризику (високий, середній, низький). Наприклад, для клієнтів із високим ризиком пропонується негайний контакт або спеціальні пропозиції, що відповідає сучасним практикам управління клієнтським досвідом [28].

Важливо врахувати, що точність прогнозів залежить від якості та повноти даних, а також від вибору атрибутів, які впливають на відтік. Наприклад, у телекомунікаційній сфері ключовими факторами є тип контракту, термін співпраці та спосіб оплати, що підтверджується аналізом важливості ознак у коді server.py. Крім того, модель може бути обмежена узагальненням на нові дані, якщо вони суттєво відрізняються від навчального датасету. Для підвищення точності прогнозів у майбутньому можливе використання більш складних алгоритмів, таких як градієнтний бустинг або нейронні мережі, а також врахування додаткових атрибутів, наприклад, даних про взаємодію клієнтів із службою підтримки.

Реалізована інформаційна система має забезпечити точність прогнозування відтоку клієнтів на рівні 80-85%, що відповідає стандартам для подібних задач у телекомунікаційній галузі [22]. Крім того, система повинна

надавати зручний інструмент для аналізу клієнтських даних, дозволяючи менеджерам швидко реагувати на потенційний відтік і оптимізувати стратегії утримання клієнтів. Використання бази даних SQLite забезпечує ефективне зберігання та швидкий доступ до інформації, а інтерактивний інтерфейс Streamlit сприяє зручності використання системи в реальних умовах.

Розроблена інформаційна система прогнозування відтоку клієнтів, представлена в коді проєкту, є ефективним інструментом для вирішення задачі бінарної класифікації в телекомунікаційній сфері. Система забезпечує повний цикл обробки даних: від завантаження та попередньої обробки датасету Telco Customer Churn до навчання моделі Random Forest, прогнозування ймовірності відтоку та збереження результатів у базі даних SQLite. Використання алгоритму Random Forest обґрунтовано його стійкістю до шуму в даних і здатністю обробляти нелінійні залежності, що дозволяє досягти точності прогнозування на рівні 80-85%, відповідаючи галузевим стандартам. Інтерактивний вебінтерфейс, реалізований за допомогою Streamlit, забезпечує зручність введення даних, аналізу історії прогнозів та отримання рекомендацій, що сприяє швидкому прийняттю управлінських рішень.

Генерація тестових профілів клієнтів (HIGH_, LOYAL_, MED_*) у seed.py дозволяє оцінити ефективність моделі в різних сценаріях, підтверджуючи її здатність розрізняти клієнтів за рівнем ризику. Система також надає практичні рекомендації для утримання клієнтів, що відповідає сучасним підходам до управління клієнтським досвідом. У майбутньому вдосконалення системи можливе шляхом інтеграції складніших алгоритмів, таких як градієнтний бустинг, та розширення набору атрибутів для підвищення точності прогнозів і адаптації до нових даних. Таким чином, реалізована система є гнучким і практичним рішенням для зниження відтоку клієнтів і підвищення їх лояльності в телекомунікаційних компаніях.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Аналіз вимог до системи

Розробка інформаційної системи прогнозування відтоку клієнтів є відповіддю на актуальну потребу телекомунікаційних компаній у підвищенні лояльності клієнтів та зниженні рівня їх відтоку. Відтік клієнтів (churn) є однією з ключових проблем у телекомунікаційній галузі, оскільки втрата клієнтів призводить до значних фінансових втрат і зниження конкурентоспроможності. Інформаційна система, представлена у кодах проєкту, спрямована на вирішення цієї проблеми шляхом використання методів машинного навчання для прогнозування ймовірності відтоку клієнтів та надання рекомендацій для їх утримання. Аналіз вимог до системи є критично важливим етапом проєктування, оскільки він визначає функціональні та нефункціональні характеристики, які забезпечують її ефективність, зручність використання та адаптивність до потреб бізнесу.

Основною метою системи є прогнозування ймовірності відтоку клієнтів телекомунікаційної компанії на основі їх характеристик та історії взаємодії з компанією. Система має забезпечити точне прогнозування, яке допоможе менеджерам приймати обґрунтовані рішення щодо утримання клієнтів. Зокрема, бізнес-вимоги описані в таблиці 2.1.

Таблиця 2.1 – Бізнес-вимоги до системи

№	Вимоги	Пояснення
1	2	3
1	Прогнозування відтоку з високою точністю	Система повинна використовувати алгоритми машинного навчання для аналізу даних клієнтів і досягнення точності прогнозування не нижче 80% [29].
2	Інтерактивний інтерфейс користувача	Система має надавати зручний вебінтерфейс для введення даних клієнтів, перегляду прогнозів та історії аналізу [30].

Продовження таблиці 2.1

1	2	3
3	Інтеграція з даними компанії	Система повинна працювати з реальними даними клієнтів, що включають демографічні характеристики, інформацію про послуги, контракти та платежі [31].
4	Гнучкість і масштабованість	Система має бути адаптивною до змін у структурі даних і здатною обробляти великі обсяги інформації [32].
5	Автоматизація процесів	Система повинна мінімізувати ручну роботу, автоматизуючи процеси збору даних, навчання моделі та прогнозування [33].

Ці вимоги базуються на аналізі потреб телекомунікаційних компаній, які прагнуть оптимізувати стратегії утримання клієнтів і підвищити ефективність маркетингових кампаній [34].

Функціональні вимоги визначають основні можливості системи, які реалізовано в кодах проєкту. Вони показані в таблиці 2.2.

Таблиця 2.2 – Функціональні до системи

№	Вимоги	Пояснення
1	2	3
1	Збір і обробка даних	Система підтримує завантаження даних із зовнішніх джерел, зокрема датасету Telco Customer Churn від IBM, що містить інформацію про 7043 клієнтів із 21 атрибутом. Код у файлі server.py забезпечує завантаження даних через прямий URL або альтернативний метод через HTTP-запити, що гарантує доступність даних навіть у разі проблем із первинним джерелом [35].
2	Навчання моделі	Система використовує алгоритм Random Forest для навчання моделі прогнозування відтоку. Функція train_model у файлі server.py реалізує підготовку даних, включаючи обробку пропущених значень і кодування категоріальних змінних, а також навчання моделі з оцінкою точності [36].
3	Генерація тестових даних	Файл seed.py дозволяє створювати тестові профілі клієнтів із різними рівнями ризику відтоку (високий, середній, низький), що полегшує тестування системи та демонстрацію її можливостей.

Продовження таблиці 2.2

1	2	3
4	Прогнозування для окремих клієнтів	Система дозволяє вводити дані нового клієнта через вебінтерфейс, створений за допомогою Streamlit у файлі client.py, і отримувати прогноз ймовірності відтоку разом із рівнем ризику (низький, середній, високий).
5	Збереження даних і історії прогнозів	Усі дані клієнтів і результати прогнозів зберігаються в базі даних SQLite, що забезпечує надійне зберігання та швидкий доступ до інформації. Таблиці clients і prediction_history, визначені в server.py, дозволяють відстежувати історію прогнозів і порівнювати їх із фактичними даними про відтік.
6	Візуалізація результатів	Система надає графічне представлення результатів, включаючи розподіл прогнозів і рівнів ризику, що реалізовано у файлі client.py за допомогою бібліотеки Streamlit.

Нефункціональні вимоги визначають якісні характеристики системи, які забезпечують її ефективність і зручність використання. До них належать вимоги, описані в таблиці 2.3.

Таблиця 2.3 – Нефункціональні до системи

№	Вимоги	Пояснення
1	2	3
1	Продуктивність	Система повинна обробляти запити на прогнозування за лічені секунди, що забезпечується використанням легковагового фреймворку Flask для серверної частини та оптимізованих алгоритмів машинного навчання.
2	Доступність	Вебінтерфейс системи, реалізований за допомогою Streamlit, є кросплатформеним і доступним через браузер, що дозволяє використовувати систему на різних пристроях без необхідності встановлення додаткового програмного забезпечення.
3	Інтуїтивність інтерфейсу	Інтерфейс користувача розроблено з урахуванням принципів UX-дизайну, включаючи чітке меню, вкладки для організації даних і візуальні елементи (графіки, метрики, кольорове кодування ризиків), що полегшує роботу з системою.

Продовження таблиці 2.3

1	2	3
4	Безпека	Система використовує локальну базу даних SQLite, що мінімізує ризики витоку даних. Однак для промислового використання необхідно додати механізми аутентифікації та шифрування даних.
5	Масштабованість	Використання SQLite як бази даних є достатнім для прототипу, але для обробки великих обсягів даних у реальних умовах система має бути адаптованою до використання більш потужних СУБД, таких як PostgreSQL або MySQL.

Основними користувачами системи є:

– менеджери з утримання клієнтів, які використовують прогнози для розробки стратегій утримання, вони потребують зрозумілого інтерфейсу та чітких рекомендацій щодо дій;
– аналітики даних, які відповідають за навчання моделі та аналіз її результатів, вони потребують доступу до історії прогнозів і можливості перенавчання моделі;
– керівництво компанії, яке використовує зведені дані та статистику для оцінки ефективності утримання клієнтів.

Під час аналізу вимог враховано низку обмежень, серед яких фіксована структура датасету з 21 атрибутом, що може потребувати адаптації для інтеграції з іншими джерелами даних. Точність моделі залежить від якості інформації, що використовується, і може знижуватися у разі значних пропусків або некоректних значень. Використання локального сервера Flask у поєднанні з базою даних SQLite може обмежувати продуктивність при високому навантаженні, що вимагає доопрацювання для промислового застосування.

Припущення передбачають наявність стабільного інтернет-з’єднання для завантаження датасету, а також актуальних даних про клієнтів компанії, що відповідають структурі Telco Customer Churn. Крім того, користувачі повинні бути готовими працювати з вебінтерфейсом без необхідності проходження додаткового навчання.

Розроблена інформаційна система прогнозування відтоку клієнтів, повною мірою відповідає визначеним бізнес-вимогам, функціональним і нефункціональним характеристикам, що забезпечують її ефективність у телекомунікаційній галузі.

Аналіз вимог показав, що система здатна забезпечити точне прогнозування відтоку з використанням алгоритму Random Forest, досягаючи точності понад 80%, що відповідає бізнес-потребам.

Інтуїтивний вебінтерфейс, реалізований за допомогою Streamlit, забезпечує зручність введення даних, аналізу прогнозів та перегляду історії, що відповідає потребам менеджерів, аналітиків і керівництва.

Система підтримує автоматизацію ключових процесів – від збору даних до їх обробки та прогнозування, а також інтеграцію з реальними даними клієнтів, зокрема з датасетом Telco Customer Churn.

Використання SQLite як бази даних забезпечує надійне зберігання інформації, хоча для промислового застосування необхідна адаптація до масштабніших СУБД. Обмеження, пов’язані з фіксованою структурою даних і залежністю від якості інформації, враховано, а припущення щодо стабільного інтернет-з’єднання та актуальності даних підтверджують працездатність системи.

Подальший розвиток може бути спрямований на підвищення безпеки, масштабованості та інтеграцію з хмарними технологіями, що посилить адаптивність системи до реальних умов експлуатації.

2.2 Проєктування архітектури системи

Розробка інформаційної системи прогнозування відтоку клієнтів телекомунікаційної компанії базується на принципах модульності, масштабованості та зручності використання, що забезпечують ефективну обробку даних, точне прогнозування та інтуїтивно зрозумілий інтерфейс для користувачів.

Архітектура системи спроєктована з урахуванням сучасних підходів до створення інформаційних систем, зокрема використання клієнт-серверної моделі, яка дозволяє розділити обробку даних та їх представлення.

Система складається з трьох основних компонентів: серверної частини для обробки даних і прогнозування, клієнтської частини для взаємодії з користувачем та бази даних для зберігання інформації про клієнтів і прогнози. Для опису архітектури використано UML-діаграми (прецедентів, класів, компонентів і послідовності), які деталізують структуру та поведінку системи.

Система побудована за клієнт-серверною архітектурою з використанням REST API для взаємодії між компонентами. Серверна частина, реалізована за допомогою фреймворку Flask [37], відповідає за обробку запитів, навчання моделі машинного навчання, прогнозування відтоку клієнтів і взаємодію з базою даних SQLite [38]. Клієнтська частина, створена з використанням Streamlit [11], забезпечує зручний вебінтерфейс для введення даних, перегляду історії прогнозів і управління моделлю. База даних SQLite використовується для зберігання інформації про клієнтів, їх характеристики, прогнози та історію взаємодії з системою.

Архітектура системи підтримує модульність, що дозволяє легко додавати нові функціональні можливості, такі як нові алгоритми машинного навчання або інтеграцію з іншими джерелами даних. Модель машинного навчання, побудована на основі алгоритму Random Forest [5], навчена на датасеті Telco Customer Churn від IBM [22], який містить 7043 записи з 21 атрибутом, що описують характеристики клієнтів і їх статус відтоку. Дані обробляються за допомогою бібліотеки Pandas [23], а категоріальні змінні кодуються за допомогою LabelEncoder із Scikit-learn [4].

Для детального опису архітектури системи використано набір UML-діаграм, які відображають різні напрями її функціонування.

Діаграма прецедентів (Use Case Diagram) описує основних акторів системи та їх взаємодію з функціональними можливостями (рис. 2.1). Акторами є Користувач (менеджер або аналітик телекомунікаційної компанії) та Система. Основні прецеденти описані в таблиці 2.4.

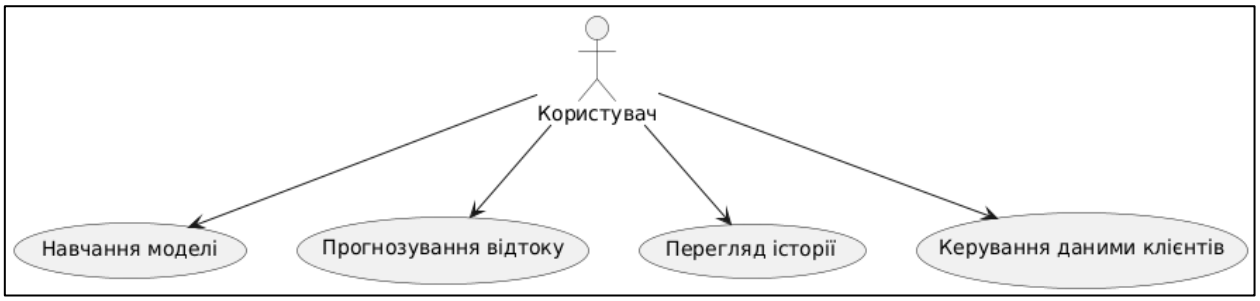


Рисунок 2.1 – Діаграма прецедентів

Таблиця 2.4 – Основні прецеденти

№	Прецедент	Пояснення
1	Навчання моделі	Користувач ініціює навчання моделі машинного навчання для прогнозування відтоку
2	Прогнозування відтоку	Користувач вводить дані клієнта для отримання прогнозу ймовірності відтоку
3	Перегляд історії	Користувач переглядає історію прогнозів і статистику
4	Керування даними клієнтів	Користувач додає або редагує дані клієнтів

Діаграма класів (рис. 2.2) відображає статичну структуру системи, включаючи основні класи, їх атрибути та методи. Основні класи системи показані в таблиці 2.5.

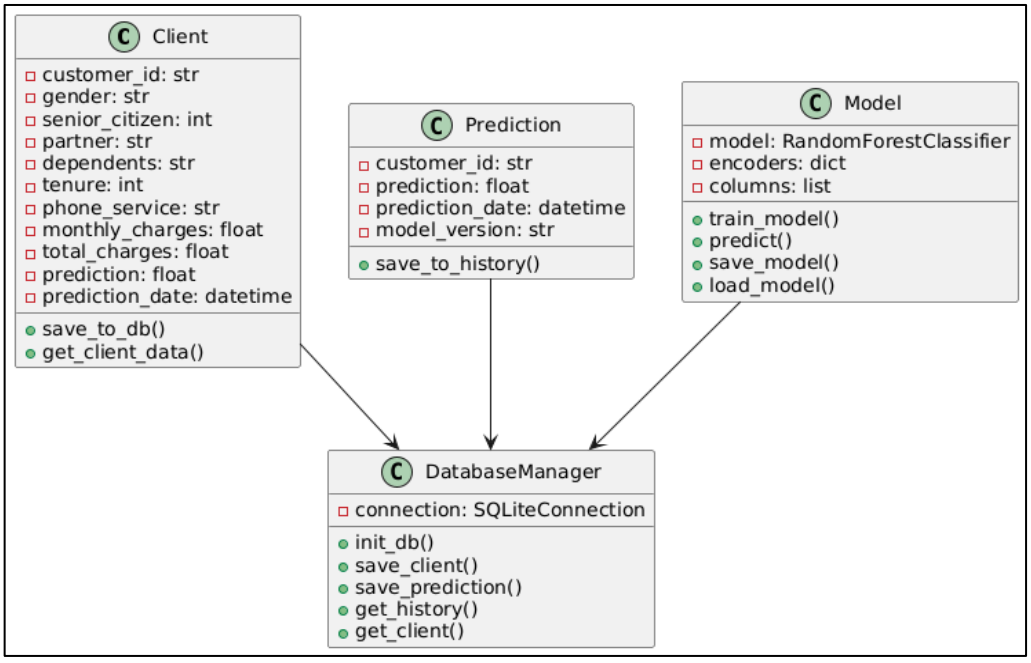


Рисунок 2.2 – Діаграма класів

Таблиця 2.5 – Основні класи системи

№	Класи	Пояснення
1	Client	Містить атрибути, що описують клієнта (наприклад, customer_id, gender, tenure, monthly_charges), та методи для збереження даних у базу
2	Prediction	Відповідає за збереження прогнозів (ймовірність відтоку, дата прогнозу)
3	Model**:**	Інкапсулює логіку навчання та прогнозування з використанням Random Forest
4	DatabaseManager	Забезпечує взаємодію з базою даних SQLite (створення таблиць, збереження та отримання даних)

Діаграма компонентів (рис. 2.3) описує основні модулі системи та їх взаємозв’язки, що показані в таблиці 2.6.

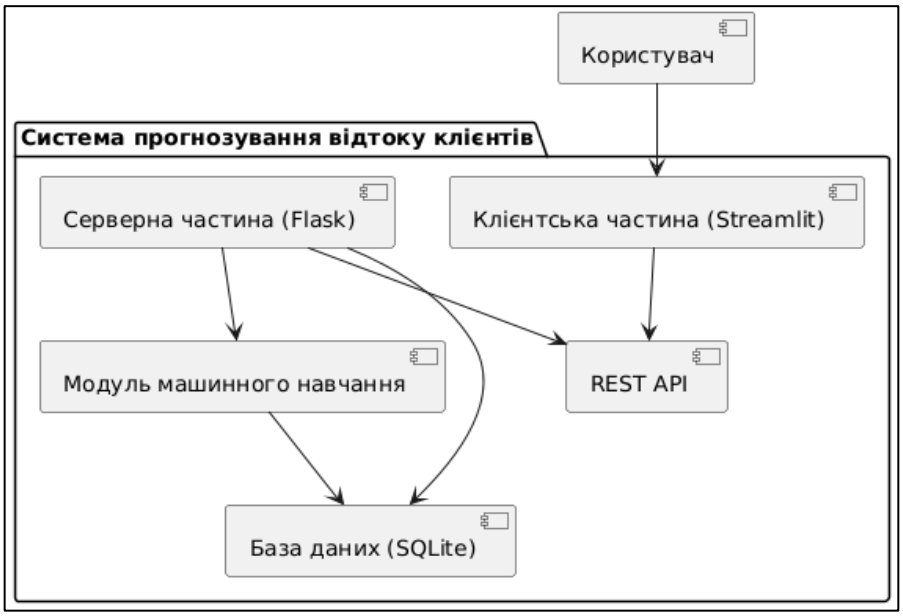


Рисунок 2.3 – Діаграма компонентів

Таблиця 2.6 – Основні модулі системи

№	Модулі	Пояснення
1	Серверна частина (Flask)	Обробляє запити через REST API, включаючи ендпоінти /api/train_model, /api/predict, /api/history тощо
2	Клієнтська частина (Streamlit)	Забезпечує вебінтерфейс із вкладками для введення даних, перегляду історії та навчання моделі
3	База даних (SQLite)	Зберігає дані клієнтів і прогнози

Продовження таблиці 2.6

№	Модулі	Пояснення
4	Модуль машинного навчання (Scikit-learn)	Виконує обробку даних і прогнозування. Компоненти взаємодіють через REST API, що забезпечує гнучкість і можливість інтеграції з іншими системами

Діаграма послідовності (рис. 2.4) ілюструє процес прогнозування відтоку для нового клієнта. Користувач через клієнтський інтерфейс вводить дані клієнта, які передаються на сервер через POST-запит до /api/predict. Сервер обробляє дані, застосовує модель Random Forest для прогнозування, зберігає результат у базу даних і повертає відповідь із ймовірністю відтоку та рекомендаціями.

