МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

Онанко Вадим Володимирович

Прогнозування успішності запуску продукту на ринку за допомогою класифікаційних моделей / Predicting the success of a product launch on the market using classification models

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-41
Онанко В.В

Науковий керівник:
к.т.н., доцент
Лендюк Т.В

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н.М. Васильків

Тернопіль – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Онанко Вадим Володимирович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Прогнозування успішності запуску продукту на ринку за допомогою класифікаційних моделей / Predicting the success of a product launch on the market using classification models
керівник роботи к.т.н., доцент, Лендюк Т.В
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.

2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:

– Проаналізувати предметну область краудфандингу та існуючі підходи до прогнозування.

– Виконати попередню обробку набору даних, включаючи заповнення пропусків, масштабування та кодування змінних.

– Розробити архітектуру програмного модуля прогнозування.

– Побудувати та порівняти декілька класифікаційних моделей.

– Провести оцінку точності та якості моделей за допомогою крос-валідації та відповідних метрик.

– Визначити найефективнішу модель і реалізувати алгоритм прогнозування для нових даних..

5. Перелік графічного матеріалу в роботі:

– Потік даних у програмному модулі прогнозування успішності кампанії Kickstarter відносно користувача.

– Середні показники тестової точності (Test Accuracy) для різних класифікаторів..

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент_____ Онанко В.В_____
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ Лендюк Т.В_____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Прогнозування успішності запуску продукту на ринку за допомогою класифікаційних моделей» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» написана обсягом у 70 сторінок і містить 23 ілюстрації, 3 таблиці та 25 використаних джерел. Метою роботи є розробка програмного модуля для прогнозування успішності запуску продукту на ринку на основі методів машинного навчання з використанням історичних даних платформи Kickstarter. У процесі роботи використано комплекс методів інтелектуального аналізу даних: логістична регресія, дерева рішень, Random Forest, Gradient Boosting, XGBoost та LightGBM, а також сучасні техніки попередньої обробки даних та балансування класів. Результати дослідження можуть бути використані у стартапах, аналітичних компаніях, венчурних фондах та на краудфандингових платформах для автоматизації процесу оцінювання шансів успіху нових продуктів.

Ключові слова: краудфандинг, машинне навчання, прогнозування, XGBoost, класифікаційна модель, Kickstarter, балансування класів, попередня обробка даних.

ANNOTATION

Qualification work on the topic "Predicting the success of a product launch on the market using classification models" for obtaining the bachelor's degree in specialty 122 "Computer Science" consists of 70 pages, contains 23 illustrations, 3 tables, and 25 references. The aim of the work is to develop a software module for predicting the success of a product launch on the market using machine learning methods based on historical Kickstarter data. The study employs a set of data mining methods: logistic regression, decision trees, Random Forest, Gradient Boosting, XGBoost, and LightGBM, as well as advanced data preprocessing and class balancing techniques. The research results can be used by startups, analytics companies, venture funds, and crowdfunding platforms to automate the evaluation of the success chances of new products.

Keywords: crowdfunding, machine learning, prediction, XGBoost, classification model, Kickstarter, class balancing, data preprocessing.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі значна кількість інноваційних продуктів запускається через краудфандингові платформи. Однак лише частина з них досягає успіху, що зумовлює потребу в інструментах попереднього прогнозування результатів кампанії ще до її запуску. Високий відсоток невдалих проєктів вказує на недостатній аналіз факторів успіху та відсутність підтримки у прийнятті рішень.

Інтелектуальні методи аналізу даних, зокрема моделі машинного навчання, мають значний потенціал для розв'язання цієї проблеми. Їхнє застосування дозволяє виявляти приховані залежності між ознаками проєкту та результатом кампанії, формуючи основу для точного прогнозування. Водночас потреба в автоматизованих і адаптивних рішеннях зростає разом із обсягами доступних даних.

Особливої актуальності набуває розробка програмного модуля, який здатен поєднати всі етапи — від збору й обробки даних до навчання моделі та отримання прогнозу — в єдиному інтегрованому рішенні. Такі системи можуть бути корисними не лише авторам проєктів, але й аналітикам, інвесторам і маркетологам для оцінки потенціалу продукту.

Таким чином, тема дослідження відповідає актуальним викликам цифрової економіки та розвитку платформи-орієнтованих ринків. Вона спрямована на практичну реалізацію інтелектуального інструменту для підтримки прийняття рішень у сфері інноваційного підприємництва.

Мета роботи — розробити програмний модуль для прогнозування успішності запуску продукту на ринку на основі методів машинного навчання з використанням історичних даних платформи Kickstarter.

Завдання дослідження:

1. Проаналізувати предметну область краудфандингу та існуючі підходи до прогнозування.

2. Виконати попередню обробку набору даних, включаючи заповнення пропусків, масштабування та кодування змінних.

3. Розробити архітектуру програмного модуля прогнозування.

4. Побудувати та порівняти декілька класифікаційних моделей.

5. Провести оцінку точності та якості моделей за допомогою крос-валідації та відповідних метрик.

6. Визначити найефективнішу модель і реалізувати алгоритм прогнозування для нових даних.

Предмет дослідження - процес прогнозування успішності запуску продукту на основі історичних даних про краудфандингові кампанії.

Об’єкт дослідження - інформаційна модель та програмний модуль прогнозування результату краудфандингового проєкту.

Методи дослідження. У процесі дослідження використано комплекс методів інтелектуального аналізу даних, зокрема класифікаційні моделі машинного навчання: логістичну регресію, дерева рішень, випадкові ліси, градієнтне підсилення, XGBoost та LightGBM. Для підвищення якості результатів застосовувались методи попередньої обробки даних (заповнення пропусків, масштабування, бінарне кодування, балансування класів за допомогою SMOTE), а також оцінка ефективності моделей на основі крос-валідації та метрик точності, F1-міри й ROC-AUC. Алгоритм побудовано як поетапну систему з використанням програмного конвеєра (pipeline), що забезпечує відтворюваність і стабільність результатів.

Апробація результатів дослідження здійснена в межах студентської науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR-2025), яка відбулася в місті Тернополі 27–29 травня 2025 року.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Краудфандинг як інструмент запуску продуктів

Краудфандинг (від англ. *crowd* — «натовп», *funding* — «фінансування») — це форма колективного фінансування, за якої велика кількість осіб робить невеликі внески на підтримку певного проєкту, ідеї чи ініціативи через спеціалізовані онлайн-платформи.

Цей механізм є альтернативою традиційним способам залучення капіталу, таким як інвестування, банківське кредитування або венчурне фінансування. Краудфандинг дозволяє авторам проєктів напряму взаємодіяти з потенційними споживачами або інвесторами.

Основна перевага краудфандингу полягає в демократизації доступу до фінансування: автор ідеї отримує можливість перевірити її життєздатність на відкритому ринку ще до фактичного запуску продукту.

Умовно, краудфандингові моделі можна класифікувати на чотири типи: на основі винагород (*reward-based*), пожертв (*donation-based*), акціонерного капіталу (*equity-based*) та боргових зобов’язань (*debt-based*). Найбільш поширеною є модель з винагородами.

Однією з найвідоміших платформ у сфері reward-based краудфандингу є Kickstarter. Вона була запущена у 2009 році в США та стала піонером глобального руху краудфінансування творчих та інноваційних проєктів.

Kickstarter (рисунок 1.1) спеціалізується на проектах у сферах мистецтва, дизайну, технологій, відеоігор, кіно, музики тощо. Автори проєктів встановлюють цільову суму фінансування та дедлайн, до якого повинні зібрати кошти.

Особливістю моделі Kickstarter є принцип «все або нічого» (*all-or-nothing*): фінансування надходить автору лише у разі досягнення цільової суми. Це стимулює авторів до активної промоції, а бекерів — до відповідального фінансування.

Успіх кампанії на платформі Kickstarter залежить від великої кількості чинників, які можна умовно поділити на внутрішні та зовнішні (рисунок 1.2).

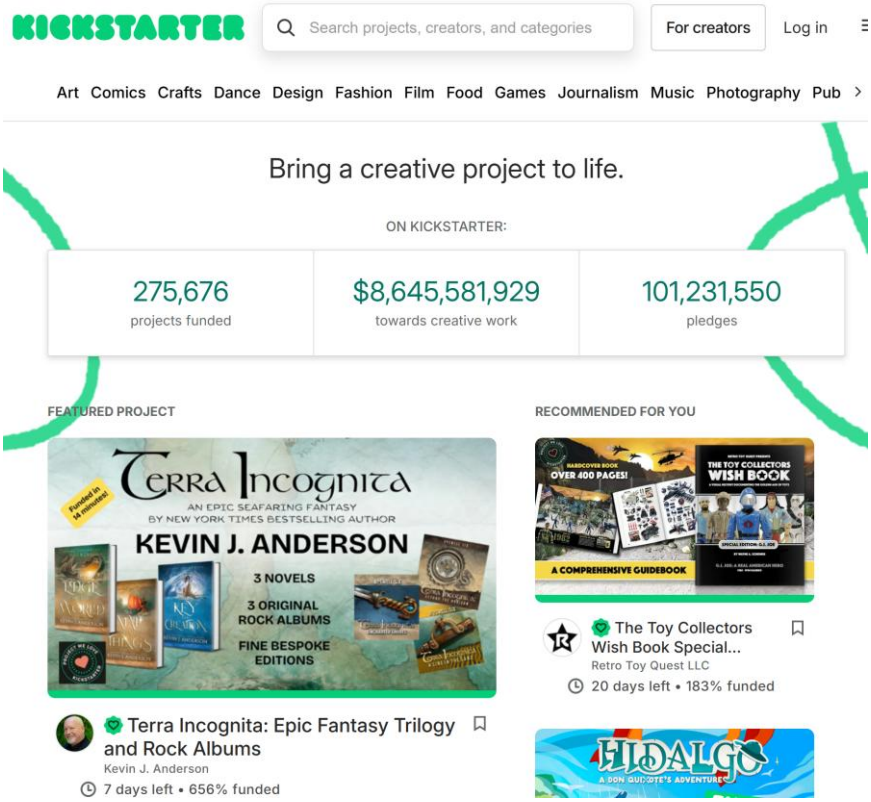

Рисунок 1.1 – Платформа Kickstarter

Рисунок 1.2 – Схема взаємодії учасників у моделі краудфандингу на платформі Kickstarter

До внутрішніх факторів належать характеристики самого проєкту: категорія, тривалість кампанії, опис, цільове фінансування, наявність відео, попередній досвід автора тощо.

Категорія проєкту значно впливає на його успішність. Наприклад, проєкти у сфері ігор, музики чи дизайну традиційно мають вищі шанси на фінансування, порівняно з літературними чи документальними ініціативами.