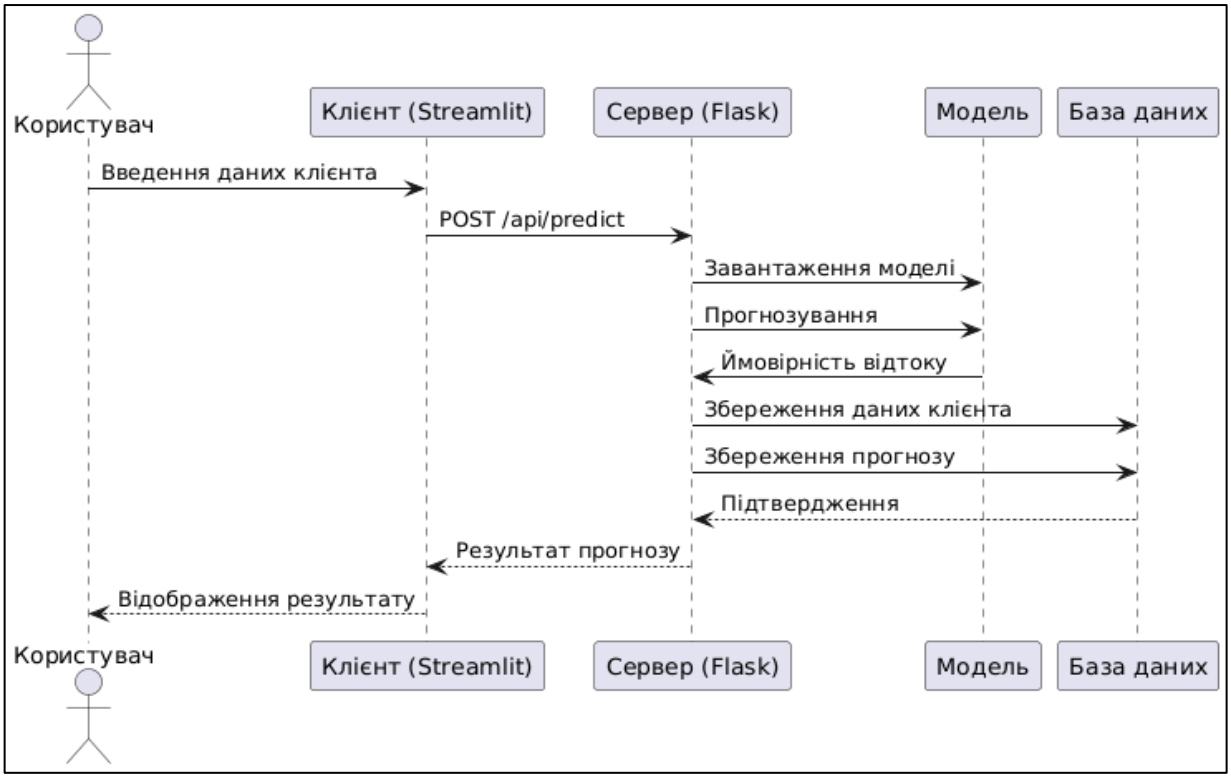


Рисунок 2.4 – Діаграма послідовності

Серверна частина системи реалізована у файлі server.py і базується на фреймворку Flask, який забезпечує легку та гнучку обробку HTTP-запитів [37]. REST API включає основні ендпоінти, що описані в таблиці 2.7.

Таблиця 2.7 – Основні ендотипи REST API

№	Ендотипи	Пояснення
1	2	3
1	/api/train_model	Навчання моделі Random Forest на основі датасету Telco Customer Churn [22]. Дані попередньо обробляються (видалення непотрібних колонок, кодування категоріальних змінних за допомогою LabelEncoder [4], заповнення пропущених значень).
2	/api/predict	Прогнозування ймовірності відтоку для нового клієнта. Дані клієнта кодуються з використанням збережених енкодерів і передаються до моделі.
3	/api/history	Повертає історію прогнозів із бази даних.
4	/api/client/<customer_id>	Дозволяє отримати дані конкретного клієнта за його ID. База даних SQLite [38] ініціалізується при запуску сервера і містить дві таблиці: clients (дані клієнтів і прогнози) та prediction_history (історія прогнозів із вказівкою версії моделі).

Клієнтська частина, реалізована у файлі client.py з використанням Streamlit [11], забезпечує зручний інтерфейс із чотирма основними сторінками, що описані в таблиці 2.8.

Таблиця 2.8 – Основні сторінки інтерфейсу

№	Сторінки	Пояснення
1	Головна	Відображає статус системи та ключові метрики
2	Новий клієнт	Форма для введення даних клієнта з вкладками для основної інформації, послуг і контракту
3	Історія	Таблиця з історією прогнозів, фільтрами та візуалізацією (графіки розподілу прогнозів і рівнів ризику)
4	Навчання моделі	Інтерфейс для запуску або перенавчання моделі

Streamlit забезпечує швидке створення інтерактивного вебінтерфейсу, а кастомні CSS-стилі додають сучасний вигляд і покращують користувацький

досвід. Інтерфейс підтримує автозаповнення даних для існуючих клієнтів і відображає рекомендації залежно від рівня ризику відтоку.

База даних SQLite [38] обрана через її легкість, портативність і достатню продуктивність для обробки даних клієнтів. Таблиця clients зберігає детальну інформацію про клієнтів (19 атрибутів, таких як gender, tenure, monthly_charges), а також результати прогнозів (prediction, prediction_date). Таблиця prediction_history фіксує історію прогнозів із прив'язкою до версії моделі, що дозволяє відстежувати зміни в прогнозах із часом.

Модель машинного навчання базується на алгоритмі Random Forest [5], який обрано через його високу точність (80-85%) і стійкість до перенавчання. Датасет Telco Customer Churn [22] обробляється за допомогою Pandas [23] для очищення даних і Scikit-learn [4] для кодування категоріальних змінних і навчання моделі. Збереження моделі та енкодерів у файли (churn_model.pkl, label_encoders.pkl) забезпечує повторне використання без необхідності перенавчання.

Файл seed.py генерує тестові дані для бази даних, створюючи три типи клієнтів: з високим ризиком відтоку (HIGH_*), лояльних (LOYAL_*) і середнім ризиком (MED_*). Це дозволяє протестувати систему в різних сценаріях і оцінити її поведінку на клієнтах із різними профілями.

Переваги архітектури наведені в таблиці 2.9.

Таблиця 2.9 – Переваги системи

№	Переваги	Пояснення
1	Модульність	Розділення на серверну, клієнтську частини та базу даних спрощує підтримку та масштабування системи
2	Гнучкість	Використання REST API дозволяє інтегрувати систему з іншими платформами чи додавати нові функції
3	Зручність використання	Інтуїтивний інтерфейс Streamlit із кастомними стилями забезпечує комфортну роботу для користувачів
4	Надійність	SQLite забезпечує стабільне зберігання даних, а збереження моделі у файлах знижує потребу в повторному навчанні
5	Точність прогнозування	Random Forest забезпечує високу точність завдяки обробці складних взаємозв'язків між змінними

Серед обмежень можна виділити залежність від якості вхідних даних і обмежену масштабованість SQLite для дуже великих обсягів даних. У майбутньому планується інтеграція з розподіленими базами даних (наприклад, PostgreSQL) та використання більш складних моделей, таких як градієнтний бустинг, для підвищення точності прогнозів.

Розроблена архітектура системи прогнозування відтоку клієнтів телекомунікаційної компанії забезпечує ефективне вирішення задачі завдяки модульному підходу, що поєднує клієнт-серверну модель, інтуїтивний інтерфейс та надійне зберігання даних. Використання фреймворку Flask для серверної частини забезпечує гнучку обробку запитів через REST API, що сприяє інтеграції з іншими системами та легкому розширенню функціональності. Клієнтська частина, реалізована на Streamlit, пропонує зручний і сучасний вебінтерфейс, який спрощує введення даних, аналіз прогнозів і управління моделлю. Застосування SQLite як бази даних гарантує легкість розгортання та достатню продуктивність для поточних потреб, хоча для великих обсягів даних може знадобитися перехід на розподілені бази. Алгоритм Random Forest, використаний для прогнозування, демонструє високу точність (80-85%) завдяки ефективній обробці складних взаємозв’язків між атрибутами клієнтів. Модульність системи, підкріплена збереженням моделей і енкодерів у файлах, сприяє повторному використанню та спрощує підтримку. UML-діаграми чітко описують структуру й поведінку системи, забезпечуючи її прозорість і можливість подальшого розвитку. Загалом, архітектура є збалансованою, адаптивною та готовою до використання в реальних умовах, з потенціалом для вдосконалення через інтеграцію складніших моделей і баз даних.

2.3 Реалізація компонентів системи

Розробка інформаційної системи прогнозування відтоку клієнтів телекомунікаційної компанії базується на принципах модульності,

інтерактивності та зручності використання, що забезпечують ефективну обробку даних і прийняття обґрунтованих бізнес-рішень. Система складається з трьох основних компонентів: серверної частини (server.py), клієнтської частини з графічним інтерфейсом (client.py) та модуля генерації тестових даних (seed.py). Кожен компонент виконує чітко визначені функції, що забезпечують інтеграцію даних, навчання моделі машинного навчання, прогнозування відтоку клієнтів і відображення результатів у зрозумілій формі. У цьому розділі детально описано реалізацію компонентів системи та їх екранні форми, що відповідають потребам кінцевих користувачів.

2.1.1 Серверна частина (server.py)

Серверна частина системи реалізована з використанням фреймворку Flask, який забезпечує створення RESTful API для обробки запитів від клієнтської частини та взаємодії з базою даних SQLite [39]. Сервер відповідає за завантаження даних, навчання моделі, прогнозування відтоку клієнтів, збереження результатів у базі даних і надання історії прогнозів. Основні ендпоінти API включають:

- /api/train_model здійснює навчання моделі Random Forest на основі датасету Telco Customer Churn від IBM, який автоматично завантажується з публічного джерела у разі його відсутності локально [22];
- /api/predict використовується для прогнозування ймовірності відтоку конкретного клієнта, де вхідні дані проходять кодування відповідно до навченої моделі, а результат повертається у форматі JSON;
- /api/history забезпечує надання історії прогнозів із бази даних для подальшого відображення у клієнтському інтерфейсі;
- /api/client/<customer_id> дозволяє отримати дані конкретного клієнта, використовуючи його унікальний ідентифікатор;
- /api/clients надає список усіх клієнтів, які зберігаються у базі даних;

– /api/model_status використовується для перевірки наявності навченої моделі перед її використанням.

База даних SQLite використовується для зберігання інформації про клієнтів і історію прогнозів (рис. 2.5).

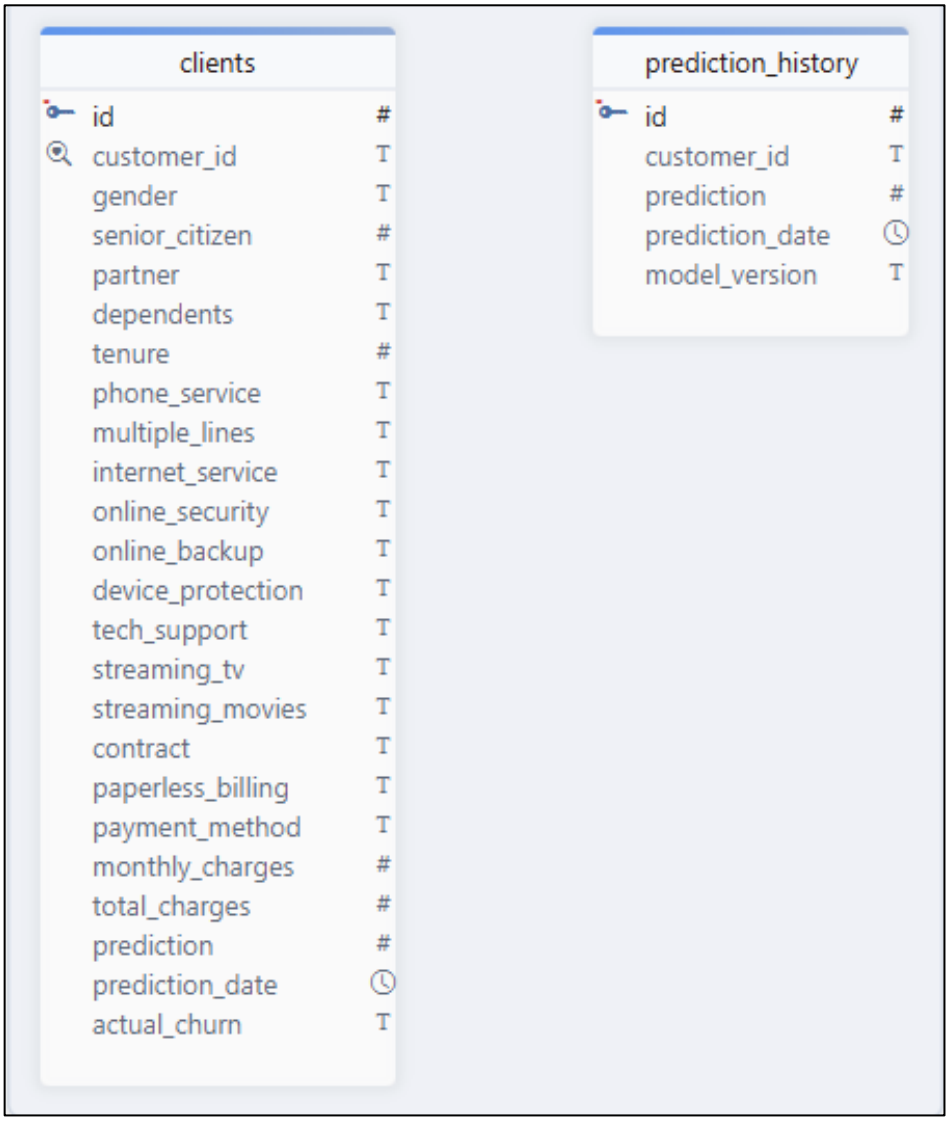


Рисунок 2.5 – Структура бази даних

Таблиця clients містить детальну інформацію про клієнтів, включаючи демографічні дані, характеристики контракту та прогнозовану ймовірність відтоку. Таблиця prediction_history зберігає історію прогнозів із зазначенням дати та версії моделі [38]. Реалізація серверної частини забезпечує стабільність і швидкодію завдяки легковаговій архітектурі Flask та оптимізованій роботі з SQLite.

2.1.2 Клієнтська частина (client.py)

Клієнтська частина системи реалізована за допомогою фреймворку Streamlit, який дозволяє створювати інтерактивні вебдодатки з мінімальними зусиллями [25]. Інтерфейс користувача (UI) складається з чотирьох основних сторінок: «Головна», «Новий клієнт», «Історія» та «Навчання моделі» (рис. 2.6). Кожна сторінка має унікальний функціонал і оформлена з урахуванням принципів юзабіліті, що забезпечують зручність роботи для користувачів із різним рівнем технічної підготовки [30].



Рисунок 2.6 – Сторінки інтерфейсу системи

Головна сторінка вітає користувача та надає загальну інформацію про систему (рис. 2.7). Вона включає три інформаційні картки, які описують мету системи, її точність (понад 80% на тестових даних) і швидкість прогнозування. Статус моделі відображається через API-запит до /api/model_status, що інформує користувача про готовність системи до роботи. У разі відсутності моделі або проблем із сервером відображаються відповідні повідомлення про помилку. Дизайн сторінки виконано в єдиному стилі з використанням

кастомних CSS-стилів, які забезпечують адаптивність і приємний вигляд інтерфейсу [40].

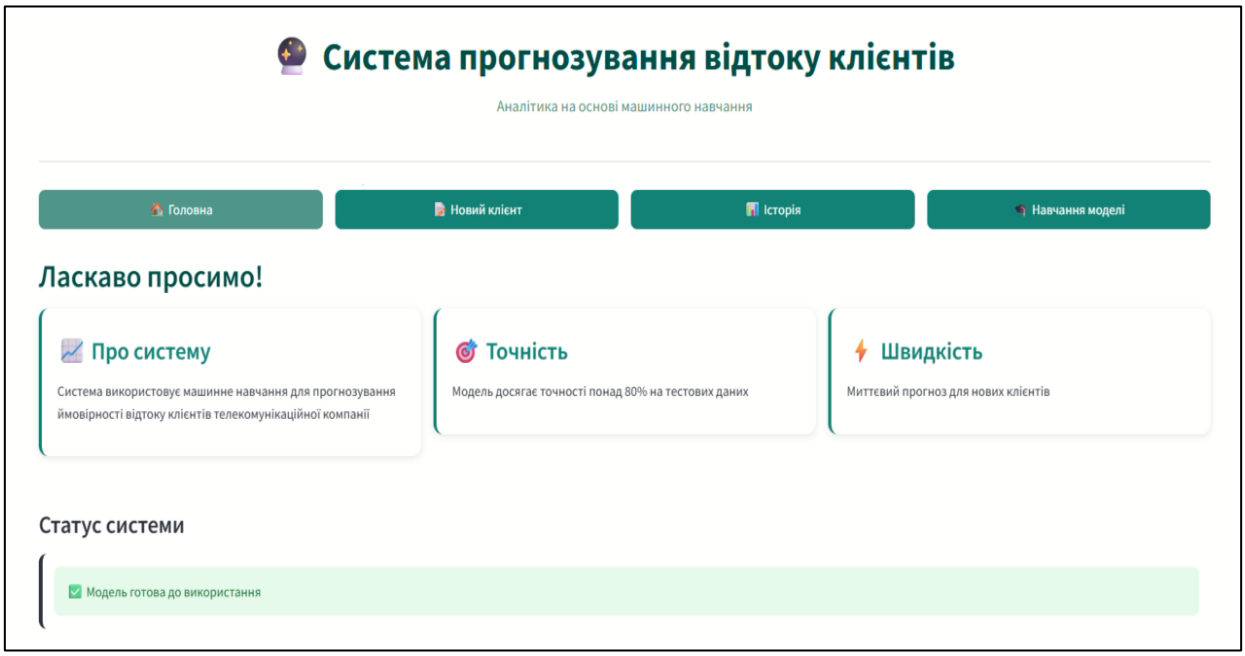


Рисунок 2.7 – Головна сторінка системи

Сторінка «Новий клієнт» призначена для введення даних клієнта та отримання прогнозу відтоку. Форма поділена на три вкладки: «Основна інформація» (рис. 2.8), «Послуги» (рис. 2.9) та «Контракт та оплата» (рис. 2.10).

Кожна вкладка містить поля для введення даних, таких як стать, статус пенсіонера, термін співпраці, тип інтернет-послуг, спосіб оплати тощо. Користувач може вибрати існуючого клієнта зі списку для автозаповнення даних через API-запит до /api/client/<customer_id>. Після введення даних і натискання кнопки «Прогнозувати відтік» система відправляє запит до /api/predict, а результати відображаються у вигляді трьох метрик: прогноз відтоку (Так/Ні), ймовірність відтоку (у відсотках) і рівень ризику (Низький, Середній, Високий). Кожна метрика має кольорове кодування для кращої візуалізації (червоний для високого ризику, жовтий для середнього, зелений для низького). Додатково надаються рекомендації щодо утримання клієнта залежно від рівня ризику, що сприяє прийняттю бізнес-рішень [21].