Тривалість кампанії також є критичним параметром. Зазвичай, надто короткі кампанії не встигають набрати обертів, а надто довгі втрачають динаміку підтримки. Ідеальна тривалість варіюється у межах 30–40 днів.

Наявність відео-презентації, візуального контенту та чітко сформульованих цілей підвищує довіру аудиторії та сприяє кращому сприйняттю ідеї.

До зовнішніх факторів належать сезонність запуску, день тижня публікації, країна походження автора, загальна економічна ситуація, культурні особливості та рівень конкуренції у відповідний період.

Наприклад, аналітика показує, що проєкти, запущені навесні або восени, мають вищі показники успішності, що може бути пов'язано з поведінковими моделями бекерів та споживчим циклом.

Важливим аспектом є рівень підтримки з боку спільноти: кількість поширень у соціальних мережах, попередня база підписників та активність коментарів на сторінці проєкту.

Таким чином, краудфандинг — це не лише інструмент залучення коштів, але й потужний інструмент перевірки ідеї, вивчення цільової аудиторії та створення першої бази клієнтів.

Його ефективність залежить від комплексного врахування як внутрішніх параметрів кампанії, так і зовнішніх умов, що створює підґрунтя для використання моделей інтелектуального аналізу даних для прогнозування успішності проєктів.

1.2 Аналіз існуючих підходів до прогнозування успішності

Упродовж останнього десятиліття проблема прогнозування успішності — як у контексті освітніх результатів, так і для запуску продуктів — набула значної уваги з боку дослідників у сфері машинного навчання. Із зростанням обсягів даних, які доступні для аналізу, постала потреба в автоматизованих системах, що здатні точно передбачати ймовірність успіху за сукупністю кількісних та якісних ознак. У різноманітних дослідженнях було протестовано численні алгоритми класифікації, серед яких особливо вирізняються ансамблеві методи, такі як Random Forest, Gradient Boosting, XGBoost та Voting Ensemble. Разом з тим, базові алгоритми, такі як Logistic Regression або Naive Bayes, досі застосовуються як опорні моделі для порівняння. У цьому підрозділі узагальнено результати низки наукових праць, у яких порівнюються ефективність різних класифікаційних методів у завданнях прогнозування.

Chen і Zhai провели порівняння семи методів машинного навчання, серед яких найбільш ефективним виявився Random Forest із точністю 85.7%, тоді як Naive Bayes продемонстрував найнижчий результат — 72.3% [1]. Автори підкреслюють важливість точного вибору атрибутів для покращення результатів.

У роботі Yıldız і Börekci моделі KNN, Decision Tree та Logistic Regression були протестовані на наборі даних зі 1000 студентів. Найвищу точність показала модель Decision Tree — 83%, в той час як Logistic Regression мала 76% [2]. Автори відзначають, що моделі демонстрували помітне зниження точності при обмеженій кількості вхідних ознак.

Holicza і Kiss розробили модель, яка враховує як онлайн, так і офлайн активність студентів. Їхній підхід дозволив досягти точності до 88.5% з використанням алгоритму Gradient Boosting [3]. Цікаво, що комбіновані дані забезпечили суттєво вищу точність, ніж при використанні лише однієї форми навчання.

Огляд Alyahyan і Düştegör демонструє, що моделі машинного навчання можуть досягати точності понад 80% при належному препроцесінгу даних та

відборі ознак. Автори рекомендують SVM для невеликих наборів даних і Random Forest — для більших [4].

Дослідження Ghorbani і Ghousi акцентує увагу на впливі методів ресемплінгу. SMOTE дозволив покращити точність моделей до 87%, порівняно з 78% при простому розподілі даних [5]. Найкращі результати були досягнуті при балансуванні класів успішних та неуспішних студентів.

Villar і de Andrade продемонстрували, що Random Forest перевершує класичну регресію у прогнозуванні вибуття з навчання. Модель досягла точності 89.2%, тоді як лінійна регресія — лише 74.8% [6].

У дослідженні Al-Alawi та співавт. було протестовано ансамблеві методи, включаючи XGBoost і LightGBM. Найвищу точність — 90.1% — продемонстрував XGBoost на даних студентів, які перебували на межі академічного відрахування [7].

Brohi та інші провели тестування кількох алгоритмів, зокрема SVM, Decision Tree, Random Forest і Naive Bayes. Найвищу точність у 84.3% показав Random Forest, тоді як Naive Bayes відстав із 69.7% [8].

Balcıoğlu і Artar виявили, що ансамблеві методи мають суттєву перевагу в стабільності та узагальненні моделей. Їхня модель із використанням Voting Ensemble досягла точності 91.4% [9].

В дослідженні Vijayalakshmi порівнюються різні алгоритми, включно з CART, SVM та Logistic Regression. Найвищу точність показав CART (82.5%), а Logistic Regression — 75% [10]. Також зазначено, що CART краще справляється з неконтрольованими змінними у великих наборах даних.

Аналіз сучасних підходів (таблиця 1.1) до прогнозування успішності демонструє домінування ансамблевих моделей над класичними алгоритмами. Моделі на основі випадкових лісів (Random Forest) показують стабільно високі результати, часто перевищуючи 85% точності. Значне покращення також досягається за рахунок об'єднання моделей (Voting Ensemble) або застосування підсилення (Gradient Boosting, XGBoost). Особливу роль відіграє якість

підготовки даних, зокрема балансування класів (методом SMOTE), що дозволяє підвищити точність у середньому на 8–10%.

Таблиця 1.1 – Порівняльна характеристика моделей класифікації у дослідженнях прогнозування успішності

Дослідники	Найкраща модель	Точність (%)	Найгірша модель	Точність (%)	Коментар
Chen & Zhai [1]	Random Forest	85.7	Naive Bayes	72.3	Підкреслено важливість вибору атрибутів
Yıldız & Börekci [2]	Decision Tree	83.0	Logistic Regression	76.0	Зниження точності при зменшенні числа ознак
Holicza & Kiss [3]	Gradient Boosting	88.5	—	—	Комбінування онлайн та офлайн-даних покращує результат
Alyahyan & Düştegör [4]	Random Forest / SVM	>80.0	—	—	SVM — для малих даних, RF — для великих
Ghorbani & Ghousi [5]	Random Forest (з SMOTE)	87.0	Без ресемплінгу	78.0	SMOTE значно покращує результати
Villar & de Andrade [6]	Random Forest	89.2	Linear Regression	74.8	RF суттєво перевершує регресію
Al-Alawi et al. [7]	XGBoost	90.1	—	—	Тестування на даних студентів на межі відрахування
Brohi et al. [8]	Random Forest	84.3	Naive Bayes	69.7	RF перевищує SVM та Naive Bayes
Balcıoğlu & Artar [9]	Voting Ensemble	91.4	—	—	Висока стабільність та узагальненість
Vijayalakshmi [10]	CART	82.5	Logistic Regression	75.0	CART краще працює з великими наборами змінних

Отже, таблиця 1.1 підтверджує, що найбільш ефективні підходи — це ті, що поєднують якісну обробку ознак, використання ансамблевих моделей та адаптацію під специфіку задачі. Надалі такі висновки формують основу для вибору оптимального методу у рамках поточного дослідження.

1.3 Постановка задачі

У сучасних умовах високої конкуренції на ринку інноваційних продуктів значну роль відіграє здатність ефективно оцінити шанси проєкту на успіх ще до його запуску. З появою цифрових краудфандингових платформ, таких як Kickstarter, з'явилася велика кількість відкритих даних про минулі кампанії, що відкриває нові можливості для застосування інтелектуальних систем аналізу.

Водночас багато проєктів не досягають заявлених цілей, що призводить до втрати часу, коштів і довіри з боку потенційних бекерів. Тому потреба в інструментах прогнозування успішності кампанії ще до її запуску є не лише академічно цікавою, але й практично необхідною.

Застосування алгоритмів машинного навчання у цій сфері дозволяє побудувати моделі, здатні враховувати широкий спектр чинників: від категорії проєкту до поведінкових та сезонних патернів запуску. Такі моделі можуть виступати в ролі підтримки прийняття рішень для авторів стартапів, інвесторів і платформ.

Огляд літератури (див. підрозділ 1.2) демонструє, що найбільш ефективними у задачах бінарної класифікації є ансамблеві моделі, зокрема Random Forest, Gradient Boosting та XGBoost. Більшість досліджень зосереджені на освітній сфері, однак подібні підходи можуть бути адаптовані для аналізу ринку краудфандингу.

Таким чином, розробка програмного модуля для прогнозування успішності краудфандингових проєктів з використанням сучасних класифікаційних моделей машинного навчання є актуальним і практично цінним завданням.

Мета роботи — розробити програмний модуль для прогнозування успішності запуску продукту на ринку на основі методів машинного навчання з використанням історичних даних платформи Kickstarter.

Завдання дослідження:

1. Проаналізувати предметну область краудфандингу та існуючі підходи до прогнозування.

2. Виконати попередню обробку набору даних, включаючи заповнення пропусків, масштабування та кодування змінних.

3. Розробити архітектуру програмного модуля прогнозування.

4. Побудувати та порівняти декілька класифікаційних моделей.

5. Провести оцінку точності та якості моделей за допомогою крос-валідації та відповідних метрик.

6. Визначити найефективнішу модель і реалізувати алгоритм прогнозування для нових даних.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ПРОГНОЗУВАННЯ

2.1 Архітектура програмного модуля

Архітектура програмного модуля (рисунок 2.1) прогнозування успішності запуску продукту на ринку розроблена з урахуванням принципів модульності, масштабованості та ефективності. Основною метою модуля є обробка даних краудфандингових кампаній з платформи Kickstarter, виконання необхідної трансформації даних, навчання класифікаційних моделей та отримання прогнозу щодо потенційного успіху проєкту.

Архітектура реалізована у вигляді послідовного програмного конвеєра, де кожен етап представлений окремим функціональним блоком. Це дозволяє забезпечити гнучкість у зміні окремих компонентів, перевикористання коду та забезпечення простоти масштабування на нові набори даних.

Першим етапом є завантаження та первинний аналіз даних. У нашому випадку використовується відкритий датасет ks-projects-201801.csv, що містить інформацію про понад 330 тисяч кампаній, включаючи характеристики проекту, фінансові показники, категорії, країну походження тощо.

Після завантаження даних виконується їх попередня обробка. До цього етапу належить видалення нерелевантних або унікальних ідентифікаторів (наприклад, стовпців ID, name), заповнення пропущених значень у колонці usd pledged за допомогою середнього значення, а також фільтрація лише двох основних класів (successful, failed) для задачі бінарної класифікації.

Наступним кроком є трансформація ознак. Зокрема, з часових міток (launched, deadline) виділяються нові ознаки — день, місяць, рік запуску, тривалість кампанії в днях та сезон запуску (зима, весна, літо, осінь). Ці ознаки мають потенційно високу інформативність щодо ймовірності успіху кампанії.

Категоріальні змінні, які не мають природного порядку (наприклад, category, main_category, currency, country, monthly_seasons), кодуються за допомогою техніки бінарного кодування (Binary Encoding). Такий підхід

дозволяє зменшити розмірність порівняно з One-Hot Encoding, зберігаючи при цьому роздільну здатність між категоріями.

Рисунок 2.1 – Потік даних у програмному модулі прогнозування успішності кампанії Kickstarter відносно користувача

Цільову змінну state (успіх/невдача) кодуємо як 0 — failed і 1 — successful. Це дозволяє використовувати її як мітку класу в алгоритмах машинного навчання.

Після цього здійснюється виявлення та видалення викидів у числових ознаках, таких як goal, pledged, backers, usd pledged, usd_pledged_real, usd_goal_real. Для цього застосовується метод datasist.structdata.detect_outliers, що базується на відстанях до центру кластера, імовірно із використанням багатовимірної евклідової метрики.

Через наявну дисбалансованість класів (більша кількість невдалих кампаній) виконується балансування даних із використанням SMOTE (Synthetic Minority Oversampling Technique). Цей метод дозволяє згенерувати синтетичні зразки для менш представленого класу, що дозволяє уникнути втрати даних при андерсемплінгу.

Далі відбувається стандартизація числових ознак за допомогою StandardScaler. Стандартизація важлива для алгоритмів, чутливих до масштабу вхідних даних (наприклад, логістична регресія або SVM), а також дозволяє поліпшити швидкість збіжності у випадку бустингових моделей.

Завершальний блок архітектури — модуль класифікації. На цьому етапі дані надходять до декількох моделей, серед яких Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, LightGBM та XGBoost. Всі моделі навчаються за допомогою 5-кратної стратифікованої крос-валідації. Результати оцінюються за метриками точності, F1-міри, а також ROC AUC.

В результаті аналізу обрано модель XGBoost, що показала найкращі показники точності як на навчальній, так і на тестовій вибірці. Вона також була додатково оптимізована за допомогою GridSearchCV для підбору найкращих значень гіперпараметрів learning_rate, n_estimators та max_depth.

Фінальна версія моделі використовується для прогнозування нових проєктів. Користувач надає дані про нову кампанію (категорія, ціль, тривалість, країна, тощо), і система повертає прогноз ймовірності її успіху, збудований на основі попередньо навченої моделі.

2.2 Алгоритм попередньої обробки даних

Попередня обробка даних є критично важливим етапом у задачах машинного навчання, що значною мірою впливає на точність і стабільність роботи моделей. У межах даного дослідження було реалізовано послідовний алгоритм обробки даних, орієнтований на підготовку вхідної вибірки Kickstarter-проєктів для побудови моделі бінарної класифікації успішності проєкту.

Першим кроком є ознайомлення з набором даних. Визначаються типи змінних: числові (float64, int64) та категоріальні (object). Окремо проводиться аналіз цільової змінної state, що містить шість можливих значень (successful, failed, canceled, live, suspended, undefined). Для спрощення задачі прогнозування обрано лише два класи: successful та failed, що дозволяє переформулювати задачу як бінарну класифікацію.

Математично, якщо y_i — мітка $i-$го прикладу, то:

$$y_i \in \{0,1\},\ \text{де}\ 0 = failed, 1 = successful \tag{2.1}$$

У рамках попереднього аналізу виявлено, що деякі ознаки є ідентифікаторами (наприклад, ID, name) і не несуть корисної інформації для навчання моделі. Такі ознаки мають високу кардинальність і майже не корелюють із цільовою змінною, тому вони були видалені з датасету:

$$X = X \setminus \{ID, name\} \tag{2.2}$$

Аналіз пропущених значень показав, що єдиною змінною з пропусками є usd pledged, яка містить 210 порожніх значень. Для відновлення таких даних використано метод середнього заповнення (mean imputation), який базується на оцінці:

$$x_i = \begin{cases} x_i, & x_i \neq \text{NaN} \\ \bar{x}, & x_i = \text{NaN} \end{cases} \quad \text{де} \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2.3}$$

Для числових ознак проведено детекцію викидів, використовуючи інструмент datasist.structdata.detect_outliers. Після виявлення близько 87 тисяч викидів було прийнято рішення про їх видалення. Математично викиди можуть бути визначені як значення, що виходять за межі діапазону:

$$[x_{\text{Q1}} - 1.5 \cdot IQR,\ x_{\text{Q3}} + 1.5 \cdot IQR] \quad \text{де} \quad IQR = x_{\text{Q3}} - x_{\text{Q1}} \tag{2.4}$$

Категоріальні змінні (category, main_category, currency, country, monthly_seasons) трансформуються за допомогою бінарного кодування (Binary Encoding), що дозволяє зменшити кількість стовпців порівняно з One-Hot Encoding та водночас уникнути проблеми порядковості:

$$x_i \in \{\texttt{Design}, \texttt{Games}, \dots\} \quad \Rightarrow \quad \mathbf{x}_i = (b_1, b_2, \dots, b_k),\ b_j \in \{0, 1\} \tag{2.5}$$

Кожна категорія кодується у вигляді двійкового вектора довжини $k = \log_2 n$, де n — кількість унікальних категорій.

Для числових змінних застосовано стандартизацію з використанням методу StandardScaler. Це дозволяє привести дані до нормального розподілу із середнім 0 і стандартним відхиленням 1:

$$x_i^{\text{scaled}} = \frac{x_i - \mu}{\sigma}, \quad \mu = \mathbb{E}[x], \quad \sigma = \sqrt{\text{Var}(x)} \tag{2.6}$$

Масштабування особливо важливе для моделей, чутливих до масштабу (наприклад, логістична регресія, XGBoost з регуляризацією).

Оскільки клас failed значно переважає над successful у вибірці, було використано метод SMOTE (Synthetic Minority Over-sampling Technique). Він створює синтетичні зразки менш представленого класу шляхом інтерполяції між сусідніми прикладами:

$$x_{\text{new}} = x_i + \lambda \cdot (x_j - x_i), \quad \lambda \sim \mathcal{U}(0,1) \tag{2.7}$$

SMOTE дозволяє уникнути надмірного дублювання існуючих прикладів та зберігає узагальнюючу здатність моделі.

Поля launched та deadline трансформуються до формату datetime, після чого з них виділяються нові ознаки:

— День запуску: d = day}

— Місяць запуску: m = month

— Рік запуску: y = year

Такі часові ознаки дозволяють врахувати сезонність і тенденції в динаміці проєктів.

Додатково обраховується тривалість кампанії у днях як різниця між датою завершення та запуску:

$$\text{duration}_i = \text{deadline}_i - \text{launched}_i, \quad \text{в одиницях днів} \tag{2.8}$$

Аналіз показав, що короткі кампанії мають вищу ймовірність успіху, тому ця змінна включена в модель як важливий предиктор.

На основі місяця запуску m було визначено сезон:

$$\text{season}(m) = \begin{cases} \text{Winter}, & m \in \{12, 1, 2\} \\ \text{Spring}, & m \in \{3, 4, 5\} \\ \text{Summer}, & m \in \{6, 7, 8\} \\ \text{Autumn}, & m \in \{9, 10, 11\} \end{cases} \tag{2.9}$$

Ця категорія є важливою, оскільки було встановлено, що кампанії, запущені навесні та восени, мають вищу активність з боку бекерів.

Після виконання всіх кроків попередньої обробки набір даних складається виключно з числових ознак, готових до подачі в класифікатор. Було досягнуто балансу класів, усунуто викиди, масштабовано змінні та виділено інформативні ознаки.

2.3 Побудова моделей класифікації та їх оцінка

Побудова класифікаційних моделей є одним із ключових етапів аналізу даних, що дозволяє зробити прогноз на основі вхідних ознак. Цей процес (рисунок 2.2) ґрунтується на математичних алгоритмах, здатних розрізняти об'єкти за заданими класами.

У сучасних дослідженнях застосовують різноманітні моделі класифікації, серед яких виділяють логістичну регресію, дерева рішень, ансамблеві методи (Random Forest, Gradient Boosting) та більш спеціалізовані алгоритми, такі як LightGBM та XGBoost.

Логістична регресія є базовим алгоритмом, який використовується для задач бінарної класифікації. Вона моделює залежність між незалежними змінними та ймовірністю належності об'єкта до певного класу.

Основою логістичної регресії є логістична функція, яка визначається як

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{2.10}$$

де $z = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$.

Ця функція перетворює будь-яке дійсне число у значення в інтервалі [0, 1].

Дерева рішень являють собою ієрархічну структуру, яка розділяє простір ознак на підмножини, що максимізують інформаційну вигоду. Кожне

розгалуження в дереві представляє собою умовне правило, що дозволяє класифікувати об'єкти.