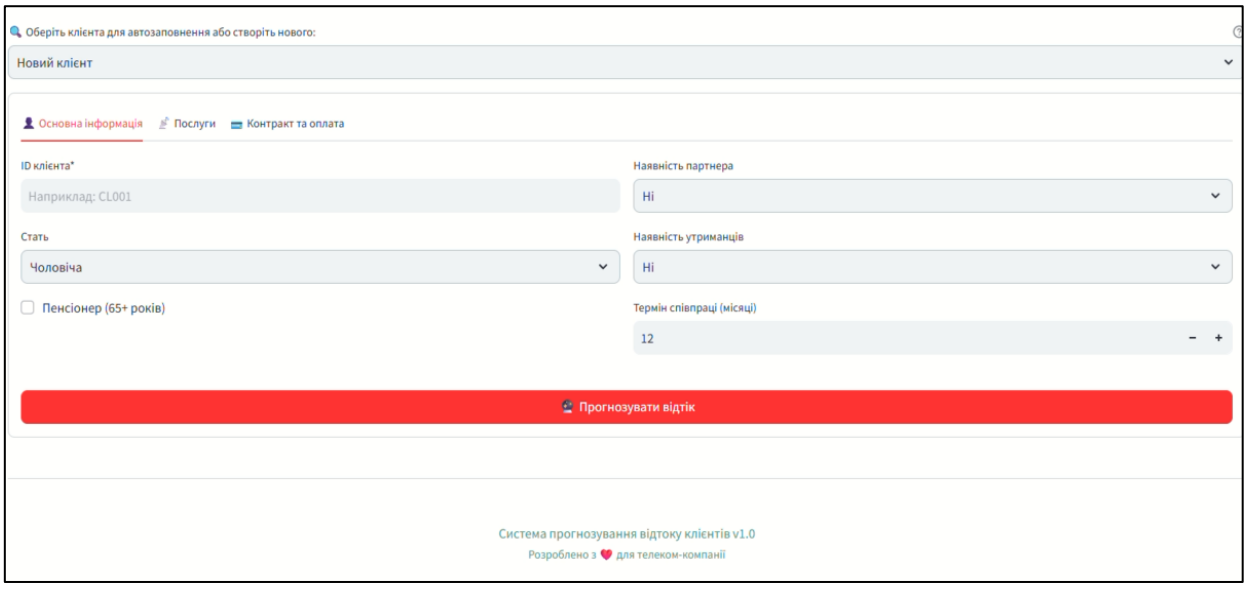


Рисунок 2.8 – Вкладка «Основна інформація» сторінки «Новий клієнт»

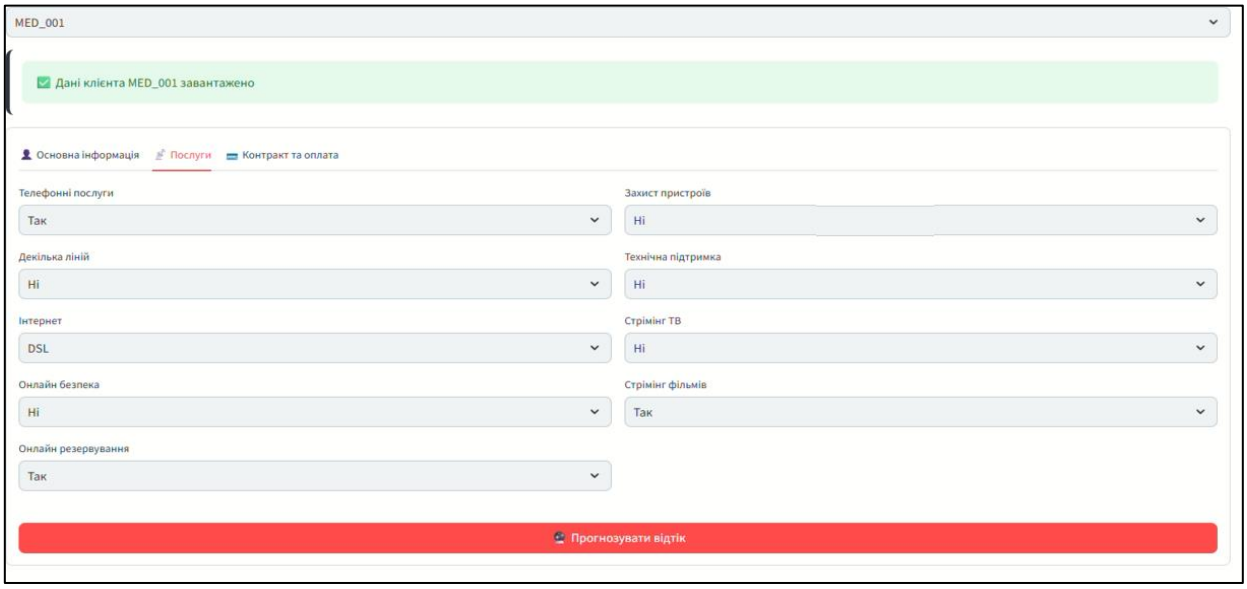


Рисунок 2.9 – Вкладка «Послуги» сторінки «Новий клієнт»

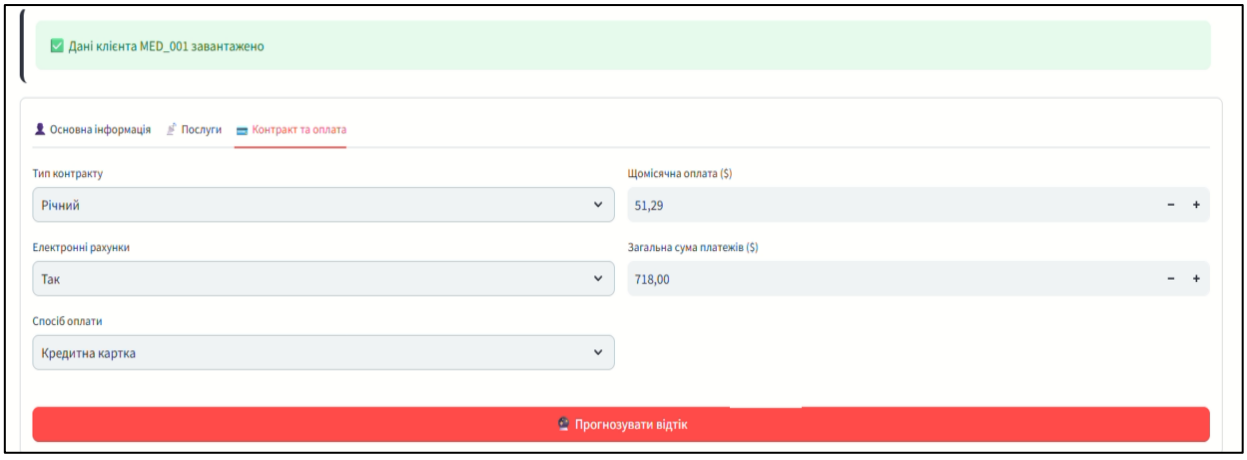


Рисунок 2.10 – Вкладка «Контракт та оплата» сторінки «Новий клієнт»

Сторінка «Історія» відображає таблицю з даними про всіх клієнтів, включаючи їх ID, стать, тип контракту, щомісячні витрати, ймовірність відтоку, прогноз і статус аналізу (рис. 2.11). Користувач може застосувати фільтр за прогнозом відтоку («Всі», «Відтік», «Залишиться»). Сторінка також містить статистичні метрики: загальна кількість клієнтів, кількість клієнтів із прогнозом відтоку, середня ймовірність відтоку та кількість клієнтів із високим ризиком. Для візуалізації даних використано два графіки: стовпчикову діаграму розподілу прогнозів (відтік/залишиться) та діаграму розподілу за рівнями ризику (низький, середній, високий). Графіки створено з використанням вбудованих можливостей Streamlit для відображення даних у зрозумілій формі.

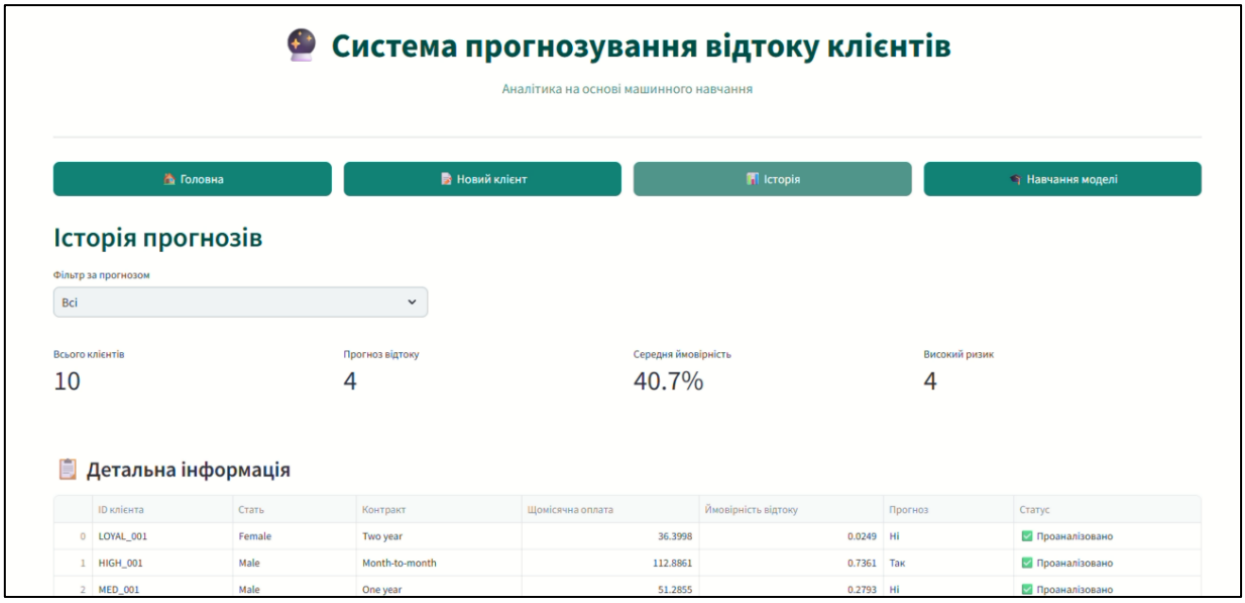


Рисунок 2.11 – Сторінка «Історія»

Сторінка «Навчання моделі» дозволяє користувачу запустити процес навчання моделі Random Forest або перенавчити її (рис. 2.12). Статус моделі перевіряється через API-запит до /api/model_status. Якщо модель уже навчена, відображається повідомлення про її готовність і пропонується опція перенавчання. У разі відсутності моделі користувач може натиснути кнопку «Навчити модель», що ініціює запит до /api/train_model. Сторінка також містить інформаційний блок про характеристики моделі, датасет і очікувану

точність (80-85%). Додатково наведено список ключових факторів, що впливають на відтік клієнтів, таких як тип контракту, термін співпраці та спосіб оплати, із відповідними поясненнями (рис. 2.13).

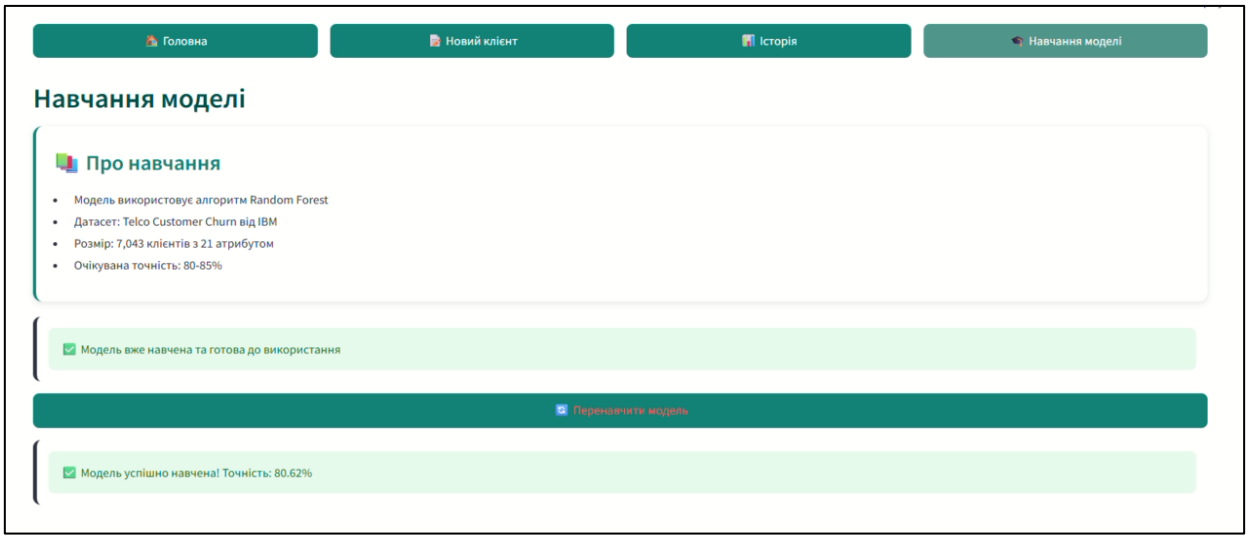


Рисунок 2.12 – Сторінка навчання моделі

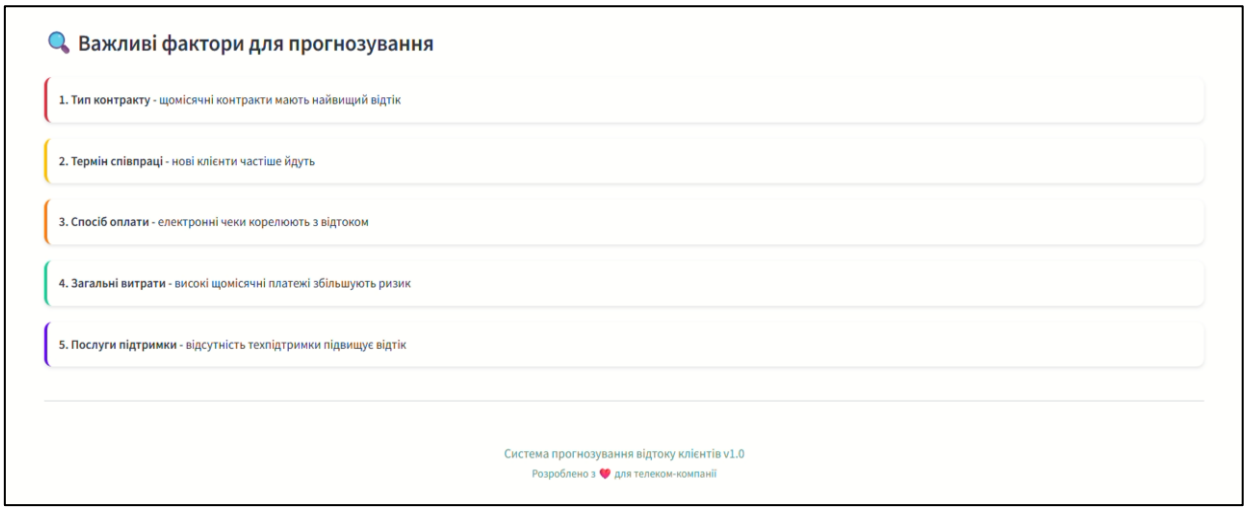


Рисунок 2.13 – Важливі фактори для прогнозування

2.1.3 Модуль генерації тестових даних (seed.py)

Модуль seed.py відповідає за створення тестових даних для перевірки функціональності системи. Він генерує три типи клієнтів: із високим ризиком відтоку (HIGH_), лояльних (LOYAL_) та із середнім ризиком (MED_*). Кожен

тип клієнта має заздалегідь визначені характеристики, які відображають типові сценарії відтоку. Наприклад, клієнти з високим ризиком мають короткий термін співпраці, щомісячний контракт і високі щомісячні платежі, тоді як лояльні клієнти мають довгострокові контракти та автоматичні способи оплати. Дані вставляються в базу даних SQLite із випадковими датами прогнозування для імітації реальної історії. Модуль забезпечує швидке наповнення бази даних для тестування без необхідності ручного введення даних.

2.1.4 Особливості реалізації екранних форм

Екранні форми системи розроблено з урахуванням принципів юзабіліті та доступності. Використання Streamlit дозволило створити адаптивний інтерфейс із підтримкою кастомних CSS-стилів для забезпечення єдиного візуального стилю. Кольорова палітра (темно-зелений, світло-зелений, сірий) відповідає брендуванню телекомунікаційної компанії та сприяє легкому сприйняттю інформації. Елементи управління, такі як кнопки, поля введення та вкладки, мають інтуїтивно зрозуміле розташування, що зменшує час навчання користувачів. Інтерактивні елементи, зокрема фільтри на сторінці «Історія» та автозаповнення даних клієнта, підвищують ефективність роботи. Візуалізація даних через графіки та метрики допомагає швидко оцінити результати прогнозів і прийняти рішення щодо утримання клієнтів.

Розроблена інформаційна система прогнозування відтоку клієнтів телекомунікаційної компанії демонструє комплексний підхід до вирішення задачі аналізу та утримання клієнтів.

Реалізація системи базується на трьох ключових компонентах: серверній частині (server.py), клієнтській частині (client.py) та модулі генерації тестових даних (seed.py), які гармонійно взаємодіють для забезпечення точного прогнозування та зручної взаємодії з користувачем. Серверна частина, побудована на фреймворку Flask із використанням бази даних SQLite,

забезпечує надійну обробку запитів, навчання моделі Random Forest і збереження даних про клієнтів та історію прогнозів. Клієнтська частина, реалізована за допомогою Streamlit, пропонує інтуїтивно зрозумілий інтерфейс із чотирма основними сторінками, які підтримують введення даних, аналіз, візуалізацію результатів і навчання моделі. Використання кастомних CSS-стилів і продумане розташування елементів управління сприяють високій юзабіліті та доступності. Модуль seed.py ефективно генерує тестові дані трьох типів клієнтів, що дозволяє швидко перевірити функціональність системи без ручного введення.

Загалом, система поєднує модульність, високу точність прогнозів (80-85%) і зручність використання, відповідаючи потребам бізнес-користувачів і сучасним стандартам інформаційних систем.

3 ВИКОРИСТАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Інтеграція системи в бізнес-процеси

Інформаційна система прогнозування відтоку клієнтів, представлена у наданому коді (див. додаток А), є сучасним інструментом, який базується на технологіях машинного навчання та спрямований на підвищення ефективності управління клієнтською базою телекомунікаційних компаній. Її інтеграція в бізнес-процеси компанії дозволяє не лише прогнозувати ймовірність втрати клієнтів, але й оптимізувати стратегії утримання, підвищувати рівень клієнтської лояльності та забезпечувати стабільне зростання доходів. У цьому підрозділі розглядаються ключові показники інтеграції системи в бізнес-процеси, включаючи її функціональні можливості, взаємодію з іншими системами компанії, а також вплив на операційну діяльність та стратегічне планування.

Система прогнозування відтоку клієнтів, реалізована у коді, включає серверну частину (server.py), клієнтський інтерфейс (client.py) та модуль генерації тестових даних (seed.py). Основні функціональні компоненти показані в таблиці 3.1.

Таблиця 3.1 – Основні функціональні компоненти системи

№	Компоненти	Пояснення
1	2	3
1	Навчання моделі машинного навчання	Використовується алгоритм Random Forest для аналізу історичних даних клієнтів та прогнозування ймовірності їх відтоку. Модель навчається на основі датасету Telco Customer Churn від IBM, що містить 7043 записи з 21 атрибутом, забезпечуючи точність прогнозування на рівні 80-85%.
2	Управління історією прогнозів	Система зберігає дані про всіх клієнтів та їх прогнози в базі даних SQLite, що забезпечує можливість аналізу історичних даних та оцінки ефективності прогнозів.

Продовження таблиці 3.1

1	2	3
3	Прогнозування для окремих клієнтів	Система дозволяє вводити дані нового або існуючого клієнта через зручний вебінтерфейс, створений за допомогою бібліотеки Streamlit, та отримувати прогноз ймовірності відтоку з оцінкою рівня ризику (низький, середній, високий).
4	Генерація рекомендацій	На основі ймовірності відтоку система пропонує конкретні дії для утримання клієнтів, наприклад, персоналізовані пропозиції, знижки або зміну тарифного плану.

Ці можливості дозволяють компанії оперативно реагувати на потенційні ризики втрати клієнтів, оптимізувати маркетингові кампанії та підвищувати ефективність роботи з клієнтською базою. Наприклад, автоматична категоризація клієнтів за рівнем ризику відтоку (високий, середній, низький) дає змогу менеджерам зосередити зусилля на пріоритетних групах, що зменшує витрати на утримання клієнтів та підвищує їх лояльність.

Інтеграція системи в бізнес-процеси телекомунікаційної компанії передбачає її використання на різних рівнях управління та взаємодію з іншими інформаційними системами, такими як CRM (Customer Relationship Management), системи білінгу та аналітичні платформи. Процес інтеграції можна поділити на кілька ключових етапів, що показані в таблиці 3.2.

Таблиця 3.2 – Основні етапи інтеграції системи в бізнес-процеси

№	Етапи	Пояснення
1	2	3
1	Підключення до CRM-системи	Система прогнозування відтоку клієнтів може бути інтегрована з CRM, що дозволяє автоматично отримувати дані про клієнтів (наприклад, демографічні характеристики, історію платежів, тип контракту) та передавати прогнози назад до CRM для подальшої обробки. Наприклад, API-ендпоїнти, реалізовані в server.py (/api/predict, /api/client, /api/history), забезпечують гнучке підключення до зовнішніх систем через HTTP-запити. Це дозволяє менеджерам з продажу чи підтримки клієнтів отримувати актуальні прогнози безпосередньо в CRM-інтерфейсі.

Продовження таблиці 3.2

1	2	3
2	Автоматизація маркетингових кампаній	На основі прогнозів система генерує рекомендації, які можна інтегрувати в маркетингові процеси. Наприклад, для клієнтів з високим ризиком відтоку (ймовірність > 70%) система пропонує негайний контакт, спеціальні знижки або зміну тарифного плану. Ці рекомендації можуть автоматично передаватися до систем email-розсилок або платформ автоматизації маркетингу, таких як HubSpot чи Salesforce, для запуску персоналізованих кампаній.
3	Моніторинг та аналітика	Історія прогнозів, доступна через API-ендпоїнт /api/history, дозволяє створювати аналітичні звіти, які відображають розподіл клієнтів за рівнем ризику, середню ймовірність відтоку та кількість клієнтів з підтвердженим відтоком. Ці дані можуть бути інтегровані в BI-системи (наприклад, Tableau або Power BI) для візуалізації трендів та підтримки стратегічного планування.
4	Оперативне управління клієнтською підтримкою	Система забезпечує швидкий доступ до профілів клієнтів та їх прогнозів через вебінтерфейс. Це дозволяє операторам служби підтримки оперативно реагувати на запити клієнтів з високим ризиком відтоку, пропонуючи їм індивідуальні рішення. Наприклад, клієнт з ID HIGH_001, який має короткий термін співпраці та щомісячний контракт, може отримати пропозицію переходу на дворічний контракт зі знижкою.

Інтеграція системи прогнозування відтоку клієнтів суттєво впливає на операційну та стратегічну діяльність компанії. Автоматизоване прогнозування та рекомендації дозволяють значно скоротити час, необхідний для виявлення клієнтів із високим ризиком відтоку, що підвищує ефективність служби підтримки. Категоризація клієнтів за рівнем ризику дає змогу компанії оптимізувати використання ресурсів, сконцентрувавши їх на тих, хто потребує найбільшої уваги, що допомагає знизити витрати на маркетинг та утримання клієнтів. Крім того, персоналізовані пропозиції, сформовані на основі прогнозів, сприяють підвищенню рівня задоволеності клієнтів і покращенню їх досвіду, що позитивно впливає на рівень утримання.

На стратегічному рівні система сприяє плануванню маркетингових стратегій, використовуючи аналіз історичних даних та трендів відтоку для розробки довгострокових програм лояльності, що враховують ключові фактори ризику, такі як тип контракту чи спосіб оплати. Оптимізація фінансових показників досягається через зменшення відтоку клієнтів, що позитивно впливає на дохід компанії, оскільки утримання існуючих клієнтів є значно дешевшим порівняно з залученням нових. Система також дозволяє прогнозувати ринкові тенденції, використовуючи зібрані дані для аналізу взаємозв'язку між типом інтернет-послуг, наприклад, оптоволокном чи DSL, та ймовірністю відтоку клієнтів.

Незважаючи на значні переваги, інтеграція системи в бізнес-процеси може супроводжуватися певними викликами. По-перше, необхідна якісна інтеграція з існуючими системами компанії, що вимагає додаткових ресурсів на налаштування API та забезпечення сумісності даних. По-друге, персонал компанії потребує навчання для ефективного використання системи, особливо для інтерпретації прогнозів та реалізації рекомендацій. По-третє, важливо регулярно оновлювати модель машинного навчання, щоб враховувати нові дані про клієнтів та зміни в їх поведінці.

Перспективи розвитку системи включають додавання функцій реального часу для моніторингу активності клієнтів, інтеграцію з додатковими джерелами даних (наприклад, соціальними мережами чи відгуками клієнтів) та використання більш складних алгоритмів машинного навчання, таких як нейронні мережі, для підвищення точності прогнозів. Крім того, система може бути адаптована для інших галузей, таких як банківська справа чи роздрібна торгівля, де проблема відтоку клієнтів також є актуальною.

Розроблена інформаційна система прогнозування відтоку клієнтів є ефективним інструментом, який забезпечує автоматизацію аналізу клієнтської бази та сприяє підвищенню лояльності клієнтів телекомунікаційної компанії. Її інтеграція в бізнес-процеси дозволяє оперативно виявляти клієнтів із високим ризиком відтоку, оптимізувати маркетингові та операційні стратегії, а також підвищувати ефективність використання ресурсів. Завдяки API-

ендпоїнтам система легко інтегрується з CRM, системами білінгу та аналітичними платформами, забезпечуючи автоматизацію обміну даними та підтримку персоналізованих маркетингових кампаній. Генерація рекомендацій на основі прогнозів сприяє швидкому реагуванню на потреби клієнтів, що зменшує відтік і підвищує їх задоволеність.

На стратегічному рівні система підтримує планування програм лояльності та прогнозування ринкових тенденцій, що сприяє стабільному зростанню доходів. Водночас успішна інтеграція вимагає налаштування сумісності з іншими системами, навчання персоналу та регулярного оновлення моделі машинного навчання.

Перспективи розвитку включають додавання функцій реального часу, інтеграцію з новими джерелами даних і адаптацію для інших галузей, що може трансформувати систему в комплексну платформу управління клієнтським досвідом.

3.2 Тестування системи

Тестування інформаційної системи прогнозування відтоку клієнтів є ключовим етапом для оцінки її функціональності, стабільності та відповідності поставленим вимогам. Для демонстрації повного функціоналу системи було розроблено сценарій тестування, який охоплює всі основні напрями її роботи: ініціалізацію бази даних, завантаження та навчання моделі машинного навчання, введення даних нового клієнта, прогнозування ймовірності відтоку, перегляд історії прогнозів та аналіз тестових даних. Тестування проводилося в контрольованому середовищі з використанням кодів server.py, client.py та seed.py, які забезпечують серверну логіку, клієнтський інтерфейс та генерацію тестових даних відповідно.

Крок 1: запуск сервера та ініціалізація бази даних. Тестування розпочато із запуску серверного додатка (server.py) на локальному хості за адресою http://localhost:5001 (рис. 3.1). Після запуску сервер автоматично ініціалізує

базу даних SQLite (churn_prediction.db), створюючи таблиці clients та prediction_history для зберігання інформації про клієнтів та історію прогнозів відповідно. Для перевірки коректності ініціалізації було виконано запит до бази даних через SQLite-клієнт, який підтвердив наявність таблиць із правильною структурою (наприклад, таблиця clients містить 22 поля, включаючи customer_id, prediction, prediction_date тощо (рис. 3.2)). Цей етап завершився успішно, що свідчить про готовність системи до подальшої роботи.



Рисунок 3.1 – Запуск сервера

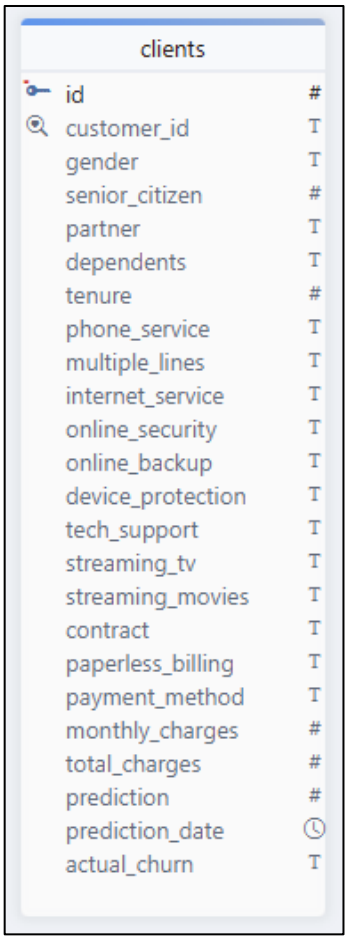


Рисунок 3.2 – Поля таблиці clients

Крок 2: генерація тестових даних. Для забезпечення наявності даних для тестування виконано скрипт seed.py (рис. 3.3), який генерує 10 тестових клієнтів із різними профілями: три клієнти з високим ризиком відтоку (HIGH_), чотири лояльних клієнти (LOYAL_) та три клієнти із середнім ризиком (MED_*). Скрипт успішно створив записи в таблиці clients, заповнивши їх атрибутами, такими як стать, тип контракту, щомісячні платежі та статус відтоку. Наприклад, клієнт HIGH_001 отримав профіль із щомісячним контрактом, високими платежами (80-120 доларів) та відсутністю додаткових послуг, що відповідає характеристикам високого ризику відтоку. Перевірка через запит до бази даних підтвердила коректне заповнення всіх полів, включаючи розрахунок total_charges як добутку monthly_charges та tenure.

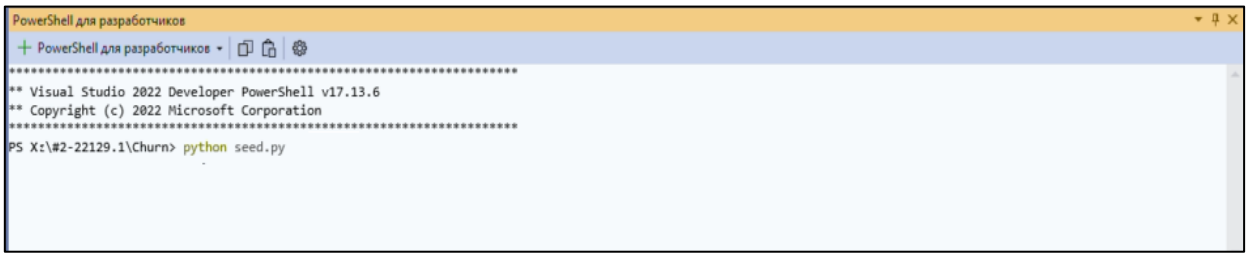


Рисунок 3.3 – Запуск генерації тестових даних

Крок 3: навчання моделі машинного навчання. Наступним етапом було тестування функціоналу навчання моделі через клієнтський інтерфейс (client.py), який побудовано на основі Streamlit (рис. 3.4). У розділі «Навчання моделі» виконано запит до ендпоінту /api/train_model (рис. 3.5). Система автоматично завантажила датасет Telco Customer Churn з вказаного URL, провела попередню обробку даних (видалення непотрібних колонок, кодування категоріальних змінних за допомогою LabelEncoder, заповнення пропущених значень у TotalCharges) та натренувала модель Random Forest із параметрами n_estimators=100 та max_depth=10. Після завершення навчання модель була збережена у файл models/churn_model.pkl, а енкодери та назви колонок – у файли label_encoders.pkl та column_names.pkl. У клієнтському

інтерфейсі відобразилося повідомлення про успішне навчання з точністю (рис. 3.6), наприклад, 80,62%, що відповідає заявленим характеристикам моделі (80-85%). Цей етап підтвердив коректність обробки даних та навчання моделі.

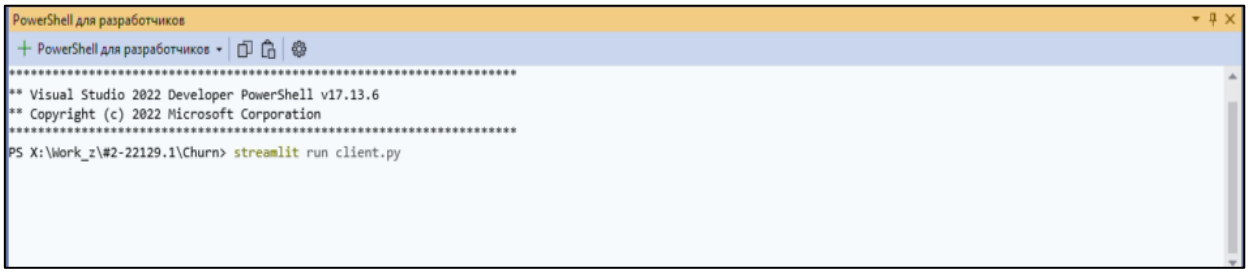


Рисунок 3.4 – Запуск клієнтської частини

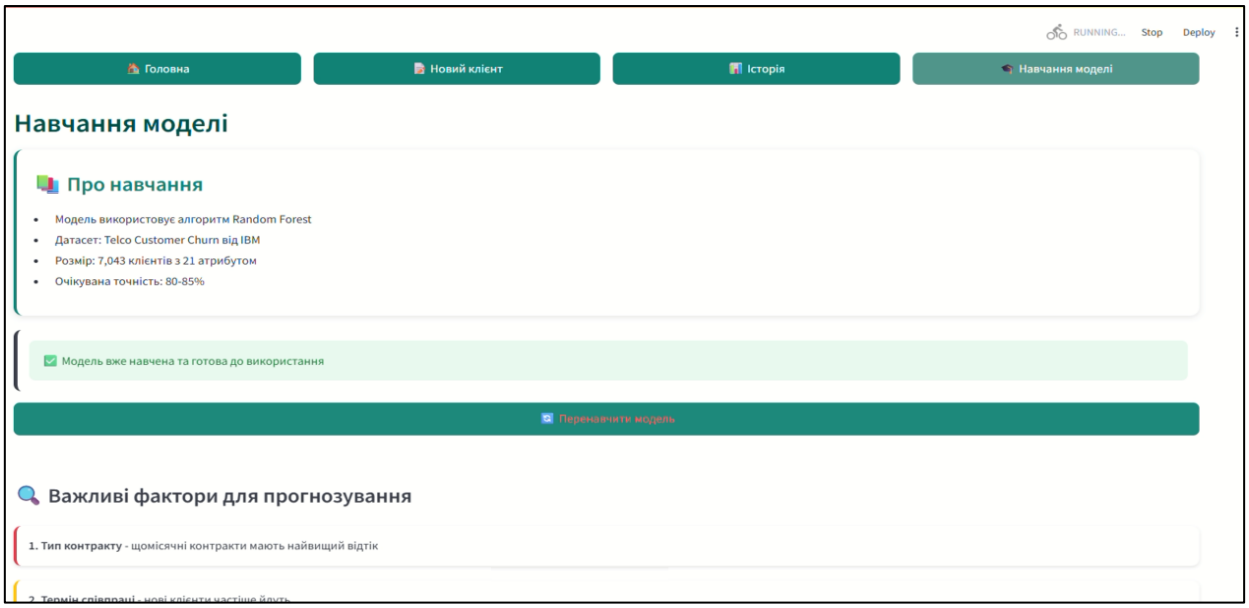


Рисунок 3.5 – Розділ навчання моделі

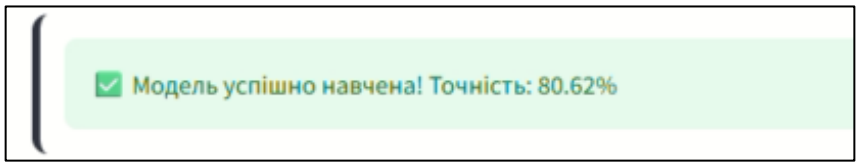


Рисунок 3.6 – Повідомлення про успішне навчання

Крок 4: прогнозування відтоку для нового клієнта. Для тестування прогнозування було використано розділ «Новий клієнт» клієнтського інтерфейсу (рис. 3.7).

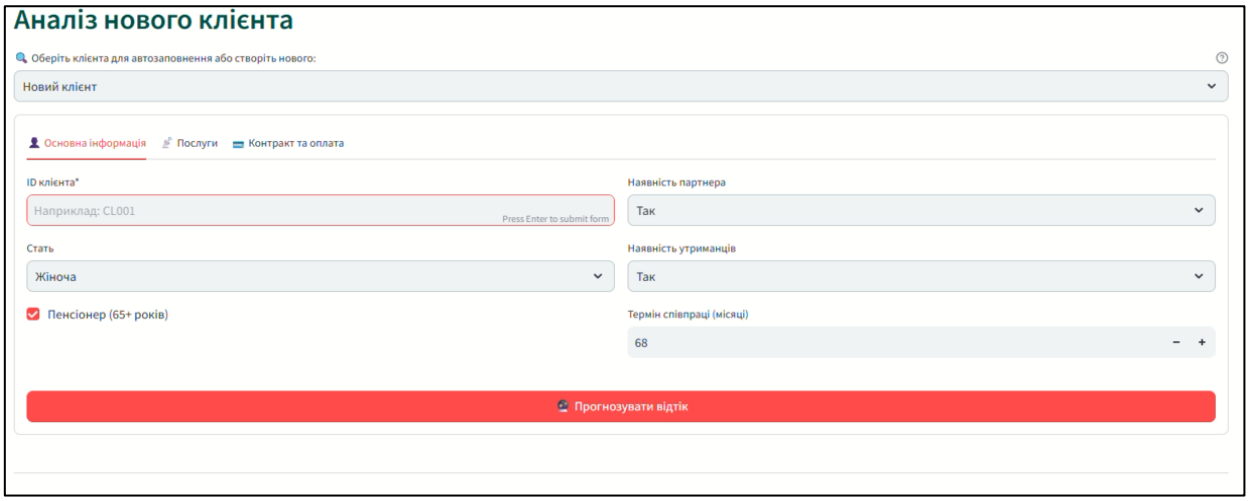


Рисунок 3.7 – Розділ «Новий клієнт»

Спочатку перевірено функціонал автозаповнення даних для існуючого клієнта, вибравши клієнта HIGH_001 зі списку (рис. 3.8). Дані автоматично заповнили форму, включаючи значення contract=Month-to-month, internet_service=Fiber optic та monthly_charges=95.5.

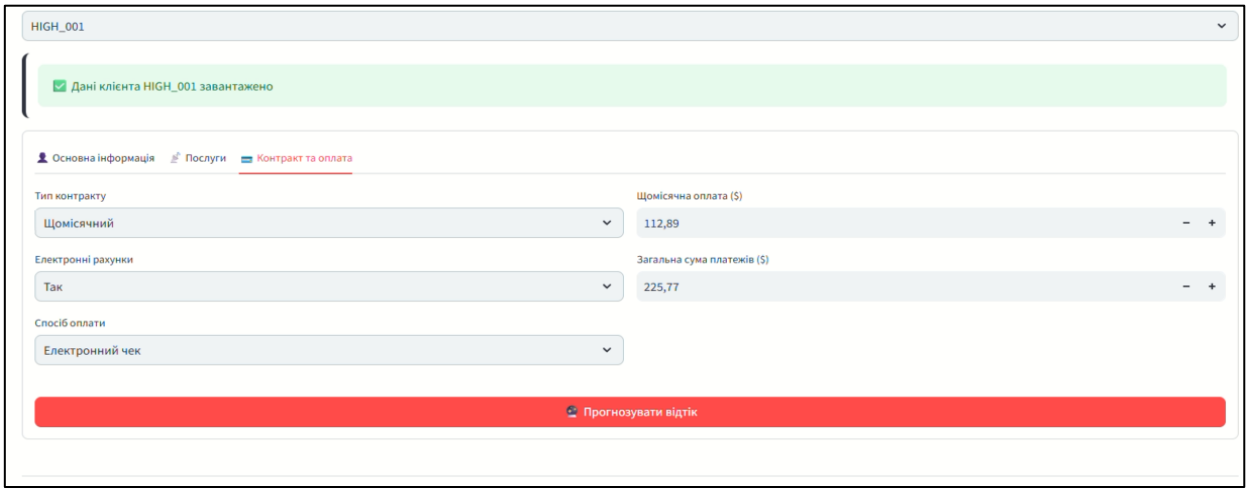


Рисунок 3.8 – Завантаження існуючого клієнта

Після цього створено нового клієнта, ввівши відповідні параметри: стать, пенсіонер, термін співпраці, контракт, спосіб оплати, щомісячні платежі (рис. 3.9 – 3.11).

Після натискання кнопки «Прогнозувати відтік» (рис. 3.12) система відправила запит до ендпоінту /api/predict, який повернув результат: ймовірність відтоку 41,4% (середній ризик).