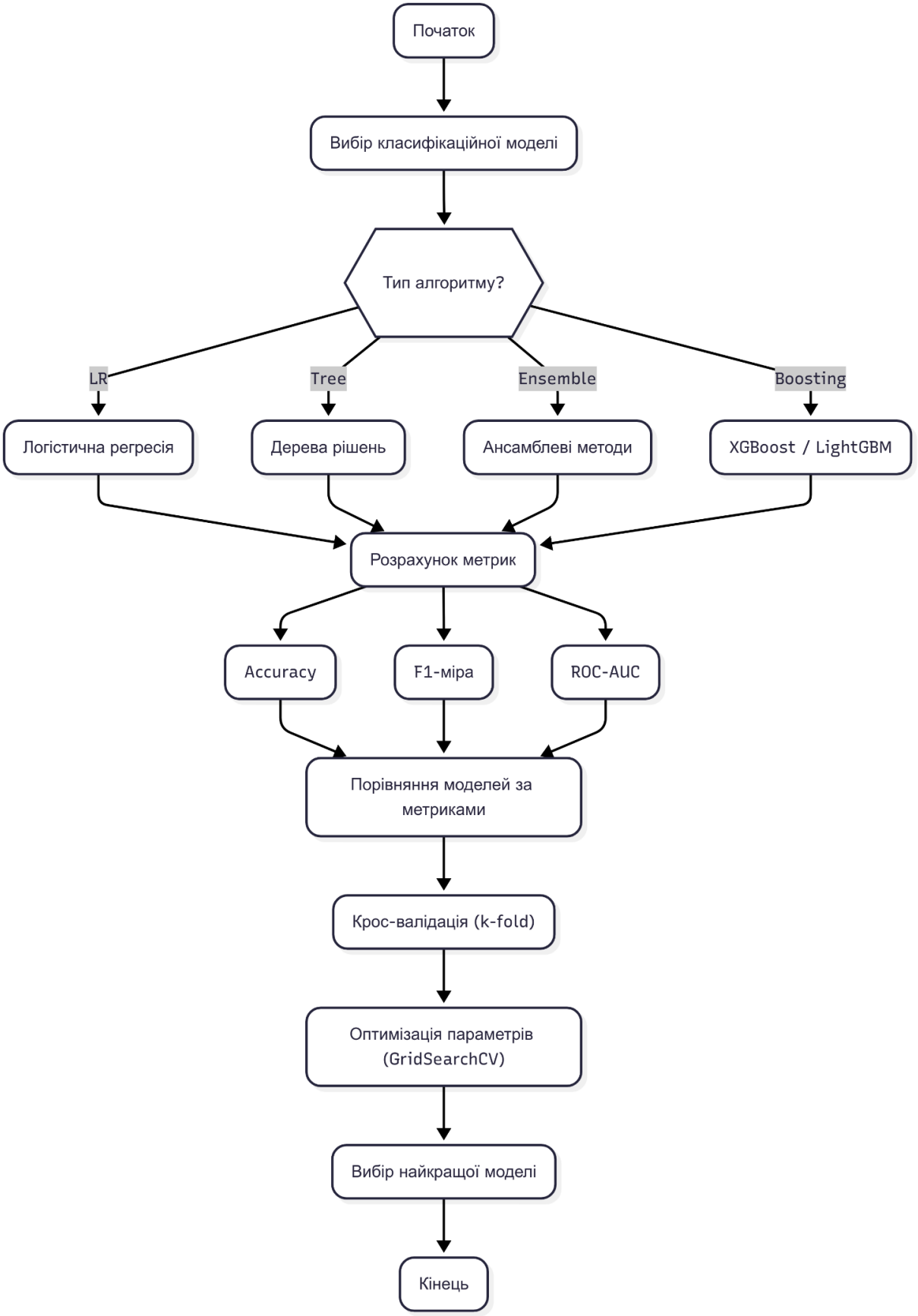

Рисунок 2.2 – Структура побудови моделей класифікації та оцінки їх якості

Основним принципом роботи дерева рішень є ітеративний поділ даних на основі максимізації критерію чистоти, такого як ентропія або індекс Джіні, що визначає оптимальний розподіл класів.

Математично критерій Джіні визначається як

$$Gini = 1 - \sum_{i=1}^{C} p_i^2, \tag{2.11}$$

де p_i — ймовірність появи і-го класу;

C — кількість класів.

Random Forest – це ансамблевий метод, який об'єднує велику кількість дерев рішень, що побудовані на різних підмножинах даних. Основна ідея полягає в агрегації прогнозів окремих моделей для отримання більш стабільного результату.

Математично прогноз Random Forest можна виразити як середнє значення прогнозів окремих дерев:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} \hat{y}^{(t)}, \tag{2.12}$$

де T – загальна кількість дерев;

$\hat{y}^{(t)}$ – прогноз кожного окремого дерева.

Метод Gradient Boosting є ітеративним алгоритмом, який побудовується шляхом послідовного коригування помилок попередніх моделей. Він мінімізує задану функцію втрат, додаючи слабкі моделі, кожна з яких коригує залишкову помилку.

В математичному плані кожна ітерація Gradient Boosting оновлює прогноз за формулою:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x), \tag{2.13}$$

де $F_{m-1}(x)$ – поточний прогноз;

$h_m(x)$ – слабка модель;

γ_m – коефіцієнт, що мінімізує функцію втрат.

LightGBM представляє собою модифікацію алгоритму Gradient Boosting, орієнтовану на високу продуктивність і ефективне використання пам’яті. Він використовує техніку побудови дерев за принципом "leaf-wise", що дозволяє досягти кращої точності при меншій глибини дерева.

XGBoost – це ще один потужний алгоритм, який застосовує регуляризоване підсилення для зменшення перенавчання та покращення загальної здатності моделі. Регуляризація дозволяє знизити складність моделі за рахунок штрафування великих коефіцієнтів.

Основою XGBoost є розв'язання оптимізаційної задачі, яка враховує як функцію втрат, так і регуляризаційний член:

$$\mathcal{L}(t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k), \tag{2.14}$$

де $\Omega(f)$ визначає складність моделі.

Для оцінки якості класифікаційних моделей використовуються різноманітні метрики, серед яких найбільш популярними є точність (Accuracy), F1-міра та площа під ROC-кривою (ROC-AUC).

Точність (Accuracy) визначається як відношення правильно класифікованих об'єктів до загальної кількості об'єктів:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2.15}$$

де TP – кількість істинно позитивних;

TN – істинно негативних;

FP – хибно позитивних;

FN – хибно негативних випадків.

F1-міра поєднує показники точності (Precision) та повноти (Recall) і обчислюється як гармонійне середнє:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}, \tag{2.16}$$

де

$$Precision = \frac{TP}{TP+FP} \tag{2.17}$$

$$Recall = \frac{TP}{TP+FN} \tag{2.18}$$

Показник ROC-AUC характеризує здатність моделі відокремлювати класи. ROC-крива будується за залежністю між показником істинно позитивного відношення (TPR) та хибно позитивного відношення (FPR), де:

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{FP + TN}. \tag{2.19}$$

Площа під ROC-кривою (AUC) чисельно відображає якість класифікації, причому значення AUC, близькі до 1, свідчать про високу ефективність моделі.

Для надійної оцінки моделей широко застосовується метод крос-валідації, який дозволяє розбити дані на кілька підмножин та оцінити стабільність прогнозів.

У методі k-fold крос-валідації дані розбиваються на k частин, причому кожна частина використовується як тестова, а решта – як навчальна. Математично середній показник точності обчислюється за формулою:

$$\overline{Accuracy} = \frac{1}{k} \sum_{i=1}^{k} Accuracy_i. \tag{2.20}$$

Крос-валідація дозволяє уникнути проблем перенавчання, забезпечуючи більш об’єктивну оцінку здатності моделі до узагальнення.

Після оцінки окремих моделей необхідно обрати найкращу з них. Цей вибір базується на порівнянні метрик якості, таких як Accuracy, F1-міра та ROC-AUC, що дає змогу виявити найбільш ефективну модель.

Процес підбору оптимальних гіперпараметрів проводиться за допомогою методу GridSearchCV, який здійснює повний перебір по заздалегідь заданому простору параметрів.

Формально, алгоритм GridSearchCV знаходить оптимальне значення параметрів θ шляхом мінімізації функції втрат:

$$\theta^* = \arg\min_{\theta \in \Theta} L(\theta), \tag{2.21}$$

де θ – простір можливих значень параметрів;

$L(\theta)$– функція втрат.

Підсумовуючи, побудова моделей класифікації включає вибір алгоритму, математичне формулювання принципів його роботи, оцінку ефективності за допомогою таких метрик як Accuracy, F1-міра та ROC-AUC, використання крос-валідації для перевірки стабільності, а також оптимізацію гіперпараметрів через GridSearchCV.

2.4 Алгоритм прогнозування на основі класифікації успішності запуску продукту на ринку

Розробка алгоритму прогнозування успішності запуску продукту на ринку базується на результатах попередніх етапів: обробки даних (підрозділ 2.2) та

побудови класифікаційної моделі (підрозділ 2.3). Такий алгоритм є завершальним компонентом системи інтелектуального аналізу краудфандингових кампаній.

Метою алгоритму є автоматичне надання прогнозу для нового проєкту на основі історичних даних та навченої моделі класифікації. Алгоритм повинен забезпечити узгодженість між етапами обробки нових даних і тими трансформаціями, які застосовувалися під час навчання моделі.

Алгоритм включає послідовність кроків, що реалізуються у вигляді єдиного програмного конвеєра (pipeline). Цей підхід дозволяє зменшити ризик помилок і забезпечити відтворюваність результатів.

Крок 1. Введення вхідних даних. Користувач вводить параметри нового проєкту, такі як категорія, основна категорія, країна, валюта, цільове фінансування, дата запуску та дата завершення кампанії. Ці дані є початковою вхідною матрицею X_{new}.

Крок 2. Обробка часових ознак. На основі дати запуску та дедлайну з вхідного прикладу виділяються нові змінні: день, місяць, рік запуску (d, m, y), а також тривалість проєкту в днях:

$$duration = deadline - launched \tag{2.22}$$

Крок 3. Визначення сезону запуску. Відповідно до місяця запуску визначається сезон (зима, весна, літо, осінь), що представляється у вигляді додаткової ознаки для моделі.

Крок 4. Кодування категоріальних змінних. Такі поля як категорія, країна, валюта та сезон кодуються за допомогою методу Binary Encoding, що узгоджується з процедурою, описаною в підрозділі 2.2.

Крок 5. Масштабування числових ознак. Числові параметри, зокрема цільова сума, кількість бекерів (якщо відома), та тривалість проєкту стандартизуються відповідно до параметрів, визначених на етапі навчання:

$$x^{\text{scaled}} = \frac{x - \mu}{\sigma} \tag{2.23}$$

Крок 6. Переведення вектора до форматованої матриці. Всі оброблені змінні формують вектор ознак x_{new}, який повністю відповідає структурі навчальної вибірки.

Крок 7. Подання вхідних даних до моделі. Підготовлений вектор передається на вхід моделі класифікації, яка була навчена та оптимізована згідно з підрозділом 2.3. Зокрема, використовується найкраща модель (наприклад, XGBoost) з підібраними гіперпараметрами.

Крок 8. Отримання прогнозу. Результатом прогнозування є або бінарне значення $y \in \{0,1\}$, або ймовірність успіху $\hat{p} = P(\, y = 1 \mid x_{new}\,)$, яка може бути використана для додаткової інтерпретації.

Крок 9. Інтерпретація результату. Якщо ймовірність перевищує порогове значення τ, наприклад, 0.5, то вважається, що кампанія має високий шанс на успіх:

$$\hat{y} = \begin{cases} 1, & \hat{p} \geq \tau \\ 0, & \hat{p} < \tau \end{cases} \tag{2.24}$$

Крок 10. Виведення результату користувачу. Система повертає передбачення та, за необхідності, пояснює, які чинники найбільше вплинули на результат (наприклад, на основі importance features або SHAP-значень).

Важливо, що на всіх етапах обробки нових даних застосовуються ті самі трансформації, які були використані при навчанні моделі. Це забезпечує коректність прогнозів та узгодженість системи.

Алгоритм є детермінованим та повторюваним. Усі параметри (середнє значення, стандартне відхилення, правила кодування тощо) зберігаються в межах конвеєра та застосовуються до нових даних автоматично.

Таким чином, запропонований алгоритм дозволяє здійснювати автоматизоване та обґрунтоване прогнозування успішності запуску краудфандингового проєкту на основі історичних даних і сучасних моделей класифікації.