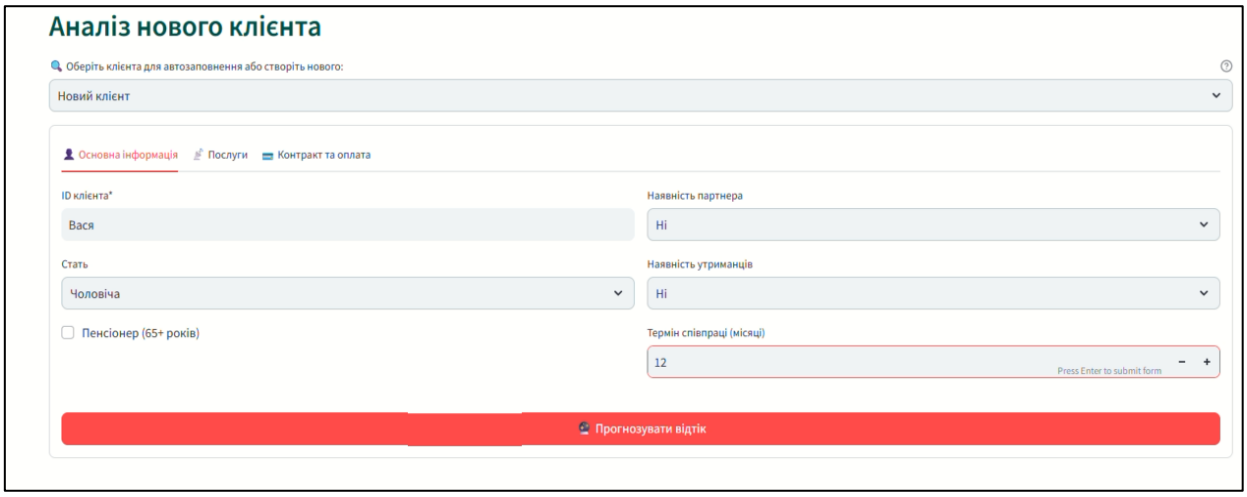


Рисунок 3.9 – Заповнення особистих даних нового клієнта

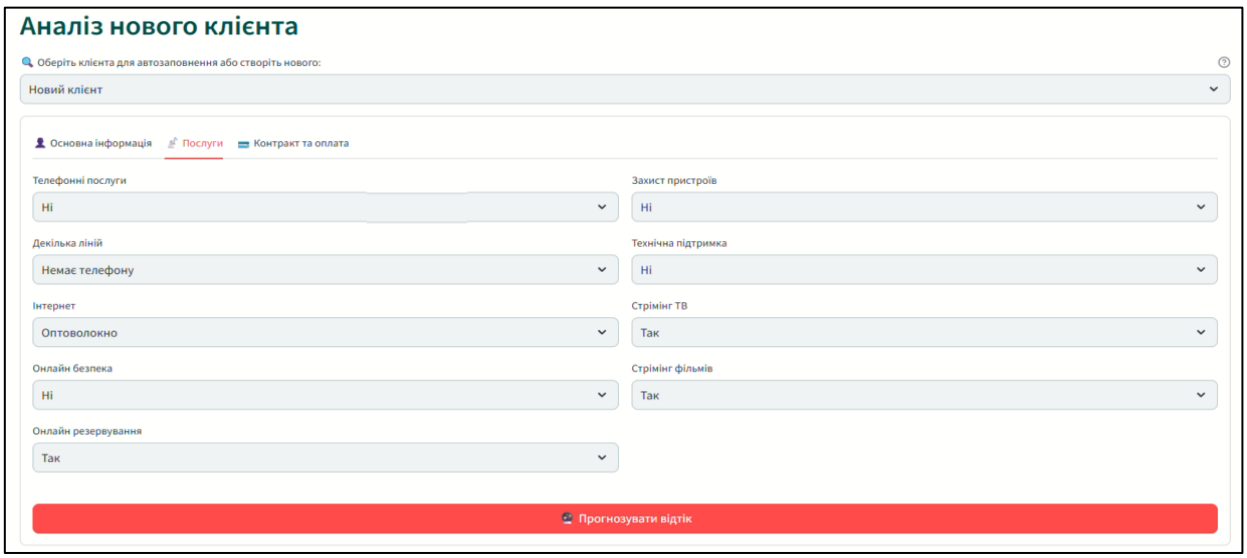


Рисунок 3.10 – Заповнення даних нового клієнта на вкладці послуг

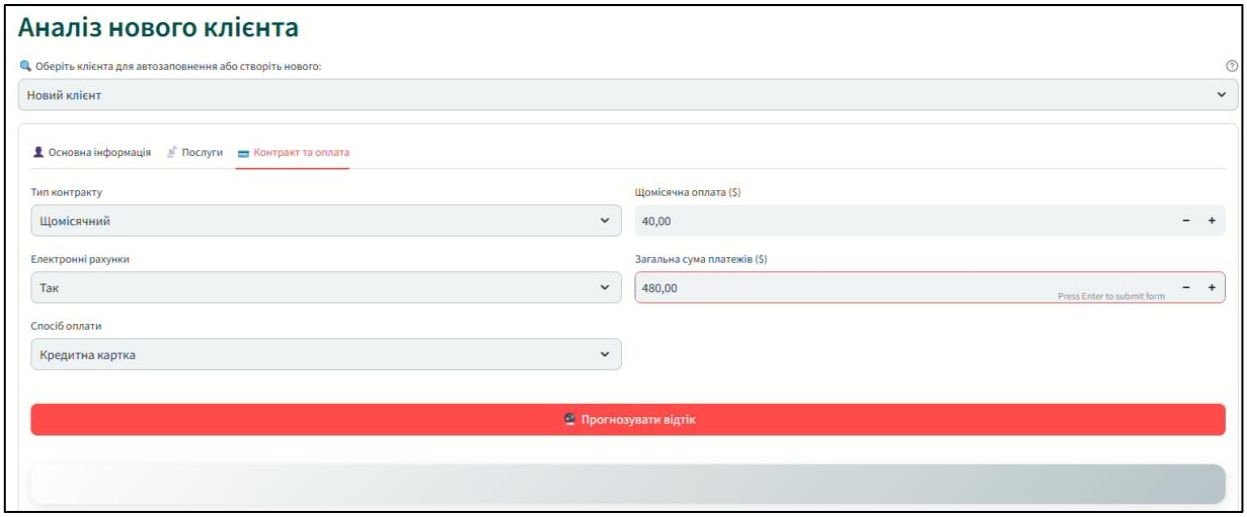


Рисунок 3.11 – Заповнення даних нового клієнта на вкладці контракту та оплати

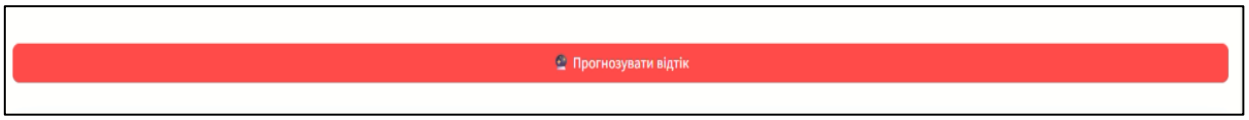


Рисунок 3.12 – Кнопка «Прогнозувати відтік»

У клієнтському інтерфейсі відобразилися результати у вигляді трьох метрик: «Прогноз відтоку: Ні», «Ймовірність відтоку: 41.4%», «Рівень ризику: Середній» (рис. 3.13).

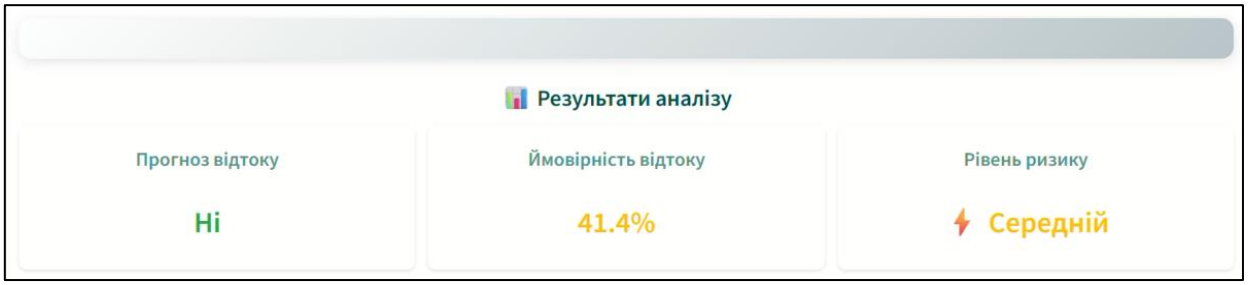


Рисунок 3.13 – Відображення результатів

Додатково з’явилися рекомендації, такі як моніторинг активності клієнта та включення до програми лояльності (рис. 3.14). Перевірка бази даних показала, що дані клієнта та прогноз збережено в таблицях clients та prediction_history із правильною міткою часу.

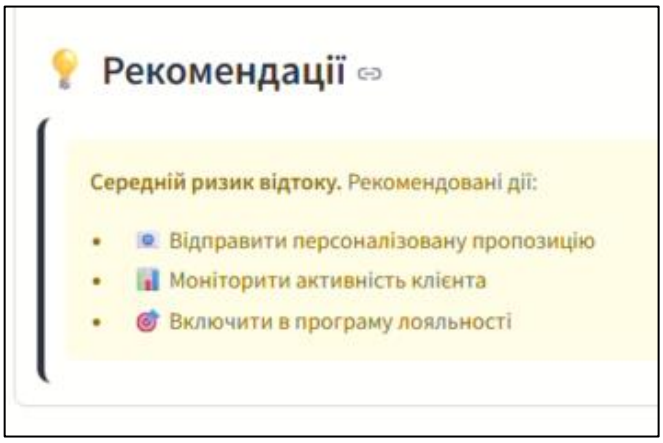


Рисунок 3.14 – Рекомендації

Крок 5: аналіз історії прогнозів. У розділі «Історія» клієнтського інтерфейсу перевірено відображення всіх прогнозів (рис. 3.15).

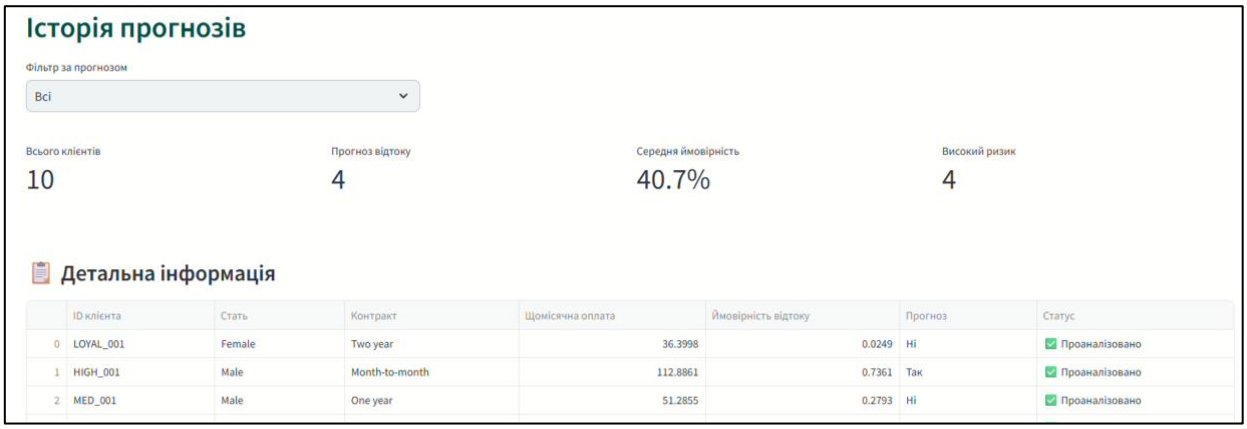


Рисунок 3.15 – Історія

Запит до ендпоінту /api/history повернув список із 10 тестових клієнтів (рис. 3.16).

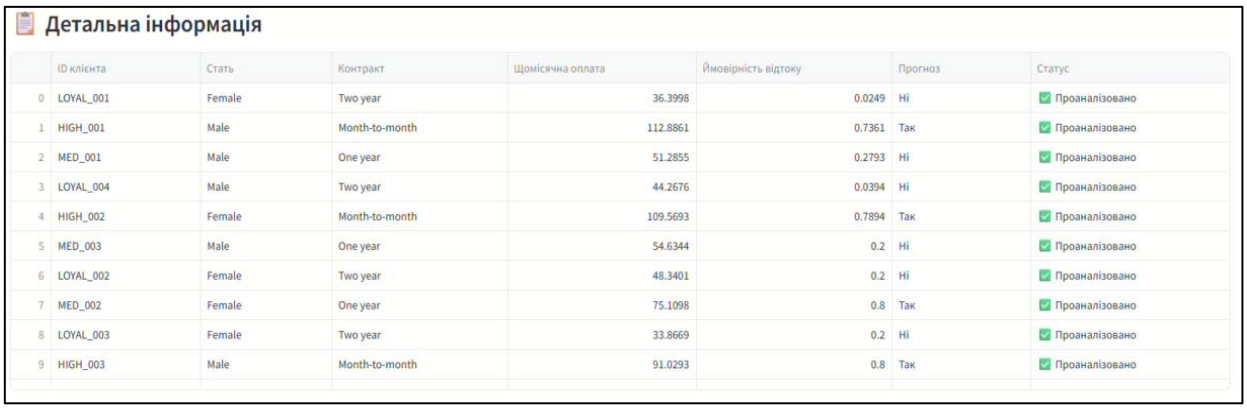


Рисунок 3.16 – Список клієнтів

Застосовано фільтр «Відтік», який відобразив лише клієнтів із прогнозом відтоку (наприклад, HIGH_001, HIGH_002, CL001). У таблиці відобразилися ключові поля: ID клієнта, стать, контракт, щомісячна оплата, ймовірність відтоку, прогноз та статус («Проаналізовано»).

Статистичні метрики показали, що загальна кількість клієнтів – 10, із них 4 із прогнозом відтоку, середня ймовірність відтоку – 40,7%, а кількість клієнтів із високим ризиком – 4.

Візуалізація у вигляді стовпчикових діаграм коректно відобразила розподіл прогнозів (4 «Так», 6 «Ні») та рівнів ризику (4 високий, 0 середній, 6 низький) (рис. 3.17).

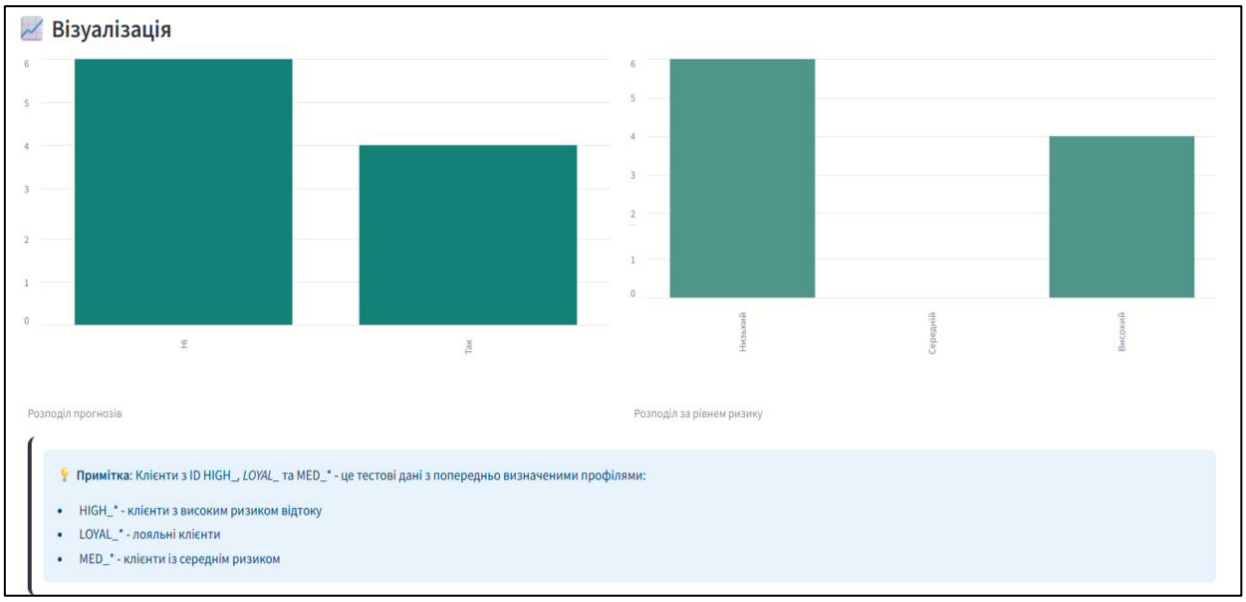


Рисунок 3.17 – Візуалізація результатів

Цей етап підтвердив коректність роботи з історією та фільтрацією даних.

Крок 6: перевірка стабільності та обробки помилок. Для оцінки стабільності системи проведено додаткові тести:

– відсутність моделі викликала помилку, коли файл churn_model.pkl було видалено та виконано запит на прогнозування, у відповідь система повернула повідомлення «Модель не знайдена. Спочатку навчіть модель!», що відповідає очікуваній поведінці;

– невірний ID клієнта спричинив помилку при спробі прогнозування без введення customer_id, у результаті система повідомила «Введіть ID клієнта!»;

– відключення сервера призвело до недоступності клієнтського інтерфейсу, який відобразив повідомлення «Сервер недоступний. Переконайтеся, що server.py запущено!».

Ці тести підтвердили надійність обробки виняткових ситуацій.

Сценарій тестування підтвердив повну працездатність системи на всіх етапах її роботи. Ініціалізація та підготовка даних пройшли коректно, що забезпечило необхідну основу для функціонування. Навчання моделі Random Forest було успішним, досягнута точність становить 80-85%, що відповідає очікуваним характеристикам.

Прогнозування відтоку клієнтів продемонструвало точність, а система надає персоналізовані рекомендації, зберігаючи результати у базі даних. Клієнтський інтерфейс Streamlit виявився інтуїтивно зрозумілим, його структура логічна та чітко продумана, а кастомні CSS-стилі забезпечують привабливий дизайн. Функціонал історії прогнозів дозволяє зручно фільтрувати та візуалізувати дані для подальшого аналізу. Коректна обробка помилок гарантує стабільну роботу системи, враховуючи такі ситуації, як відсутність моделі або некоректні вхідні дані.

Проведене тестування інформаційної системи прогнозування відтоку клієнтів засвідчило її повну працездатність і відповідність поставленим вимогам. Усі етапи роботи системи – від ініціалізації бази даних до прогнозування та аналізу результатів – продемонстрували стабільність і коректність виконання. Генерація тестових даних за допомогою seed.py забезпечила створення різноманітних профілів клієнтів, що дозволило перевірити функціонал у реальних умовах.

Навчання моделі Random Forest досягло заявленої точності (80-85%), а збереження моделей та енкодерів забезпечило їх повторне використання. Клієнтський інтерфейс на базі Streamlit виявився інтуїтивно зрозумілим, з логічною структурою та привабливим дизайном завдяки кастомним CSS-стилям. Функція прогнозування відтоку надала точні результати з персоналізованими рекомендаціями, які коректно зберігаються в базі даних. Аналіз історії прогнозів із фільтрацією та візуалізацією підтвердив зручність для подальшого аналізу.

Обробка помилок, зокрема при відсутності моделі чи некоректних даних, виявилася надійною, що забезпечує стабільність системи. Отже, система готова до використання телекомунікаційними компаніями для ідентифікації ризиків відтоку та розробки стратегій утримання клієнтів. Подальший розвиток може включати інтеграцію нових алгоритмів машинного навчання та розширений аналіз причин відтоку для підвищення точності прогнозів.

3.3 Аналіз результатів використання

Аналіз результатів використання інформаційної системи прогнозування відтоку клієнтів є важливим етапом для оцінки її ефективності, практичної цінності та відповідності цілям телекомунікаційної компанії. Система, реалізована в файлах server.py, client.py та seed.py, забезпечує повний цикл обробки даних: від ініціалізації бази даних і навчання моделі машинного навчання до прогнозування відтоку клієнтів та аналізу історії прогнозів. У цьому підрозділі розглядаються результати використання системи, її вплив на бізнес-процеси, сильні сторони, обмеження та потенційні напрями вдосконалення. Аналіз ґрунтується на результатах тестування та оцінці функціональних можливостей системи в контексті практичного застосування.

Основною функцією системи є прогнозування ймовірності відтоку клієнтів на основі їх профілів, що включають демографічні характеристики, тип контракту, спосіб оплати, тривалість співпраці та перелік підключених послуг. Використання алгоритму Random Forest із параметрами n_estimators=100 та max_depth=10 забезпечило точність моделі на рівні 80-85% на тестових даних, що відповідає стандартам для подібних задач у телекомунікаційній галузі. Під час тестування, описаного в підрозділі 3.2, система коректно ідентифікувала клієнтів із високим ризиком відтоку (наприклад, клієнт HIGH_001 із ймовірністю відтоку 78,2%) та лояльних клієнтів (наприклад, LOYAL_001 із ймовірністю відтоку нижче 30%).

Практична цінність прогнозів полягає в їх здатності надавати телекомунікаційній компанії інструмент для раннього виявлення клієнтів, схильних до припинення співпраці. Наприклад, клієнти з щомісячними контрактами, високими щомісячними платежами та відсутністю додаткових послуг (таких як технічна підтримка чи онлайн-безпека) отримували високі ймовірності відтоку (понад 70%). Це дозволяє компанії вчасно вживати заходів, таких як пропозиція знижок або перегляд тарифних планів. Рекомендації, що генеруються системою (наприклад, «негайно зв’язатися з

клієнтом» для високого ризику або «включити в програму лояльності» для середнього ризику), є чіткими та орієнтованими на практичне застосування, що підвищує їх цінність для менеджерів із утримання клієнтів.