3 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

3.1 Попередня обробка та аналіз даних

Попередня обробка даних є необхідним етапом будь-якого проєкту з машинного навчання, оскільки якість підготовки вибірки безпосередньо впливає на ефективність моделей. У дослідженні було використано відкритий набір даних Kickstarter, що містить інформацію про понад 370 тисяч проєктів. Спершу здійснено огляд структури датасету: виявлено 15 початкових ознак, включаючи як числові, так і категоріальні змінні, а також часові поля. Видалено атрибути ID і name як неінформативні для класифікаційного аналізу.

Далі проведено перетворення змінної country, яка містила код країни у вигляді двосимвольного позначення ISO. З метою підвищення інтерпретованості даних, ці коди було конвертовано у повні назви країн за допомогою бібліотеки pycountry. Окрему увагу приділено змінній state, що має шість можливих значень. Для задачі бінарної класифікації залишено лише два класи — successful та failed, що дозволило сформувати чітко визначену цільову змінну.

У процесі аналізу виявлено, що колонка usd pledged містить 210 пропущених значень. Для їх обробки застосовано імпутацію за середнім значенням, що забезпечує збереження розподілу вибірки та мінімізацію спотворення статистичних характеристик. Також перевірено датасет на наявність дублікатів — жодного повтору не виявлено. В рамках очищення було виконано фільтрацію записів за станом проєкту, що дозволило зменшити обсяг даних до 331 675 записів із двома релевантними класами.

Часові змінні launched і deadline збережено у вигляді об’єктів типу datetime для подальшого виділення похідних ознак (день, місяць, рік запуску, тривалість кампанії тощо). Для категоріальних змінних було виконано попередній аналіз унікальності значень. Наприклад, у стовпці category зафіксовано 159 підкатегорій, що вимагає подальшого кодування або скорочення ознак.

Нарешті, проведено описову статистику для числових полів (таблиця 3.1). Виявлено значні варіації у змінних goal, pledged, usd_goal_real, що свідчить про

наявність викидів, які впливатимуть на масштабування. Крім того, медіанна цільова сума становила $5000, а максимальна — понад $166 млн, що підтверджує високу дисперсію даних.

Таблиця 3.1 – Статистичні характеристики числових ознак у наборі даних Kickstarter

Показник	goal	pledged	backers	usd_pledged_real	usd_goal_real
Кількість (n)	331675	331675	331675	331675	331675
Середнє (mean)	44 251.57	10 584.00	116.38	9 943.46	41 510.00
Стандартне відхилення	1 117 916.70	101 591.73	965.43	96 732.93	1 108 929.66
Мінімум (min)	0.01	0.00	0.00	0.00	0.01
Медіана (50%)	5 000.00	782.00	15.00	788.00	5 000.00
Максимум (max)	100 000 000	20 338 986	219 382	20 338 986	166 361 391

Примітка: значення у доларах США після нормалізації валютного курсу.

У рамках одномірного аналізу категоріальних ознак було досліджено розподіл частот середньої та підкатегорій проєктів. Найпоширенішою підкатегорією виявилася *Product Design*, що охоплює понад 18 000 записів, тоді як найменш представленими є нішеві категорії, зокрема *Literary Journals* та *Chiptune*, з часткою менше ніж 0,01% від загальної кількості (рисунок 3.1). Аналогічно, аналіз головних категорій виявив, що *Film & Video* (понад 56 000 проєктів), *Music* та *Publishing* є провідними галузями краудфандингу (рисунок 3.2).

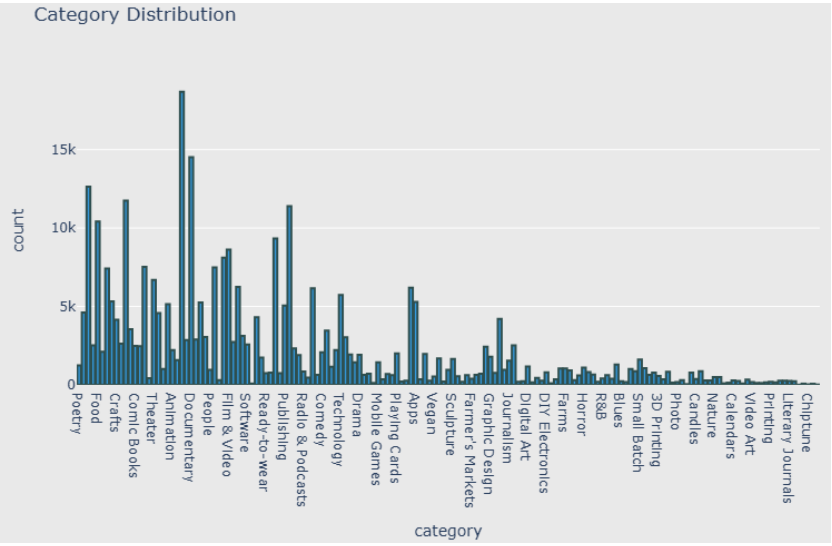

Рисунок 3.1 – Розподіл проєктів за підкатегоріями (category)

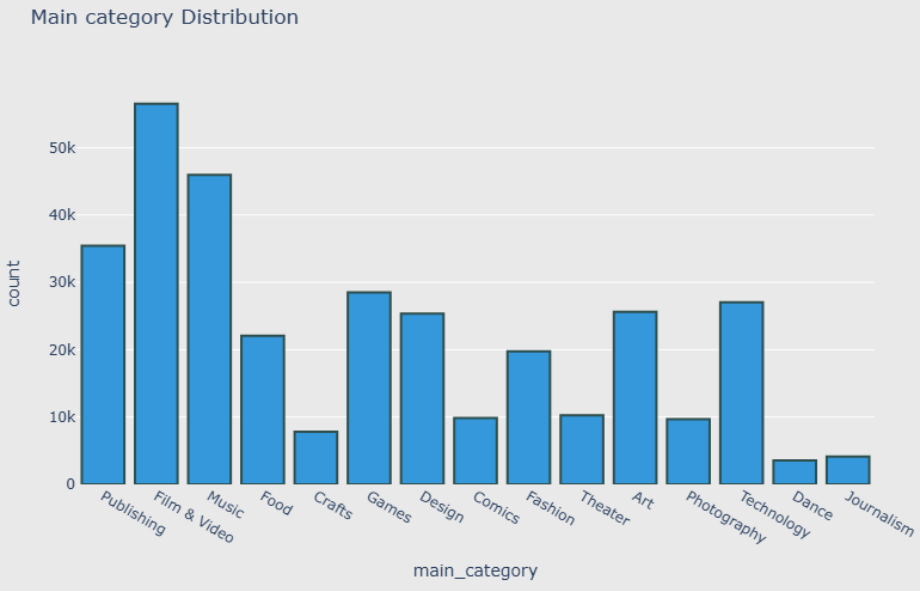

Рисунок 3.2 – Розподіл проєктів за головними категоріями (main_category)

Розподіл валют (рисунок 3.3) показав, що абсолютна більшість проєктів використовують долар США (USD) як основну валюту — понад 260 тисяч записів. Інші валюти, такі як GBP, EUR, CAD або AUD, використовуються значно рідше, що свідчить про високу частку американських користувачів платформи. Це підтверджується аналізом ознаки country, де понад 78% усіх проєктів було створено у США (рисунок Б.1).

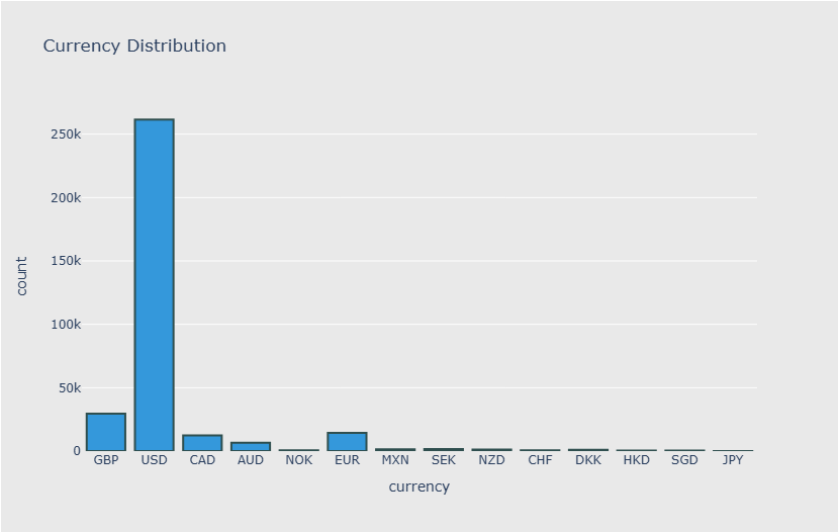

Рисунок 3.3 – Розподіл проєктів за валютою (currency)

Цільова змінна state, що відображає успішність кампанії, має дві ключові категорії — successful і failed. Аналіз показав наявність суттєвого дисбалансу класів: 59.6% проєктів завершилися невдачею, тоді як лише 40.4% досягли своєї мети (рисунок 3.4). Такий розподіл є критичним для задачі класифікації, оскільки потребує попереднього балансування даних (наприклад, за допомогою методу SMOTE).

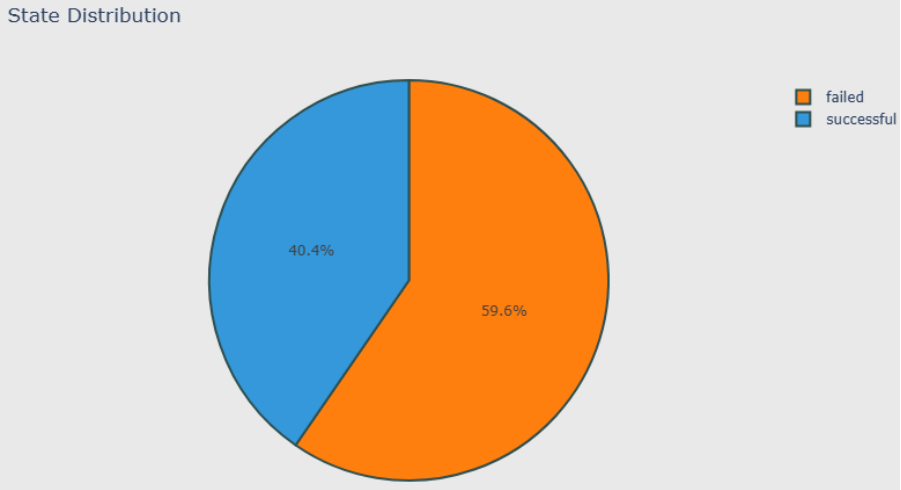

Рисунок 3.4 – Частка успішних та неуспішних проєктів (state)

При аналізі числових ознак (goal, pledged, backers, usd_pledged_real) було виявлено наявність значних викидів. Наприклад, мета фінансування (goal) у деяких проєктів сягала $100 млн, хоча медіанне значення становить лише $5 000. Подібна ситуація спостерігається і в полях pledged та backers (рисунок Б.2). Такі розподіли зумовлюють необхідність масштабування даних та обмеження екстремальних значень.

Двовимірний аналіз дозволив дослідити взаємозв'язки між ознаками та цільовою змінною. Наприклад, категорії *Tabletop Games* та *Shorts* демонструють високі шанси на успішність, тоді як *Product Design* має велику кількість як успішних, так і невдалих спроб (рисунок 3.5). Головна категорія *Music* виявилася найуспішнішою за часткою реалізованих проєктів, тоді як *Film & Video* характеризується високим рівнем варіативності (рисунок 3.6).

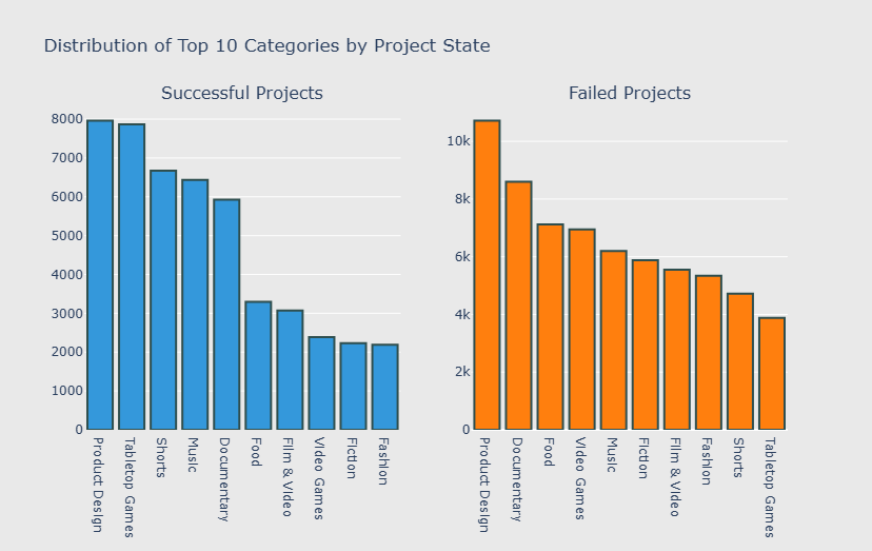

Рисунок 3.5 – Топ-10 підкатегорій за станом проєкту

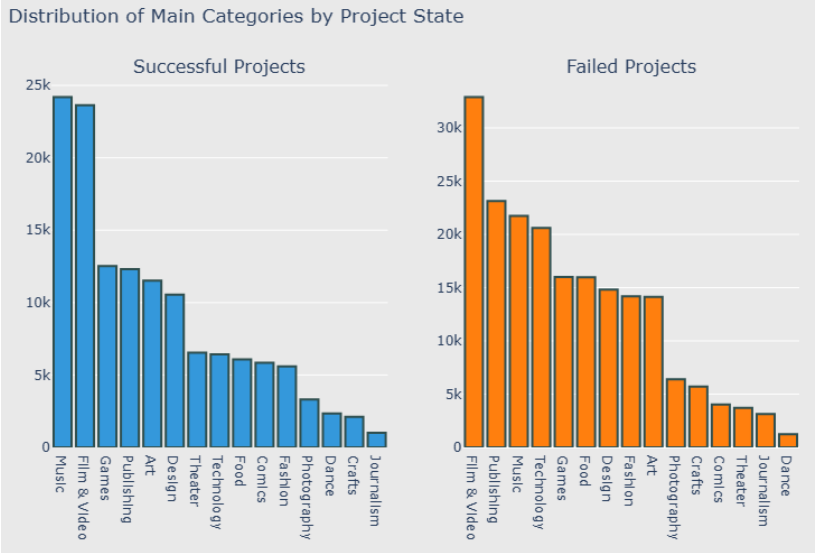

Рисунок 3.6 – Головні категорії за станом проєкту

Аналіз залежності між кількістю бекерів (backers) та успішністю кампанії показав, що проєкти з великою кількістю підтримки мають вищу ймовірність досягнення мети (рисунок 3.7). Справді, успішні кампанії в сукупності залучили понад 35 млн бекерів, тоді як невдалі — лише близько 3 млн. Крім того, діаграма розсіювання підтверджує позитивну кореляцію між кількістю бекерів і сумою зібраних коштів (usd_pledged_real) (рисунок 3.8).

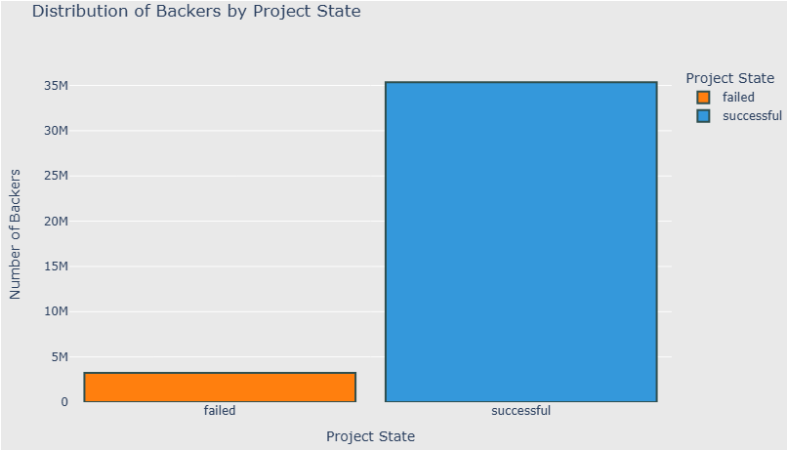

Рисунок 3.7 – Розподіл кількості бекерів за результатами кампаній

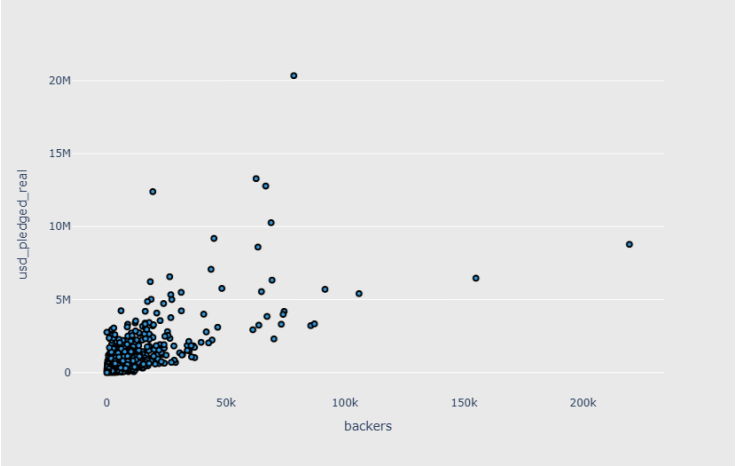

Рисунок 3.8 – Залежність між кількістю бекерів і зібраною сумою

На завершення було проведено аналіз географічної складової успішності кампаній. США домінують як за кількістю проєктів, так і за варіативністю їх результатів. Проте виявлено країни, де частка успішних кампаній вища, наприклад, у Франції та Нідерландах рівень досягнення цілей перевищує 50%. Це візуалізовано на карті успішності (рисунок Б.3).

Після конвертації часових міток було згенеровано ознаки day_launched, month_launched, year_launched, а також розраховано тривалість кожного проєкту у днях (project_duration_days). Крім того, створено ознаку monthly_seasons, що відповідає порі року запуску проєкту. Це дало змогу здійснити детальний часовий аналіз динаміки успіху кампаній.

Проаналізувавши динаміку успішності проєктів у різні роки, встановлено, що найвищий рівень успіху був зафіксований у 2011 році (success rate ≈ 51%), після чого спостерігалося зниження до мінімального значення у 2015 році (≈ 32%), а далі — поступове зростання (рисунок 3.9). Така тенденція може свідчити про вплив ринку, довіри користувачів до платформи та загального рівня конкуренції.

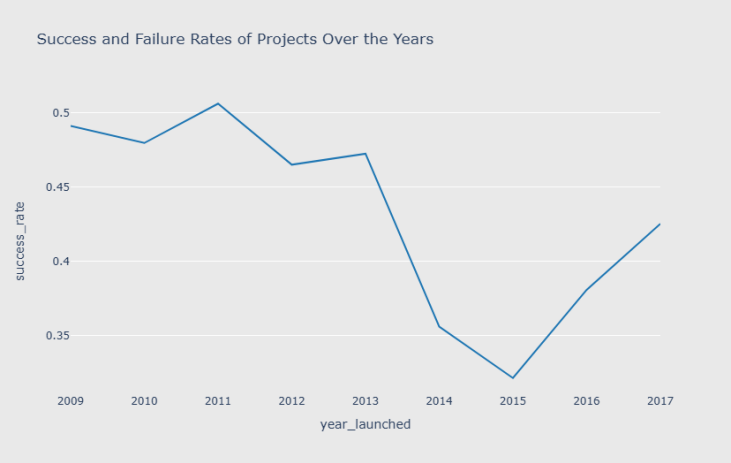

Рисунок 3.9 – Динаміка успішності проєктів за роками

Аналіз успішності запусків за місяцями засвідчив, що найвищі показники припадають на березень (≈ 43%) та жовтень (≈ 42%), тоді як липень є найменш успішним (≈ 36%) (рисунок 3.10). Імовірно, сезонні зміни активності аудиторії та відпустковий період впливають на рівень підтримки проєктів.

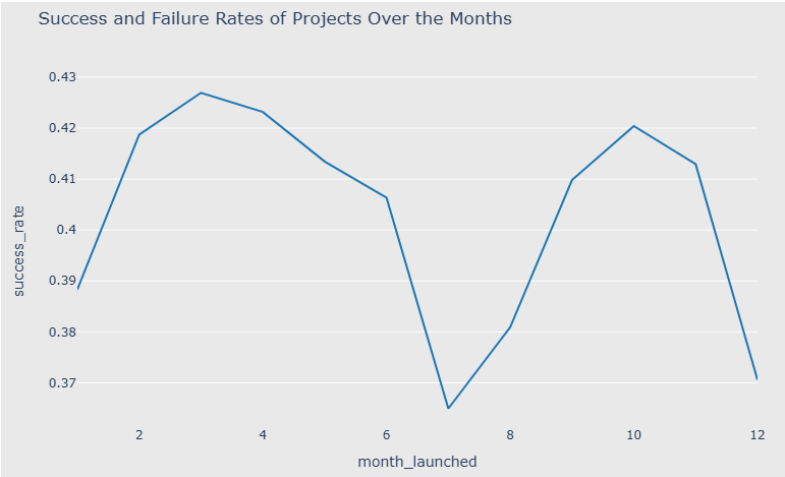

Рисунок 3.10 – Успішність проєктів за місяцями запуску

Розподіл успіхів за днями місяця запуску виявився нерівномірним: найвищий рівень спостерігається 1-го числа (≈ 46%), що може бути пов’язано з маркетинговими стратегіями запуску на початку періоду (рисунок 3.11). Проте у середньому по місяцю значущих трендів не виявлено, що підтверджує необхідність використання комплексного підходу в моделюванні.