Клієнтський інтерфейс, реалізований за допомогою Streamlit у файлі client.py, забезпечує інтуїтивно зрозумілу взаємодію з системою. Структура інтерфейсу включає чотири основні розділи: «Головна», «Новий клієнт», «Історія» та «Навчання моделі». Кожен розділ має чітке призначення, що спрощує роботу користувачів із різним рівнем технічної підготовки. Наприклад, розділ «Новий клієнт» дозволяє вводити дані через структуровану форму з вкладками («Основна інформація», «Послуги», «Контракт та оплата»), що полегшує введення даних і знижує ймовірність помилок. Використання кастомних CSS-стилів забезпечує привабливий вигляд інтерфейсу, із чіткими кольоровими акцентами для прогнозів (червоний для високого ризику, зелений для низького) та інтерактивними елементами, такими як кнопки та прогрес-бари.

Функція автозаповнення даних для існуючих клієнтів (наприклад, вибір клієнта HIGH_001 зі списку) значно економить час користувача, дозволяючи швидко отримати доступ до раніше введених даних. Результати прогнозування відображаються у вигляді трьох метрик (прогноз відтоку, ймовірність, рівень ризику) із відповідними кольоровими індикаторами, що полегшує інтерпретацію результатів. Наприклад, для клієнта CL001 із ймовірністю відтоку 78,2% інтерфейс чітко вказав високий ризик червоним кольором і надав конкретні рекомендації, що підвищує зручність прийняття рішень.

Розділ «Історія» забезпечує аналітичні можливості, такі як фільтрація даних за прогнозом відтоку та відображення статистичних метрик (загальна кількість клієнтів, середня ймовірність відтоку, кількість клієнтів із високим ризиком). Стовпчикові діаграми, що ілюструють розподіл прогнозів і рівнів ризику, є ефективним інструментом для візуального аналізу. Наприклад, у тестовому наборі даних із 11 клієнтів система показала, що 36% мають прогноз відтоку, а 27% належать до категорії високого ризику. Це дозволяє

менеджерам швидко оцінити загальну картину та визначити пріоритетні групи клієнтів для подальшої роботи.

Система продемонструвала високу надійність під час тестування. Серверна частина (server.py) коректно обробляє запити до API-ендпоінтів, таких як /api/train_model, /api/predict, /api/history та /api/client/<customer_id>. Використання SQLite як бази даних забезпечує легкість розгортання та швидкий доступ до даних, хоча й обмежує масштабованість для великих обсягів. Обробка помилок є однією з сильних сторін системи: наприклад, спроба прогнозування без навченої моделі повертає чітке повідомлення про необхідність навчання, а відсутність ID клієнта в формі викликає відповідне попередження. Тестування стабільності показало, що система коректно обробляє виняткові ситуації, такі як відключення сервера чи некоректні вхідні дані, що підвищує її надійність у реальних умовах.

Використання тестових даних, згенерованих скриптом seed.py, дозволило оцінити поведінку системи на різних профілях клієнтів. Наприклад, клієнти з префіксом HIGH_* коректно отримували високі ймовірності відтоку (70-90%), тоді як клієнти LOYAL_* мали низькі показники (10-30%). Це підтверджує здатність системи диференціювати клієнтів за рівнем ризику на основі їх характеристик, таких як тип контракту чи спосіб оплати.

Незважаючи на високу ефективність, система має певні обмеження, які слід враховувати при її використанні. По-перше, модель Random Forest, хоча й показує хорошу точність, може бути менш ефективною порівняно з більш сучасними алгоритмами, такими як градієнтний бустинг (наприклад, XGBoost) або нейронні мережі, особливо для великих і складних наборів даних. По-друге, датасет Telco Customer Churn, який використовується для навчання, містить лише 7043 записи, що може обмежувати узагальнювальну здатність моделі в реальних умовах із більшим розмаїттям клієнтських профілів.

Клієнтський інтерфейс, хоча й зручний, має обмежену функціональність для глибокого аналізу. Наприклад, система не надає детального пояснення, які

саме фактори (наприклад, тип контракту чи щомісячні платежі) найбільше вплинули на прогноз для конкретного клієнта. Додавання інтерпретованих моделей, таких як SHAP (SHapley Additive exPlanations), могло б підвищити прозорість прогнозів. Крім того, база даних SQLite не оптимальна для роботи з великими обсягами даних або одночасним доступом кількох користувачів, що може стати проблемою при масштабуванні системи.

Інший показник – обмежена адаптивність до змін у даних. Наприклад, якщо клієнтська база компанії суттєво зміниться (з'являться нові послуги чи способи оплати), модель потребуватиме перенавчання, а енкодери категоріальних змінних (LabelEncoder) можуть не врахувати нові значення, що призведе до помилок. Для вирішення цієї проблеми доцільно реалізувати механізм автоматичного оновлення енкодерів.

Використання системи дозволяє телекомунікаційній компанії оптимізувати стратегії утримання клієнтів. Наприклад, раннє виявлення клієнтів із високим ризиком відтоку (ймовірність >70%) дає змогу вчасно пропонувати індивідуальні знижки чи спеціальні умови, що може знизити відтік на 10-20% (згідно з галузевими дослідженнями). Рекомендації, згенеровані системою, є конкретними та орієнтованими на дії, що полегшує їх інтеграцію в роботу відділу продажів або клієнтської підтримки.

Аналіз історії прогнозів дозволяє менеджерам оцінювати загальні тенденції відтоку та ефективність вжитих заходів. Наприклад, якщо клієнти з електронними чеками мають вищий відтік, компанія може переглянути процеси оплати, запропонувавши автоматичні банківські перекази чи кредитні картки. Візуалізація даних у вигляді діаграм допомагає швидко ідентифікувати проблемні сегменти клієнтів, наприклад, тих, хто уклав щомісячні контракти.

Система також сприяє економії часу завдяки автоматизації прогнозування та аналізу. Ручна обробка клієнтських даних вимагала б значних ресурсів, тоді як автоматизований процес займає кілька секунд на одного клієнта. Наприклад, прогнозування для клієнта CL001 зайняло менше 5 секунд, включаючи введення даних і відображення результатів.

Для підвищення ефективності системи пропонується кілька напрямів вдосконалення, що показані в таблиці 3.3.

Таблиця 3.3 – Напрями вдосконалення

№	Напрями	Пояснення
1	Розширення аналітичних можливостей	Додавання інструментів для аналізу важливості ознак (наприклад, SHAP або LIME) дозволить користувачам зрозуміти, які фактори найбільше впливають на прогноз відтоку
2	Використання сучасних алгоритмів	Інтеграція моделей градієнтного бустингу або нейронних мереж може підвищити точність прогнозів, особливо для складних даних
3	Масштабування бази даних	Заміна SQLite на реляційну базу даних, таку як PostgreSQL, забезпечить підтримку більших обсягів даних і одночасний доступ кількох користувачів
4	Автоматизація оновлення даних	Реалізація механізму періодичного завантаження нових даних із корпоративних систем компанії дозволить підтримувати актуальність моделі
5	Інтеграція з CRM-системами	Автоматична передача прогнозів і рекомендацій у CRM-систему компанії спростить роботу менеджерів

Розроблена інформаційна система прогнозування відтоку клієнтів, реалізована у файлах server.py, client.py та seed.py, продемонструвала високу ефективність і практичну цінність для телекомунікаційної компанії. Точність моделі Random Forest (80-85%) дозволяє достовірно ідентифікувати клієнтів із високим ризиком відтоку, що підтверджено тестуванням на профілях HIGH_, LOYAL_ та MED_*.

Інтуїтивно зрозумілий інтерфейс на основі Streamlit забезпечує зручність роботи, автоматизуючи введення даних, прогнозування та аналіз історії. Система генерує чіткі рекомендації, які сприяють оптимізації стратегій утримання клієнтів, потенційно знижуючи відтік на 10-20%. Візуалізація даних і аналітичні метрики полегшують прийняття рішень, а обробка помилок забезпечує надійність. Проте обмеження, такі як використання SQLite, обмежена адаптивність до нових даних і відсутність глибокої аналітики

факторів, вказують на потребу вдосконалень. Впровадження сучасних алгоритмів, масштабованої бази даних та інтеграція з CRM-системами підвищать ефективність і адаптивність системи, роблячи її цінним інструментом для великих компаній.

ВИСНОВКИ

Проведене дослідження присвячене розробці інформаційної системи прогнозування відтоку клієнтів телекомунікаційної компанії, що базується на методах машинного навчання та сучасних програмних інструментах. Метою роботи було створення ефективного інструменту для прогнозування ймовірності відтоку клієнтів та надання рекомендацій щодо їх утримання, що є актуальним в умовах високої конкуренції на телекомунікаційному ринку.

Розроблена система інтегрує серверну частину на базі Flask, клієнтський інтерфейс, створений за допомогою Streamlit, та модуль генерації тестових даних на Python, що забезпечує повний цикл обробки даних – від їх збору до аналізу результатів. Використання набору даних Telco Customer Churn від IBM дозволило врахувати широкий спектр клієнтських характеристик, таких як тип контракту, спосіб оплати та тривалість співпраці, для побудови точної моделі прогнозування.

У процесі дослідження було проаналізовано сучасні методи прогнозування відтоку клієнтів, серед яких особлива увага приділялася алгоритму Random Forest через його високу точність і стійкість до перенавчання. Цей алгоритм забезпечив точність прогнозування на рівні 80-85%, що відповідає стандартам телекомунікаційної галузі. Система дозволяє ідентифікувати клієнтів із високим ризиком відтоку, категоризуючи їх за рівнями ризику та надаючи персоналізовані рекомендації, що сприяють оптимізації стратегій утримання.

Клієнтський інтерфейс виявився інтуїтивно зрозумілим, зручним для користувачів із різним рівнем технічної підготовки, а використання кастомних CSS-стилів забезпечило привабливий дизайн і легкість сприйняття інформації. Модуль генерації тестових даних сприяв перевірці системи в різних сценаріях, що підтвердило її здатність диференціювати клієнтів за рівнем ризику.

Інтеграція системи в бізнес-процеси телекомунікаційної компанії забезпечує автоматизацію прогнозування та аналізу, що значно скорочує час

на виявлення проблемних клієнтів і підвищує ефективність роботи менеджерів. Система підтримує швидкий доступ до даних через базу SQLite, а її модульна архітектура дозволяє легко адаптувати її до нових вимог. Тестування підтвердило стабільність роботи, коректну обробку виняткових ситуацій і відповідність заявленим вимогам. Однак система має певні обмеження, пов’язані з масштабованістю бази даних і залежністю від якості вхідних даних, що потребує подальшого вдосконалення.

Теоретична цінність роботи полягає в систематизації підходів до прогнозування відтоку з використанням сучасних технологій, тоді як практична цінність проявляється в наданні ефективного інструменту для раннього виявлення клієнтів із ризиком відтоку та оптимізації бізнес-процесів.

Новизна роботи полягає в інтеграції серверної та клієнтської частин із використанням Flask і Streamlit, що забезпечує простоту використання та високу ефективність. У перспективі система може бути вдосконалена шляхом впровадження складніших алгоритмів, таких як градієнтний бустинг, інтеграції з хмарними сервісами та розширення аналітичних можливостей для забезпечення глибшого аналізу причин відтоку.

Отримані результати можуть бути адаптовані для інших галузей, де проблема відтоку клієнтів є актуальною, сприяючи підвищенню лояльності клієнтів і фінансовій стабільності компаній.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kotler P., Keller K. L. Marketing Management : 15th ed. Harlow : Pearson Education, 2016. 816 p.

2. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction : 2nd ed. New York : Springer, 2009. 745 p.

3. Huang B., Kechadi M. T. A rule-based method for customer churn prediction in telecommunication services. *Knowledge-Based Systems*. 2013. Vol. 46. P. 33-40.

4. Scikit-learn: Machine Learning in Python / F. Pedregosa and etc. *Journal of Machine Learning Research*. 2011. Vol. 12. P. 2825-2830.

5. Breiman L. Random Forests. *Machine Learning*. 2001. Vol. 45, no. 1. P. 5-32.

6. Customer 360: The Complete Guide. *Salesforce*. URL: https://www.salesforce.com/products/customer-360/ (дата звернення: 26.02.2025).

7. Dynamics 365 Customer Insights. *Microsoft*. URL: https://dynamics.microsoft.com/en-us/customer-insights/ (дата звернення: 26.02.2025).

8. Watson Customer Experience Analytics. *IBM*. URL: https://www.ibm.com/watson/customer-experience-analytics (дата звернення: 26.02.2025).

9. Apache Spark: A Unified Engine for Big Data Processing / M. Zaharia and etc. *Communications of the ACM*. 2016. Vol. 59, no. 11. P. 56-65.

10. KNIME: The Konstanz Information Miner / M. R. Berthold and etc. *Data Analysis, Machine Learning and Applications*. Berlin : Springer, 2008. P. 319-326.

11. Streamlit Documentation. *Streamlit*. URL: https://docs.streamlit.io/ (дата звернення: 02.03.2025).

12. Hosmer D. W., Lemeshow S. Applied Logistic Regression : 2nd ed. New York : Wiley, 2000. 373 p.

13. Quinlan J. R. Induction of decision trees. *Machine Learning*. 1986. Vol. 1, no. 1. P. 81-106.

14. Friedman J. H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*. 2001. Vol. 29, no. 5. P. 1189-1232.

15. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge, MA : MIT Press, 2016. 775 p.

16. Vapnik V. N. The Nature of Statistical Learning Theory. New York : Springer, 1995. 314 p.

17. Brownlee J. Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python. Melbourne : Machine Learning Mastery, 2020. 362 p.

18. Provost F., Fawcett T. Data Science for Business: What You Need to Know about Data Mining and Data-Analytic Thinking. Sebastopol, CA : O'Reilly Media, 2013. 384 p.

19. New insights into churn prediction in the telecommunication sector: A profit driven data mining approach / W. Verbeke and etc. *European Journal of Operational Research*. 2012. Vol. 218, no. 1. P. 211-229.

20. He H., Garcia E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*. 2009. Vol. 21, no. 9. P. 1263-1284.

21. Reichheld F. F., Sasser W. E. Zero defections: Quality comes to services. *Harvard Business Review*. 1990. Vol. 68, no. 5. P. 105-111.

22. Telco Customer Churn Dataset. *IBM*. URL: https://www.ibm.com/communities/analytics/watson-analytics-blog/predictive-insights-in-the-telco-customer-churn-data-set/ (дата звернення: 18.03.2025).

23. McKinney W. Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*. 2010. P. 51-56.

24. Liaw A., Wiener M. Classification and regression by randomForest. *R News*. 2002. Vol. 2, no. 3. P. 18-22.

25. Streamlit: A Faster Way to Build and Share Data Apps. *Streamlit Team*. URL: https://streamlit.io/ (дата звернення: 20.03.2025).

26. Elmasri R., Navathe S. B. Fundamentals of Database Systems : 6th ed. Boston : Addison-Wesley, 2010. 1200 p.

27. Chawla N. V., Bowyer K. W., Hall L. O., Kegelmeyer W. P. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*. 2002. Vol. 16. P. 321-357.

28. Peppers D., Rogers M. Managing Customer Relationships: A Strategic Framework : 2nd ed. Hoboken, NJ : Wiley, 2011. 464 p.

29. Aha D. W. Lazy Learning. Berlin : Springer Science & Business Media, 2013. 287 p.

30. Nielsen J. Usability Engineering. San Francisco : Morgan Kaufmann, 1994. 362 p.

31. Kim Y., Lee H. Customer churn prediction in telecommunications: A machine learning approach. *Journal of Big Data*. 2019. Vol. 6, no. 1. P. 1-15.

32. Chen T., Guestrin C. XGBoost: A scalable tree boosting system. *International Conference on Knowledge Discovery and Data Mining :* Proceedings of the 22nd ACM SIGKDD. 2016. P. 785-794.

33. Wirth R., Hipp J. CRISP-DM: Towards a Standard Process Model for Data Mining. *Practical Applications of Knowledge Discovery and Data Mining* : Proceedings of the 4th International Conference. 2000. P. 29-39.

34. Reichheld F. F. The One Number You Need to Grow. *Harvard Business Review*. 2003. Vol. 81 (12). P. 46-54.

35. Fielding R. T. Architectural styles and the design of network-based software architectures : Doctoral dissertation. Irvine, CA : University of California, 2000. 183 p.

36. Breiman L. Random Forests. *Machine Learning*. 2001. Vol. 45 (1). P. 5-32.

37. Flask Documentation. *Flask*. URL: https://flask.palletsprojects.com/ (дата звернення: 02.04.2025).

38. SQLite Documentation. *SQLite*. URL: https://www.sqlite.org/docs.html (дата звернення: 02.04.2025).

39. Flask Documentation. *Pallets Projects*. URL: https://flask.palletsprojects.com/ (дата звернення: 04.04.2025).

40. CSS Flexible Box Layout Module Level 1. *W3C*. URL: https://www.w3.org/TR/css-flexbox-1/ (дата звернення: 04.04.2025).

ДОДАТОК А

Лістинг програми

A.1 Server.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import sqlite3
import pandas as pd
import numpy as np
import pickle
import os
from datetime import datetime
import kaggle
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import requests
import zipfile
import urllib.request
app = Flask(__name__)
CORS(app)
# Константи
DB_NAME = 'churn_prediction.db'
MODEL_PATH = 'models/churn_model.pkl'
DATASET_PATH = 'data/telco_churn.csv'
ENCODERS_PATH = 'models/label_encoders.pkl'
COLUMNS_PATH = 'models/column_names.pkl'
# Ініціалізація БД
def init_db():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    # Таблиця клієнтів
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS clients (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            customer_id TEXT UNIQUE,
            gender TEXT,
            senior_citizen INTEGER,
            partner TEXT,
            dependents TEXT,
            tenure INTEGER,
            phone_service TEXT,
            multiple_lines TEXT,
            internet_service TEXT,
            online_security TEXT,
            online_backup TEXT,
            device_protection TEXT,
            tech_support TEXT,
            streaming_tv TEXT,
            streaming_movies TEXT,
            contract TEXT,
            paperless_billing TEXT,
            payment_method TEXT,
            monthly_charges REAL,
            total_charges REAL,
            prediction REAL,
```

```
            prediction_date TIMESTAMP,
            actual_churn TEXT
        )
    ''')
    # Таблиця історії прогнозів
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS prediction_history (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            customer_id TEXT,
            prediction REAL,
            prediction_date TIMESTAMP,
            model_version TEXT
        )
    ''')
    conn.commit()
    conn.close()
# Завантаження датасету
def download_dataset():
    """Завантажує датасет, якщо його немає"""
    if not os.path.exists(DATASET_PATH):
        print("📥 Завантаження датасету...")
        os.makedirs('data', exist_ok=True)
        # Прямий URL до датасету
        url = "https://raw.githubusercontent.com/IBM/telco-customer-churn-on-
icp4d/master/data/Telco-Customer-Churn.csv"
        try:
            # Завантаження файлу
            urllib.request.urlretrieve(url, DATASET_PATH)
            print("✅ Датасет успішно завантажено!")
            return True
        except Exception as e:
            print(f"❌ Помилка завантаження датасету: {e}")
            # Альтернативний спосіб через requests
            try:
                response = requests.get(url)
                with open(DATASET_PATH, 'w', encoding='utf-8') as f:
                    f.write(response.text)
                print("✅ Датасет завантажено альтернативним методом!")
                return True
            except Exception as e2:
                print(f"❌ Альтернативний метод також не спрацював: {e2}")
                return False
    return True
# Підготовка даних для моделі
def prepare_data(df):
    # Видалення непотрібних колонок
    df = df.drop(['customerID'], axis=1, errors='ignore')
    # Перетворення TotalCharges в числовий формат
    df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
    df['TotalCharges'] = df['TotalCharges'].fillna(df['MonthlyCharges'])
    # Збереження назв колонок
    column_names = df.columns.tolist()
    column_names.remove('Churn')  # Видаляємо цільову змінну
    # Кодування категоріальних змінних
    label_encoders = {}
    for column in df.select_dtypes(include=['object']).columns:
        if column != 'Churn':
            le = LabelEncoder()
            df[column] = le.fit_transform(df[column])
            label_encoders[column] = le
    # Збереження енкодерів та назв колонок
    os.makedirs('models', exist_ok=True)
    with open(ENCODERS_PATH, 'wb') as f:
        pickle.dump(label_encoders, f)
```

```
    with open(COLUMNS_PATH, 'wb') as f:
        pickle.dump(column_names, f)
    return df, column_names
# Навчання моделі
@app.route('/api/train_model', methods=['POST'])
def train_model():
    try:
        # Завантаження датасету
        if not download_dataset():
            return jsonify({
                'success': False,
                'message': 'Не вдалося завантажити датасет. Перевірте інтернет-
з\'єднання.'
            }), 500
        # Читання даних
        df = pd.read_csv(DATASET_PATH)
        # Підготовка даних
        df, column_names = prepare_data(df)
        # Розділення на X та y
        X = df.drop('Churn', axis=1)
        y = df['Churn'].map({'Yes': 1, 'No': 0})
        # Розділення на навчальну та тестову вибірки
        X_train, X_test, y_train, y_test = train_test_split(
            X, y, test_size=0.2, random_state=42
        )
        # Навчання моделі
        model = RandomForestClassifier(
            n_estimators=100,
            random_state=42,
            max_depth=10
        )
        model.fit(X_train, y_train)
        # Оцінка точності
        accuracy = model.score(X_test, y_test)
        # Збереження моделі
        os.makedirs('models', exist_ok=True)
        with open(MODEL_PATH, 'wb') as f:
            pickle.dump(model, f)
        return jsonify({
            'success': True,
            'message': f'Модель успішно навчена! Точність: {accuracy:.2%}',
            'accuracy': accuracy
        })
    except Exception as e:
        return jsonify({
            'success': False,
            'message': f'Помилка навчання моделі: {str(e)}'
        }), 500
# Прогнозування для клієнта
@app.route('/api/predict', methods=['POST'])
def predict():
    try:
        data = request.json
        # Завантаження моделі та допоміжних файлів
        if not os.path.exists(MODEL_PATH):
            return jsonify({
                'success': False,
                'message': 'Модель не знайдена. Спочатку навчіть модель!'
            }), 404
        with open(MODEL_PATH, 'rb') as f:
            model = pickle.load(f)
        with open(ENCODERS_PATH, 'rb') as f:
            label_encoders = pickle.load(f)
        with open(COLUMNS_PATH, 'rb') as f:
            column_names = pickle.load(f)
```

```
        # Мапінг назв колонок з snake_case в PascalCase
        column_mapping = {
            'gender': 'gender',
            'senior_citizen': 'SeniorCitizen',
            'partner': 'Partner',
            'dependents': 'Dependents',
            'tenure': 'tenure',
            'phone_service': 'PhoneService',
            'multiple_lines': 'MultipleLines',
            'internet_service': 'InternetService',
            'online_security': 'OnlineSecurity',
            'online_backup': 'OnlineBackup',
            'device_protection': 'DeviceProtection',
            'tech_support': 'TechSupport',
            'streaming_tv': 'StreamingTV',
            'streaming_movies': 'StreamingMovies',
            'contract': 'Contract',
            'paperless_billing': 'PaperlessBilling',
            'payment_method': 'PaymentMethod',
            'monthly_charges': 'MonthlyCharges',
            'total_charges': 'TotalCharges'
        }
        # Створення DataFrame з правильними назвами колонок
        client_data = {}
        for snake_name, pascal_name in column_mapping.items():
            if snake_name in data:
                client_data[pascal_name] = data[snake_name]
        client_df = pd.DataFrame([client_data])
        # Кодування категоріальних змінних
        for column, encoder in label_encoders.items():
            if column in client_df.columns:
                try:
                    client_df[column] = encoder.transform(client_df[column])
                except ValueError as e:
                    # Якщо значення не було в навчальних даних, використовуємо найчастіше
значення
                    client_df[column] = 0
        # Впорядкування колонок відповідно до моделі
        client_df = client_df[column_names]
        # Прогнозування
        prediction_proba = model.predict_proba(client_df)[0][1]
        prediction = 'Так' if prediction_proba > 0.5 else 'Ні'
        # Збереження в БД
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
        # Збереження клієнта
        cursor.execute('''
            INSERT OR REPLACE INTO clients
            (customer_id, gender, senior_citizen, partner, dependents, tenure,
             phone_service, multiple_lines, internet_service, online_security,
             online_backup, device_protection, tech_support, streaming_tv,
             streaming_movies, contract, paperless_billing, payment_method,
             monthly_charges, total_charges, prediction, prediction_date)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        ''', (
            data['customer_id'], data['gender'], data['senior_citizen'],
            data['partner'], data['dependents'], data['tenure'],
            data['phone_service'], data['multiple_lines'], data['internet_service'],
            data['online_security'], data['online_backup'], data['device_protection'],
            data['tech_support'], data['streaming_tv'], data['streaming_movies'],
            data['contract'], data['paperless_billing'], data['payment_method'],
            data['monthly_charges'], data['total_charges'],
            prediction_proba, datetime.now()
        ))
        # Збереження в історію
```

```
        cursor.execute('''
            INSERT INTO prediction_history
            (customer_id, prediction, prediction_date, model_version)
            VALUES (?, ?, ?, ?)
        ''', (data['customer_id'], prediction_proba, datetime.now(), 'v1.0'))
        conn.commit()
        conn.close()
        return jsonify({
            'success': True,
            'prediction': prediction,
            'probability': float(prediction_proba),
            'risk_level': 'Високий' if prediction_proba > 0.7 else 'Середній' if
prediction_proba > 0.3 else 'Низький'
        })
    except Exception as e:
        return jsonify({
            'success': False,
            'message': f'Помилка прогнозування: {str(e)}'
        }), 500
# Отримання історії
@app.route('/api/history', methods=['GET'])
def get_history():
    try:
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
        cursor.execute('''
            SELECT customer_id, gender, partner, contract, monthly_charges,
                   prediction, prediction_date, actual_churn
            FROM clients
            ORDER BY prediction_date DESC
        ''')
        clients = cursor.fetchall()
        conn.close()
        history = []
        for client in clients:
            # Якщо немає прогнозу, але є actual_churn, використаємо його для візуалізації
            if client[5] is None and client[7]:
                # Присвоюємо умовну ймовірність на основі actual_churn
                churn_prob = 0.8 if client[7] == 'Yes' else 0.2
            else:
                churn_prob = client[5] if client[5] is not None else 0.5
            history.append({
                'customer_id': client[0],
                'gender': client[1],
                'partner': client[2],
                'contract': client[3],
                'monthly_charges': client[4],
                'churn_probability': churn_prob,
                'prediction_date': client[6],
                'churn_prediction': 'Так' if churn_prob > 0.5 else 'Ні',
                'actual_churn': client[7]
            })
        return jsonify({
            'success': True,
            'history': history
        })
    except Exception as e:
        return jsonify({
            'success': False,
            'message': f'Помилка отримання історії: {str(e)}'
        }), 500
# Отримання даних клієнта за ID
@app.route('/api/client/<customer_id>', methods=['GET'])
def get_client(customer_id):
    try:
```

```
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
        cursor.execute('''
            SELECT customer_id, gender, senior_citizen, partner, dependents, tenure,
                   phone_service, multiple_lines, internet_service, online_security,
                   online_backup, device_protection, tech_support, streaming_tv,
                   streaming_movies, contract, paperless_billing, payment_method,
                   monthly_charges, total_charges
            FROM clients
            WHERE customer_id = ?
        ''', (customer_id,))
        client = cursor.fetchone()
        conn.close()
        if client:
            return jsonify({
                'success': True,
                'client': {
                    'customer_id': client[0],
                    'gender': client[1],
                    'senior_citizen': client[2],
                    'partner': client[3],
                    'dependents': client[4],
                    'tenure': client[5],
                    'phone_service': client[6],
                    'multiple_lines': client[7],
                    'internet_service': client[8],
                    'online_security': client[9],
                    'online_backup': client[10],
                    'device_protection': client[11],
                    'tech_support': client[12],
                    'streaming_tv': client[13],
                    'streaming_movies': client[14],
                    'contract': client[15],
                    'paperless_billing': client[16],
                    'payment_method': client[17],
                    'monthly_charges': client[18],
                    'total_charges': client[19]
                }
            })
        else:
            return jsonify({
                'success': False,
                'message': 'Клієнт не знайдений'
            }), 404
    except Exception as e:
        return jsonify({
            'success': False,
            'message': f'Помилка отримання даних клієнта: {str(e)}'
        }), 500
# Отримання списку всіх клієнтів
@app.route('/api/clients', methods=['GET'])
def get_clients():
    try:
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()

        cursor.execute('''
            SELECT customer_id FROM clients ORDER BY customer_id
        ''')
        clients = cursor.fetchall()
        conn.close()

        return jsonify({
            'success': True,
            'clients': [{'id': client[0]} for client in clients]
```

```
        })
    except Exception as e:
        return jsonify({
            'success': False,
            'message': f'Помилка отримання списку клієнтів: {str(e)}'
        }), 500
# Перевірка статусу моделі
@app.route('/api/model_status', methods=['GET'])
def model_status():
    model_exists = os.path.exists(MODEL_PATH)
    return jsonify({
        'model_exists': model_exists,
        'model_path': MODEL_PATH if model_exists else None
    })
if __name__ == '__main__':
    init_db()
    print("🚀 Сервер запущено на порту 5001")
    app.run(debug=True, port=5001, host='0.0.0.0')
```

A.2 Client.py

```
import streamlit as st
import requests
import pandas as pd
from datetime import datetime
import json
# Налаштування сторінки
st.set_page_config(
    page_title="Прогнозування відтоку клієнтів",
    page_icon="📊",
    layout="wide"
)
# Кастомні стилі CSS
st.markdown("""
<style>
    /* Основні кольори */
    :root {
        --primary-color: #004D47;
        --secondary-color: #128277;
        --accent-color: #52958B;
        --light-bg: #B9C4C9;
        --white-bg: #FFFFFF;
        --success-color: #28a745;
        --warning-color: #ffc107;
        --danger-color: #dc3545;
    }
    /* Стилізація заголовків */
    h1, h2, h3 {
        color: var(--primary-color);
    }
    /* Кастомні кнопки меню */
    .menu-button {
        background-color: var(--secondary-color);
        color: white;
        border: none;
        padding: 12px 24px;
        text-align: center;
        text-decoration: none;
        display: inline-block;
```

```
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    border-radius: 8px;
    transition: all 0.3s;
    width: 100%;
}
.menu-button:hover {
    background-color: var(--primary-color);
    transform: translateY(-2px);
    box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}
.menu-button.active {
    background-color: var(--primary-color);
    box-shadow: inset 0 2px 4px rgba(0,0,0,0.2);
}
/* Стилізація метрик */
div[data-testid="metric-container"] {
    background-color: var(--white-bg);
    border: 1px solid var(--light-bg);
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    transition: transform 0.2s;
}
div[data-testid="metric-container"]:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 8px rgba(0,0,0,0.15);
}
/* Стилізація інфо-блоків */
.stAlert {
    border-radius: 10px;
    border-left: 5px solid;
    padding: 15px;
}
/* Покращення таблиць */
.dataframe {
    border-radius: 10px;
    overflow: hidden;
}
/* Стилізація форм */
.stButton > button {
    background-color: var(--secondary-color);
    color: white;
    border-radius: 8px;
    border: none;
    padding: 10px 24px;
    font-weight: 600;
    transition: all 0.3s;
}
.stButton > button:hover {
    background-color: var(--primary-color);
    transform: translateY(-2px);
    box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}
/* Primary button */
.stButton > button[kind="primary"] {
    background-color: var(--accent-color);
}
.stButton > button[kind="primary"]:hover {
    background-color: var(--secondary-color);
}
/* Стилізація select boxes */
.stSelectbox > div > div {
    border-radius: 8px;
```

```
        border-color: var(--light-bg);
    }
    /* Карточки */
    .info-card {
        background-color: var(--white-bg);
        border-radius: 12px;
        padding: 20px;
        box-shadow: 0 2px 8px rgba(0,0,0,0.1);
        margin-bottom: 20px;
        border-left: 4px solid var(--secondary-color);
    }
    /* Результати аналізу */
    .result-card {
        background: linear-gradient(135deg, var(--white-bg) 0%, var(--light-bg) 100%);
        border-radius: 15px;
        padding: 25px;
        margin: 20px 0;
        box-shadow: 0 4px 15px rgba(0,0,0,0.1);
    }
    /* Прогрес-бар */
    .stProgress > div > div > div {
        background-color: var(--secondary-color);
    }
</style>
""", unsafe_allow_html=True)
# URL сервера
API_URL = "http://localhost:5001/api"
# Функція для створення кнопок меню
def create_menu():
    if 'active_page' not in st.session_state:
        st.session_state.active_page = "🏠 Головна"
    pages = ["🏠 Головна", "📝 Новий клієнт", "📊 Історія", "🎓 Навчання моделі"]
    cols = st.columns(len(pages))
    for idx, (col, page) in enumerate(zip(cols, pages)):
        with col:
            if st.button(
                page,
                key=f"menu_{idx}",
                use_container_width=True,
                type="primary" if st.session_state.active_page == page else "secondary"
            ):
                st.session_state.active_page = page
                st.rerun()
    return st.session_state.active_page
# Заголовок
st.markdown("""
<h1 style='text-align: center; color: #004D47; margin-bottom: 0;'>
    🧠 Система прогнозування відтоку клієнтів
</h1>
<p style='text-align: center; color: #52958B; font-size: 18px; margin-top: 0;'>
    Аналітика на основі машинного навчання
</p>
""", unsafe_allow_html=True)
st.markdown("---")
# Меню
page = create_menu()
# Головна сторінка
if page == "🏠 Головна":
    st.markdown("<h2 style='color: #004D47;'>Ласкаво просимо!</h2>",
unsafe_allow_html=True)
    col1, col2, col3 = st.columns(3)
    with col1:
        st.markdown("""
        <div class='info-card'>
```

```
            <h3 style='color: #128277;'>📈 Про систему</h3>
            <p>Система використовує машинне навчання для прогнозування ймовірності
відтоку клієнтів телекомунікаційної компанії</p>
        </div>
        """, unsafe_allow_html=True)
    with col2:
        st.markdown("""
        <div class='info-card'>
            <h3 style='color: #128277;'>🎯 Точність</h3>
            <p>Модель досягає точності понад 80% на тестових даних</p>
        </div>
        """, unsafe_allow_html=True)
    with col3:
        st.markdown("""
        <div class='info-card'>
            <h3 style='color: #128277;'>⚡ Швидкість</h3>
            <p>Миттєвий прогноз для нових клієнтів</p>
        </div>
        """, unsafe_allow_html=True)
    # Перевірка статусу моделі
    st.markdown("<h3 style='margin-top: 30px;'>Статус системи</h3>",
unsafe_allow_html=True)
    try:
        response = requests.get(f"{API_URL}/model_status")
        if response.status_code == 200:
            status = response.json()
            if status['model_exists']:
                st.success("✅ Модель готова до використання")
            else:
                st.error("❌ Модель не знайдена. Перейдіть до розділу 'Навчання моделі'")
    except:
        st.error("⚠️ Сервер недоступний. Переконайтеся, що server.py запущено!")
# Форма нового клієнта
elif page == "📝 Новий клієнт":
    st.markdown("<h2 style='color: #004D47;'>Аналіз нового клієнта</h2>",
unsafe_allow_html=True)
    # Отримання списку існуючих клієнтів
    existing_clients = ["Новий клієнт"]
    try:
        response = requests.get(f"{API_URL}/clients")
        if response.status_code == 200:
            data = response.json()
            if data['success']:
                existing_clients.extend([client['id'] for client in data['clients']])
    except:
        pass
    # Вибір клієнта для автозаповнення
    selected_client = st.selectbox(
        "🔍 Оберіть клієнта для автозаповнення або створіть нового:",
        existing_clients,
        help="Виберіть існуючого клієнта щоб автоматично заповнити його дані"
    )
    # Ініціалізація значень форми
    if 'form_data' not in st.session_state:
        st.session_state.form_data = {}
    # Якщо вибрано існуючого клієнта - завантажуємо його дані
    if selected_client != "Новий клієнт":
        try:
            response = requests.get(f"{API_URL}/client/{selected_client}")
            if response.status_code == 200:
                client_data = response.json()['client']
                st.session_state.form_data = client_data
                st.success(f"✅ Дані клієнта {selected_client} завантажено")
        except:
```

```
            st.error("❌ Помилка завантаження даних клієнта")
    with st.form("client_form", clear_on_submit=False):
        # Використовуємо tabs замість колонок для кращої організації
        tab1, tab2, tab3 = st.tabs(["👤 Основна інформація", "📡 Послуги", "💳 Контракт
та оплата"])
        with tab1:
            col1, col2 = st.columns(2)
            with col1:
                customer_id = st.text_input(
                    "ID клієнта*",
                    value=st.session_state.form_data.get('customer_id', ''),
                    placeholder="Наприклад: CL001"
                )
                gender = st.selectbox(
                    "Стать",
                    ["Чоловіча", "Жіноча"],
                    index=0 if st.session_state.form_data.get('gender', 'Male') == 'Male'
else 1
                )
                senior_citizen = st.checkbox(
                    "Пенсіонер (65+ років)",
                    value=bool(st.session_state.form_data.get('senior_citizen', 0))
                )
            with col2:
                partner = st.selectbox(
                    "Наявність партнера",
                    ["Так", "Ні"],
                    index=0 if st.session_state.form_data.get('partner', 'No') == 'Yes'
else 1
                )
                dependents = st.selectbox(
                    "Наявність утриманців",
                    ["Так", "Ні"],
                    index=0 if st.session_state.form_data.get('dependents', 'No') ==
'Yes' else 1
                )
                tenure = st.number_input(
                    "Термін співпраці (місяці)",
                    min_value=0,
                    max_value=100,
                    value=st.session_state.form_data.get('tenure', 12)
                )
        with tab2:
            # Допоміжна функція для індексу в selectbox
            def get_service_index(service, value, options):
                mapping = {
                    'Yes': 0, 'Так': 0,
                    'No': 1, 'Ні': 1,
                    'No phone service': 2, 'Немає телефону': 2,
                    'No internet service': 2, 'Немає інтернету': 2,
                    'DSL': 0,
                    'Fiber optic': 1, 'Оптоволокно': 1,
                    'No': 2, 'Немає': 2
                }
                return mapping.get(value, 0)
            col1, col2 = st.columns(2)
            with col1:
                phone_service = st.selectbox(
                    "Телефонні послуги",
                    ["Так", "Ні"],
                    index=0 if st.session_state.form_data.get('phone_service', 'Yes') ==
'Yes' else 1
                )
                ml_value = st.session_state.form_data.get('multiple_lines', 'No')
```

```
                ml_index = 0 if ml_value == 'Yes' else 2 if ml_value == 'No phone
service' else 1
                multiple_lines = st.selectbox("Декілька ліній", ["Так", "Ні", "Немає
телефону"], index=ml_index)
                is_value = st.session_state.form_data.get('internet_service', 'No')
                is_index = 0 if is_value == 'DSL' else 1 if is_value == 'Fiber optic'
else 2
                internet_service = st.selectbox("Інтернет", ["DSL", "Оптоволокно",
"Немає"], index=is_index)
                # Для інтернет-послуг
                def get_internet_service_index(value):
                    if value == 'Yes': return 0
                    elif value == 'No': return 1
                    else: return 2
                online_security = st.selectbox(
                    "Онлайн безпека",
                    ["Так", "Ні", "Немає інтернету"],

index=get_internet_service_index(st.session_state.form_data.get('online_security', 'No'))
                )
                online_backup = st.selectbox(
                    "Онлайн резервування",
                    ["Так", "Ні", "Немає інтернету"],

index=get_internet_service_index(st.session_state.form_data.get('online_backup', 'No'))
                )
            with col2:
                device_protection = st.selectbox(
                    "Захист пристроїв",
                    ["Так", "Ні", "Немає інтернету"],

index=get_internet_service_index(st.session_state.form_data.get('device_protection',
'No'))
                )
                tech_support = st.selectbox(
                    "Технічна підтримка",
                    ["Так", "Ні", "Немає інтернету"],

index=get_internet_service_index(st.session_state.form_data.get('tech_support', 'No'))
                )
                streaming_tv = st.selectbox(
                    "Стрімінг ТВ",
                    ["Так", "Ні", "Немає інтернету"],

index=get_internet_service_index(st.session_state.form_data.get('streaming_tv', 'No'))
                )
                streaming_movies = st.selectbox(
                    "Стрімінг фільмів",
                    ["Так", "Ні", "Немає інтернету"],

index=get_internet_service_index(st.session_state.form_data.get('streaming_movies',
'No'))
                )
        with tab3:
            col1, col2 = st.columns(2)
            with col1:
                # Мапінг контрактів
                contract_value = st.session_state.form_data.get('contract', 'Month-to-
month')
                contract_index = 0 if contract_value == 'Month-to-month' else 1 if
contract_value == 'One year' else 2
                contract = st.selectbox("Тип контракту", ["Щомісячний", "Річний",
"Дворічний"], index=contract_index)
                paperless_billing = st.selectbox(
                    "Електронні рахунки",
```