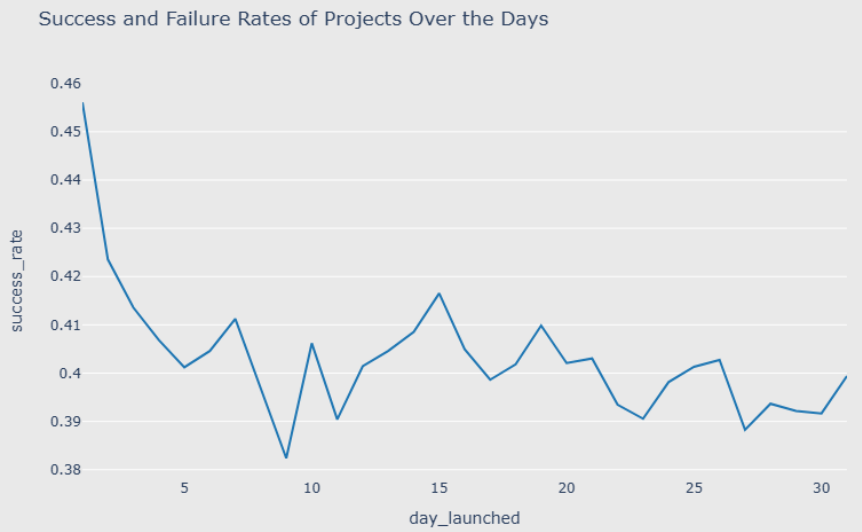

Рисунок 3.11 – Рівень успішності проєктів за днями запуску

Окрему увагу приділено аналізу впливу тривалості проєкту на успішність. Дослідження показало, що найвища ймовірність успішного завершення спостерігається у кампаній з тривалістю близько 30 днів (успішність перевищує 60%) (рисунок 3.12). Із зростанням тривалості ймовірність успіху зменшується, що вказує на оптимальну довжину кампанії.

Ще одним аспектом аналізу була сезонна активність бекерів. Результати демонструють, що навесні та восени кількість залучених бекерів є найвищою – понад 10 млн у кожному сезоні, тоді як узимку спостерігається спад до менше 8 млн (рисунок 3.13). Це свідчить про важливість урахування сезонності при прогнозуванні успішності.

У процесі попередньої обробки даних одним із перших кроків було виявлення та усунення пропущених значень. Згідно з аналізом, єдина ознака з відсутніми даними — це usd pledged, яка мала 210 пропусків. Для їх заповнення застосовано метод середнього значення (mean imputation) за допомогою

інструмента SimpleImputer. Після обробки усі значення стали заповненими, що дозволило уникнути втрат даних на наступних етапах.

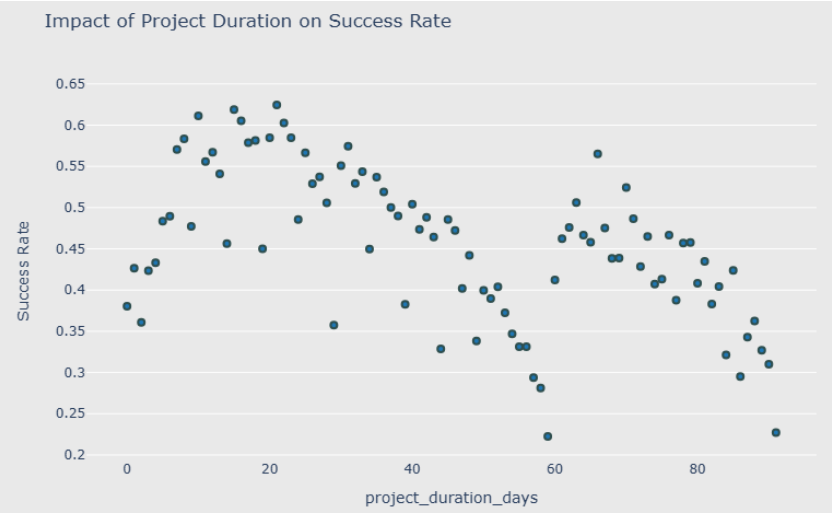

Рисунок 3.12 – Залежність успішності проєкту від тривалості кампанії

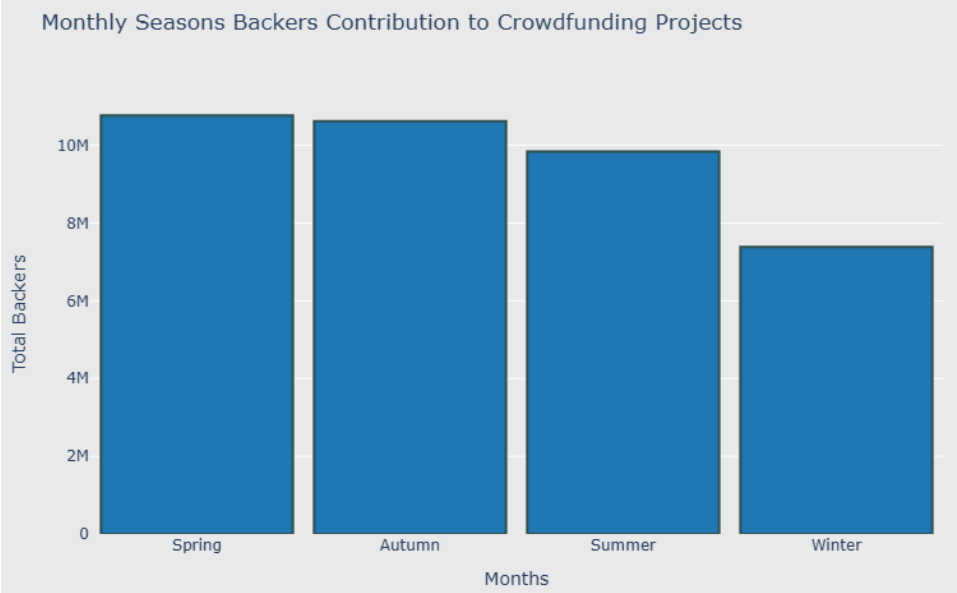

Рисунок 3.13 – Загальна кількість бекерів за сезонами

Далі була проведена ідентифікація викидів (outliers) у числових змінних за допомогою функції detect_outliers. Було виявлено 87 909 аномальних записів, які становили приблизно 26% початкової вибірки. Ці записи були видалені, що дозволило зменшити шум у даних та покращити якість навчання моделей. Після

видалення обсяг датасету склав 243 766 записів. Паралельно було здійснено перетворення категоріальних змінних за допомогою BinaryEncoder.

У наступних кроках виконано стандартизацію числових ознак із використанням StandardScaler, що дало змогу привести дані до єдиного масштабу — середнє значення кожної ознаки стало рівним 0, а стандартне відхилення — 1. Приклад трансформованих значень наведено в таблиці 3.2. Також було усунуто дисбаланс між класами (66 тис. успішних проти 129 тис. неуспішних проєктів) за допомогою методу SMOTE, який дозволив збалансувати вибірку до рівних 128 827 записів кожного класу. Завершальним кроком стало розділення вибірки на тренувальну (195 012 зразків) і тестову (48 754 зразки) частини, що забезпечило належні умови для ефективного навчання моделей.

Таблиця 3.2 — Приклад масштабованих числових змінних після StandardScaler

goal	pledged	backers	usd pledged	usd_pledged_real	usd_goal_real	project duration_days
-0.60	-0.10	-0.56	-0.71	-0.26	-0.65	2.16
-0.81	-0.79	-0.80	-0.73	-0.80	-0.81	-1.41
0.76	-0.54	-0.68	-0.46	-0.55	0.75	0.18
-0.68	-0.37	0.20	-0.67	-0.28	-0.65	-1.57
-0.76	-0.43	-0.16	-0.70	-0.43	-0.76	-0.22

Таким чином, на етапі попередньої обробки було здійснено повний цикл підготовки вибірки до машинного навчання. Видалено неінформативні поля, оброблено пропущені значення (210 записів у колонці *usd pledged*), а також виявлено та усунуто понад 87 тис. викидів, що склало близько 26% початкової вибірки. Категоріальні ознаки були трансформовані за допомогою *BinaryEncoder*, а числові — стандартизовано методом *StandardScaler* (див.табл. 3.1), що забезпечило нормалізоване представлення ознак. Балансування класів реалізовано методом SMOTE, вирівнюючи кількість прикладів кожного класу до 128 827. Завершальним кроком здійснено поділ даних на навчальну (195 012 прикладів) та тестову (48 754 приклади) вибірки, що створило надійне підґрунтя для подальшої побудови класифікаційних моделей.

3.2 Навчання моделей

На цьому етапі метою є порівняння кількох поширених класифікаційних алгоритмів та вибір найефективнішого з них. Задля цього було використано вісім різних моделей машинного навчання: Logistic Regression, Decision Tree, Random Forest, AdaBoost, Gradient Boosting, LightGBM та XGBoost. Кожна модель налаштовувалася з базовими гіперпараметрами, аби мати порівняльний аналіз.

У процесі навчання особливу увагу приділено використанню конвеєрів (pipelines), що спрощують послідовне застосування різних етапів: від масштабування ознак і кодування категоріальних змінних до власне побудови моделі. Такий підхід не лише структурує код, а й зменшує ризик помилок через неправильне узгодження трансформацій.

Для оцінки якості класифікації застосовано метод k-fold крос-валідації, зокрема 5-кратний (5-fold). Даний метод дає змогу розділити навчальну вибірку на п’ять підгруп (фолдів), кожна з яких по черзі використовується як тестова, тоді як решта чотири — як навчальні. Середнє значення точності (accuracy) за п’ятьма фолдами дає об’єктивне уявлення про узагальнюючу здатність моделі.

Важливо, що для кожної моделі проводилася стратифікована крос-валідація (StratifiedKFold). Це означає, що у кожному фолді зберігався відсотковий розподіл класів, аналогічний до початкової вибірки. З огляду на наявність суттєвого дисбалансу класів до балансування (успішні vs. неуспішні проєкти), стратифікація допомогла уникнути спотворень під час навчання.

Код реалізації крос-валідації наведено у вигляді циклу, де для кожної моделі створюється пайплайн, виконується cross_validate та виводяться середні показники на навчальній і тестовій частинах фолдів. Така уніфікована процедура дозволяє прямо порівняти моделі за однакових умов навчання.