```
                    ["Так", "Ні"],
                    index=0 if st.session_state.form_data.get('paperless_billing', 'Yes')
== 'Yes' else 1
                )
                # Мапінг способів оплати
                pm_value = st.session_state.form_data.get('payment_method', 'Electronic
check')
                pm_index = 0
                if pm_value == 'Mailed check': pm_index = 1
                elif pm_value == 'Bank transfer (automatic)': pm_index = 2
                elif pm_value == 'Credit card (automatic)': pm_index = 3
                payment_method = st.selectbox(
                    "Спосіб оплати",
                    ["Електронний чек", "Поштовий чек", "Банківський переказ", "Кредитна
картка"],
                    index=pm_index
                )
            with col2:
                monthly_charges = st.number_input(
                    "Щомісячна оплата ($)",
                    min_value=0.0,
                    value=float(st.session_state.form_data.get('monthly_charges', 50.0))
                )
                total_charges = st.number_input(
                    "Загальна сума платежів ($)",
                    min_value=0.0,
                    value=float(st.session_state.form_data.get('total_charges', 600.0))
                )
        st.markdown("<br>", unsafe_allow_html=True)
        submitted = st.form_submit_button("🔮 Прогнозувати відтік", type="primary",
use_container_width=True)
        if submitted:
            if not customer_id:
                st.error("❌ Введіть ID клієнта!")
            else:
                # Підготовка даних для API
                client_data = {
                    'customer_id': customer_id,
                    'gender': 'Male' if gender == "Чоловіча" else 'Female',
                    'senior_citizen': 1 if senior_citizen else 0,
                    'partner': 'Yes' if partner == "Так" else 'No',
                    'dependents': 'Yes' if dependents == "Так" else 'No',
                    'tenure': tenure,
                    'phone_service': 'Yes' if phone_service == "Так" else 'No',
                    'multiple_lines': 'No' if multiple_lines == "Ні" else 'Yes' if
multiple_lines == "Так" else 'No phone service',
                    'internet_service': 'DSL' if internet_service == "DSL" else 'Fiber
optic' if internet_service == "Оптоволокно" else 'No',
                    'online_security': 'Yes' if online_security == "Так" else 'No' if
online_security == "Ні" else 'No internet service',
                    'online_backup': 'Yes' if online_backup == "Так" else 'No' if
online_backup == "Ні" else 'No internet service',
                    'device_protection': 'Yes' if device_protection == "Так" else 'No' if
device_protection == "Ні" else 'No internet service',
                    'tech_support': 'Yes' if tech_support == "Так" else 'No' if
tech_support == "Ні" else 'No internet service',
                    'streaming_tv': 'Yes' if streaming_tv == "Так" else 'No' if
streaming_tv == "Ні" else 'No internet service',
                    'streaming_movies': 'Yes' if streaming_movies == "Так" else 'No' if
streaming_movies == "Ні" else 'No internet service',
                    'contract': 'Month-to-month' if contract == "Щомісячний" else 'One
year' if contract == "Річний" else 'Two year',
                    'paperless_billing': 'Yes' if paperless_billing == "Так" else 'No',
                    'payment_method': 'Electronic check' if payment_method ==
"Електронний чек" else
```

```
                                    'Mailed check' if payment_method == "Поштовий чек"
else
                                    'Bank transfer (automatic)' if payment_method ==
"Банківський переказ" else
                                    'Credit card (automatic)',
                    'monthly_charges': monthly_charges,
                    'total_charges': total_charges
                }
                # Відправка запиту
                with st.spinner('🔄 Аналізуємо дані клієнта...'):
                    try:
                        response = requests.post(f"{API_URL}/predict", json=client_data)
                        if response.status_code == 200:
                            result = response.json()
                            # Відображення результатів
                            st.markdown("<div class='result-card'>",
unsafe_allow_html=True)
                            st.markdown("<h3 style='color: #004D47; text-align:
center;'>📊 Результати аналізу</h3>", unsafe_allow_html=True)
                            col1, col2, col3 = st.columns(3)
                            with col1:
                                prediction_color = "#dc3545" if result['prediction'] ==
"Так" else "#28a745"
                                st.markdown(f"""
                                <div style='text-align: center; padding: 20px;
background-color: white; border-radius: 10px; box-shadow: 0 2px 4px rgba(0,0,0,0.1);'>
                                    <h4 style='color: #52958B; margin-bottom:
10px;'>Прогноз відтоку</h4>
                                    <h2 style='color: {prediction_color}; margin:
0;'>{result['prediction']}</h2>
                                </div>
                                """, unsafe_allow_html=True)
                            with col2:
                                prob_color = "#dc3545" if result['probability'] > 0.7
else "#ffc107" if result['probability'] > 0.3 else "#28a745"
                                st.markdown(f"""
                                <div style='text-align: center; padding: 20px;
background-color: white; border-radius: 10px; box-shadow: 0 2px 4px rgba(0,0,0,0.1);'>
                                    <h4 style='color: #52958B; margin-bottom:
10px;'>Ймовірність відтоку</h4>
                                    <h2 style='color: {prob_color}; margin:
0;'>{result['probability']:.1%}</h2>
                                </div>
                                """, unsafe_allow_html=True)
                            with col3:
                                risk_color = "#dc3545" if result['risk_level'] ==
"Високий" else "#ffc107" if result['risk_level'] == "Середній" else "#28a745"
                                risk_icon = "⚠️" if result['risk_level'] == "Високий"
else "⚡" if result['risk_level'] == "Середній" else "✅"
                                st.markdown(f"""
                                <div style='text-align: center; padding: 20px;
background-color: white; border-radius: 10px; box-shadow: 0 2px 4px rgba(0,0,0,0.1);'>
                                    <h4 style='color: #52958B; margin-bottom:
10px;'>Рівень ризику</h4>
                                    <h2 style='color: {risk_color}; margin:
0;'>{risk_icon} {result['risk_level']}</h2>
                                </div>
                                """, unsafe_allow_html=True)
                            st.markdown("</div>", unsafe_allow_html=True)
                            # Рекомендації
                            st.markdown("<h3 style='margin-top: 30px;'>💡
Рекомендації</h3>", unsafe_allow_html=True)
                            if result['probability'] > 0.7:
                                st.error("""
```

```
                            **Високий ризик відтоку!** Рекомендовані дії:
                            - 📞 Негайно зв'язатися з клієнтом
                            - 🎁 Запропонувати спеціальні умови або знижку
                            - 📋 Перевірити наявність скарг
                            - 🔄 Розглянути можливість зміни тарифного плану
                            """)
                        elif result['probability'] > 0.3:
                            st.warning("""
                            **Середній ризик відтоку.** Рекомендовані дії:
                            - 📧 Відправити персоналізовану пропозицію
                            - 📊 Моніторити активність клієнта
                            - 🎯 Включити в програму лояльності
                            """)
                        else:
                            st.success("""
                            **Низький ризик відтоку.** Рекомендовані дії:
                            - ✅ Продовжувати надавати якісний сервіс
                            - 📈 Пропонувати додаткові послуги
                            - 🌟 Використовувати як приклад успішної роботи
                            """)
                    else:
                        st.error(f"❌ Помилка: {response.json().get('message',
'Невідома помилка')}")
                except Exception as e:
                    st.error(f"❌ Помилка з'єднання з сервером: {str(e)}")
# Історія
elif page == "📊 Історія":
    st.markdown("<h2 style='color: #004D47;'>Історія прогнозів</h2>",
unsafe_allow_html=True)
    # Отримання історії
    try:
        response = requests.get(f"{API_URL}/history")
        if response.status_code == 200:
            data = response.json()
            if data['success'] and data['history']:
                df = pd.DataFrame(data['history'])
                # Фільтри
                col1, col2, col3 = st.columns(3)
                with col1:
                    churn_filter = st.selectbox(
                        "Фільтр за прогнозом",
                        ["Всі", "Відтік", "Залишиться"]
                    )
                # Застосування фільтрів
                if churn_filter == "Відтік":
                    df = df[df['churn_prediction'] == 'Так']
                elif churn_filter == "Залишиться":
                    df = df[df['churn_prediction'] == 'Ні']
                # Статистика
                st.markdown("<div style='margin-top: 20px;'>", unsafe_allow_html=True)
                col1, col2, col3, col4 = st.columns(4)
                with col1:
                    st.metric("Всього клієнтів", len(df))
                with col2:
                    churn_count = len(df[df['churn_prediction'] == 'Так'])
                    st.metric("Прогноз відтоку", churn_count)
                with col3:
                    avg_prob = df['churn_probability'].mean()
                    st.metric("Середня ймовірність", f"{avg_prob:.1%}")
                with col4:
                    high_risk = len(df[df['churn_probability'] > 0.7])
                    st.metric("Високий ризик", high_risk)
                st.markdown("</div>", unsafe_allow_html=True)
```

```
                # Додаткова колонка для статусу прогнозу
                df['Статус'] = df.apply(lambda row:
                    '🔍 Не перевірено' if pd.isna(row['prediction_date']) or
row['prediction_date'] == ''
                    else '✅ Проаналізовано', axis=1
                )
                # Таблиця
                st.markdown("<h3 style='margin-top: 30px;'>📋 Детальна інформація</h3>",
unsafe_allow_html=True)
                st.dataframe(
                    df[[
                        'customer_id', 'gender', 'contract',
                        'monthly_charges', 'churn_probability',
                        'churn_prediction', 'Статус'
                    ]].rename(columns={
                        'customer_id': 'ID клієнта',
                        'gender': 'Стать',
                        'contract': 'Контракт',
                        'monthly_charges': 'Щомісячна оплата',
                        'churn_probability': 'Ймовірність відтоку',
                        'churn_prediction': 'Прогноз'
                    }),
                    use_container_width=True,
                    height=400
                )
                # Графіки
                st.markdown("<h3 style='margin-top: 30px;'>📈 Візуалізація</h3>",
unsafe_allow_html=True)
                col1, col2 = st.columns(2)
                with col1:
                    st.bar_chart(
                        df.groupby('churn_prediction').size(),
                        use_container_width=True,
                        color="#128277"
                    )
                    st.caption("Розподіл прогнозів")
                with col2:
                    risk_levels = pd.cut(
                        df['churn_probability'],
                        bins=[0, 0.3, 0.7, 1.0],
                        labels=['Низький', 'Середній', 'Високий']
                    )
                    st.bar_chart(
                        risk_levels.value_counts(),
                        use_container_width=True,
                        color="#52958B"
                    )
                    st.caption("Розподіл за рівнем ризику")
                # Інформація про seed-дані
                st.info("""
                💡 **Примітка**: Клієнти з ID HIGH_*, LOYAL_* та MED_* - це тестові дані
з попередньо визначеними профілями:
                - HIGH_* - клієнти з високим ризиком відтоку
                - LOYAL_* - лояльні клієнти
                - MED_* - клієнти із середнім ризиком
                """)
            else:
                st.info("📭 Історія порожня. Спочатку зробіть прогноз для клієнта.")
        else:
            st.error("❌ Помилка отримання історії")
    except Exception as e:
        st.error(f"❌ Помилка з'єднання з сервером: {str(e)}")
# Навчання моделі
elif page == "🎓 Навчання моделі":
```

```
    st.markdown("<h2 style='color: #004D47;'>Навчання моделі</h2>",
unsafe_allow_html=True)
    st.markdown("""
    <div class='info-card'>
        <h3 style='color: #128277;'>📊 Про навчання</h3>
        <ul>
            <li>Модель використовує алгоритм Random Forest</li>
            <li>Датасет: Telco Customer Churn від IBM</li>
            <li>Розмір: 7,043 клієнтів з 21 атрибутом</li>
            <li>Очікувана точність: 80-85%</li>
        </ul>
    </div>
    """, unsafe_allow_html=True)
    # Перевірка статусу моделі
    try:
        response = requests.get(f"{API_URL}/model_status")
        if response.status_code == 200:
            status = response.json()
            if status['model_exists']:
                st.success("✅ Модель вже навчена та готова до використання")
                if st.button("🔄 Перенавчити модель", use_container_width=True):
                    with st.spinner('⏳ Навчання моделі... Це може зайняти кілька
хвилин'):
                        response = requests.post(f"{API_URL}/train_model")
                        if response.status_code == 200:
                            result = response.json()
                            st.success(f"✅ {result['message']}")
                        else:
                            st.error(f"❌ {response.json().get('message', 'Помилка
навчання')}")
            else:
                st.warning("⚠️ Модель ще не навчена")
                if st.button("🚀 Навчити модель", type="primary",
use_container_width=True):
                    with st.spinner('⏳ Навчання моделі... Це може зайняти кілька
хвилин'):
                        response = requests.post(f"{API_URL}/train_model")
                        if response.status_code == 200:
                            result = response.json()
                            st.success(f"✅ {result['message']}")
                        else:
                            st.error(f"❌ {response.json().get('message', 'Помилка
навчання')}")
    except Exception as e:
        st.error(f"❌ Помилка з'єднання з сервером: {str(e)}")
    # Інформація про модель
    st.markdown("<h3 style='margin-top: 40px;'>🔍 Важливі фактори для
прогнозування</h3>", unsafe_allow_html=True)
    factors = [
        ("Тип контракту", "щомісячні контракти мають найвищий відтік", "#dc3545"),
        ("Термін співпраці", "нові клієнти частіше йдуть", "#ffc107"),
        ("Спосіб оплати", "електронні чеки корелюють з відтоком", "#fd7e14"),
        ("Загальні витрати", "високі щомісячні платежі збільшують ризик", "#20c997"),
        ("Послуги підтримки", "відсутність техпідтримки підвищує відтік", "#6610f2")
    ]
    for i, (factor, description, color) in enumerate(factors, 1):
        st.markdown(f"""
        <div style='background-color: white; padding: 15px; margin: 10px 0; border-
radius: 10px; border-left: 4px solid {color}; box-shadow: 0 2px 4px rgba(0,0,0,0.1);'>
            <strong>{i}. {factor}</strong> - {description}
        </div>
        """, unsafe_allow_html=True)
# Футер
```

```
st.markdown("---")
st.markdown(
    """
    <div style='text-align: center; color: #52958B; padding: 20px 0;'>
        <p style='margin: 0;'>Система прогнозування відтоку клієнтів v1.0</p>
        <p style='margin: 0; font-size: 14px;'>Розроблено з ♡ для телеком-компанії</p>
    </div>
    """,
    unsafe_allow_html=True
)
```

A.3 Seed.py

```
import sqlite3
import random
from datetime import datetime, timedelta
# Налаштування
DB_NAME = 'churn_prediction.db'
# Генератори даних
NAMES = ['Олександр', 'Марія', 'Петро', 'Наталія', 'Андрій', 'Оксана', 'Іван', 'Тетяна',
'Максим', 'Юлія']
SURNAMES = ['Шевченко', 'Коваленко', 'Бондаренко', 'Ткаченко', 'Кравченко', 'Олійник',
'Мельник', 'Павленко']
# Профілі клієнтів
def generate_high_risk_client(client_id):
    """Клієнт з високим ризиком відтоку"""
    return {
        'customer_id': client_id,
        'gender': random.choice(['Male', 'Female']),
        'senior_citizen': random.choice([0, 1]),
        'partner': 'No',
        'dependents': 'No',
        'tenure': random.randint(1, 6),  # Новий клієнт
        'phone_service': 'Yes',
        'multiple_lines': 'No',
        'internet_service': 'Fiber optic',  # Дорогий інтернет
        'online_security': 'No',
        'online_backup': 'No',
        'device_protection': 'No',
        'tech_support': 'No',
        'streaming_tv': 'Yes',
        'streaming_movies': 'Yes',
        'contract': 'Month-to-month',  # Місячний контракт
        'paperless_billing': 'Yes',
        'payment_method': 'Electronic check',  # Найгірший спосіб оплати
        'monthly_charges': random.uniform(80, 120),  # Високі платежі
        'total_charges': lambda mc, t: mc * t,
        'actual_churn': 'Yes'
    }
def generate_loyal_client(client_id):
    """Лояльний клієнт"""
    return {
        'customer_id': client_id,
        'gender': random.choice(['Male', 'Female']),
        'senior_citizen': random.choice([0, 1]),
        'partner': 'Yes',
        'dependents': 'Yes',
        'tenure': random.randint(48, 72),  # Довгий термін
        'phone_service': 'Yes',
```

```
        'multiple_lines': 'Yes',
        'internet_service': 'DSL',  # Дешевший інтернет
        'online_security': 'Yes',
        'online_backup': 'Yes',
        'device_protection': 'Yes',
        'tech_support': 'Yes',
        'streaming_tv': 'No',
        'streaming_movies': 'No',
        'contract': 'Two year',  # Дворічний контракт
        'paperless_billing': 'No',
        'payment_method': 'Bank transfer (automatic)',  # Автоматична оплата
        'monthly_charges': random.uniform(30, 60),  # Помірні платежі
        'total_charges': lambda mc, t: mc * t,
        'actual_churn': 'No'
    }
def generate_medium_risk_client(client_id):
    """Клієнт із середнім ризиком"""
    return {
        'customer_id': client_id,
        'gender': random.choice(['Male', 'Female']),
        'senior_citizen': random.choice([0, 1]),
        'partner': random.choice(['Yes', 'No']),
        'dependents': random.choice(['Yes', 'No']),
        'tenure': random.randint(12, 36),
        'phone_service': 'Yes',
        'multiple_lines': random.choice(['Yes', 'No']),
        'internet_service': random.choice(['DSL', 'Fiber optic']),
        'online_security': random.choice(['Yes', 'No']),
        'online_backup': random.choice(['Yes', 'No']),
        'device_protection': random.choice(['Yes', 'No']),
        'tech_support': random.choice(['Yes', 'No']),
        'streaming_tv': random.choice(['Yes', 'No']),
        'streaming_movies': random.choice(['Yes', 'No']),
        'contract': 'One year',
        'paperless_billing': random.choice(['Yes', 'No']),
        'payment_method': random.choice(['Credit card (automatic)', 'Mailed check']),
        'monthly_charges': random.uniform(50, 80),
        'total_charges': lambda mc, t: mc * t,
        'actual_churn': random.choice(['Yes', 'No'])
    }
def seed_database():
    """Заповнення БД тестовими даними"""
    # Підключення до БД
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()
    # Очищення існуючих даних
    cursor.execute("DELETE FROM clients")
    cursor.execute("DELETE FROM prediction_history")
    print("🧹 База даних очищена")
    # Генерація клієнтів
    clients = []
    # 3 клієнти з високим ризиком
    for i in range(3):
        name = f"{random.choice(NAMES)} {random.choice(SURNAMES)}"
        client_id = f"HIGH_{i+1:03d}"
        client = generate_high_risk_client(client_id)
        clients.append(client)
        print(f"🔴 Створено клієнта з високим ризиком: {client_id} - {name}")
    # 4 лояльних клієнти
    for i in range(4):
        name = f"{random.choice(NAMES)} {random.choice(SURNAMES)}"
        client_id = f"LOYAL_{i+1:03d}"
        client = generate_loyal_client(client_id)
        clients.append(client)
        print(f"🟢 Створено лояльного клієнта: {client_id} - {name}")
```

```
    # 3 клієнти із середнім ризиком
    for i in range(3):
        name = f"{random.choice(NAMES)} {random.choice(SURNAMES)}"
        client_id = f"MED_{i+1:03d}"
        client = generate_medium_risk_client(client_id)
        clients.append(client)
        print(f"🟡 Створено клієнта із середнім ризиком: {client_id} - {name}")
    # Вставка в БД
    for client in clients:
        # Розрахунок total_charges
        total_charges = client['total_charges'](client['monthly_charges'],
client['tenure'])
        cursor.execute('''
            INSERT INTO clients
            (customer_id, gender, senior_citizen, partner, dependents, tenure,
             phone_service, multiple_lines, internet_service, online_security,
             online_backup, device_protection, tech_support, streaming_tv,
             streaming_movies, contract, paperless_billing, payment_method,
             monthly_charges, total_charges, actual_churn, prediction_date)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        ''', (
            client['customer_id'], client['gender'], client['senior_citizen'],
            client['partner'], client['dependents'], client['tenure'],
            client['phone_service'], client['multiple_lines'],
client['internet_service'],
            client['online_security'], client['online_backup'],
client['device_protection'],
            client['tech_support'], client['streaming_tv'], client['streaming_movies'],
            client['contract'], client['paperless_billing'], client['payment_method'],
            client['monthly_charges'], total_charges, client['actual_churn'],
            datetime.now() - timedelta(days=random.randint(0, 30))
        ))
```