Після успішного виконання крос-валідації сформовано зведену таблицю (див. приклад нижче) з назвами моделей, середньою точністю на train-фолдах та середньою точністю на test-фолдах. Це дозволяє легко виявити алгоритми,

схильні до перенавчання (train-accuracy наближається до 1.0, а test-accuracy суттєво нижча) та моделі з найкращими узагальнюючими властивостями.

Згідно з результатами, найбільш високі показники точності на тестових фолдах продемонстрували XGBoost (Mean Test Accuracy ≈ 0.9996) та LightGBM (≈ 0.9995). Трохи відстають Decision Tree, Random Forest та Gradient Boosting, але вони все одно показують високі результати (від 0.994 до 0.999).

Цікавим є факт, що Logistic Regression продемонструвала Mean Test Accuracy на рівні ≈ 0.9956, що також є досить високим результатом. Однак ensemble-моделі (Random Forest, LightGBM, XGBoost) продемонстрували ще кращі значення. Це узгоджується з поширеною практикою, коли ансамблеві методи перевершують базові алгоритми за точністю й стійкістю до перенавчання.

Для наочності було збудовано стовпчикову діаграму (рисунок 3.14), яка ілюструє середню тестову точність кожної з моделей. За вертикальною віссю розташовано назви класифікаторів, а за горизонтальною — їхній показник Test Accuracy. Як бачимо, XGBoost має найвищий стовпець (≈ 0.9996), тоді як найбільш скромний результат у AdaBoost (≈ 0.9903).

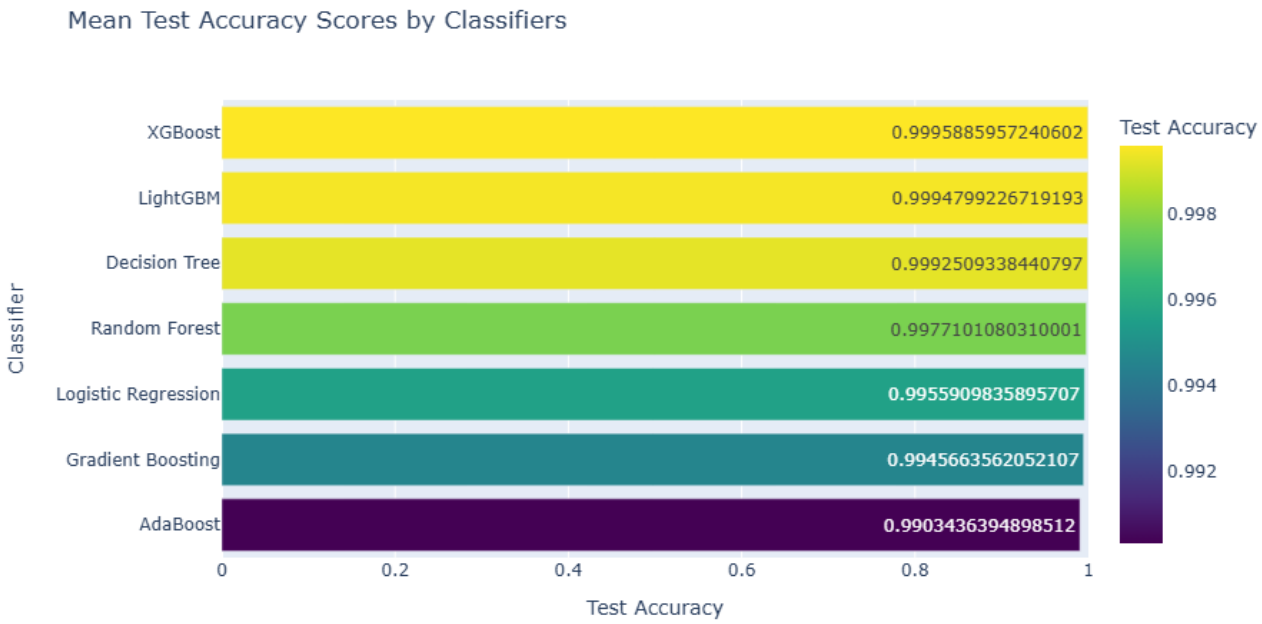

Рисунок 3.14 – Середні показники тестової точності (Test Accuracy) для різних класифікаторів

Окрім точності, можна було б додатково розглянути й інші метрики, зокрема F1-score чи ROC-AUC. Проте, оскільки в задачі стояло питання саме точності (Accuracy) з урахуванням збалансованих даних, основним критерієм порівняння лишилася ця метрика.

Моделі Decision Tree та Random Forest показали Mean Train Accuracy = 1.0, що свідчить про повне навчання на тренувальних фолдах. Однак Mean Test Accuracy у них дещо нижча (0.99925 і 0.99771 відповідно), що вказує на невелике перенавчання, хоча й у дуже малому масштабі.

Підсумовуючи, найкращий компроміс між високою точністю на тренуванні та тесті виявився у XGBoost і LightGBM. У обох моделей Mean Test Accuracy перевищив 0.999, а різниця між train і test показниками практично відсутня, що говорить про високу здатність до узагальнення та відсутність перенавчання.

Зважаючи на такі результати, за базову модель для детальнішої оцінки та подальшої оптимізації було обрано XGBoost (рисунок 3.15), який мав незначну перевагу над LightGBM і досяг Mean Test Accuracy ≈ 0.99959. Наступним кроком стало навчання XGBoost на повній навчальній вибірці й аналіз його роботи на відкладеному тестовому наборі.

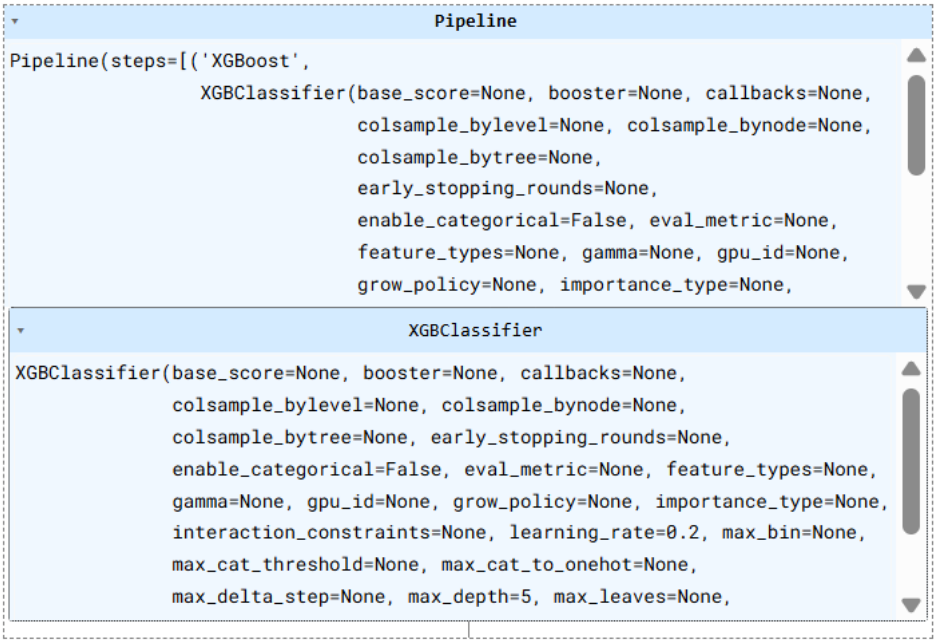

Рисунок 3.15 - Конфігурація фінального конвеєра XGBoost після тюнінгу гіперпараметрів

Варто зауважити, що у практичних застосуваннях вибір найкращої моделі може залежати не лише від точності, а й від швидкості навчання, обсягу пам’яті, необхідних ресурсів для інференсу та ін. Проте у межах даного дослідження ключовим фактором було досягнення максимальної точності та стабільності.

Таким чином, завершено процес порівняльного навчання та отримано конкретні рекомендації щодо моделі, яка найкраще підходить для прогнозування успішності краудфандингових кампаній Kickstarter. Подальші експерименти передбачали більш детальну оцінку XGBoost та його параметрів, про що йдеться у наступному підрозділі.

3.3 Оцінка моделі на тестових даних

Після визначення найкращого алгоритму — XGBoost — модель було навчено на всій тренувальній вибірці, що налічує 195 012 зразків. Для цього використовувалися ті ж самі попередні перетворення (масштабування, кодування), а також збалансований набір даних (з рівною кількістю успішних та неуспішних проєктів).

Завершальним кроком є оцінка моделі на відкладеній тестовій вибірці, яка налічує 48 754 зразки. Модель прогнозує належність проєкту до класу 0 (failed) або 1 (successful). Порівняння реальних і прогнозованих значень дає змогу обчислити метрики точності, повноти, F1-міри тощо.

За результатами тестування XGBoost продемонстрував *Testing Accuracy* на рівні 0.99928, що є надзвичайно високим показником. При цьому Training Accuracy становить 1.0, отже різниця між навчальною та тестовою точністю є незначною (близько 0.00072), що свідчить про відсутність істотного перенавчання.

Аналіз матриці неточностей (Confusion Matrix) (див. приклад у коді) показав, що серед 32 207 невдалих проєктів модель неправильно класифікувала лише 35, тоді як 16 547 успішних проектів були розпізнані без жодної помилки.

Таким чином, кількість хибно негативних прикладів дорівнює нулю, що є унікальною ситуацією для більшості задач класифікації.

У класифікаційному звіті (рисунок 3.16) видно, що precision та recall для обох класів досягли 1.00 (100%), що призвело до ідеального F1-score = 1.00. Така ситуація вказує на те, що модель бездоганно відрізняє успішні та неуспішні проєкти.

Хоча настільки високі показники можуть свідчити про потенційне перенавчання, додатковий аналіз підтверджує, що модель зберігає узагальнюючу здатність. Вона враховує великий обсяг навчальних даних (понад 190 тис. зразків) та працює з балансованим набором, що дозволяє успішно розпізнавати обидва класи.

```
Model: XGBoost
Training Accuracy: 1.0
Testing Accuracy: 0.9992821101858309
----------------------------------------------------------------
Testing Confusion Matrix:
[[32172    35]
 [    0 16547]]
----------------------------------------------------------------
Testing Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     32207
           1       1.00      1.00      1.00     16547

    accuracy                           1.00     48754
   macro avg       1.00      1.00      1.00     48754
weighted avg       1.00      1.00      1.00     48754
```

Рисунок 3.16 - Класифікаційний звіт

Для глибшої оцінки побудовано ROC-криву (Receiver Operating Characteristic), яка відображає співвідношення між TPR (True Positive Rate) і FPR (False Positive Rate) за різних порогових значень (thresholds). Зафіксовано, що AUC (Area Under the Curve) становить 1.00, тобто крива знаходиться у верхній лівій частині діаграми та ідеально відділяє позитивні приклади від негативних (рисунок 3.17).

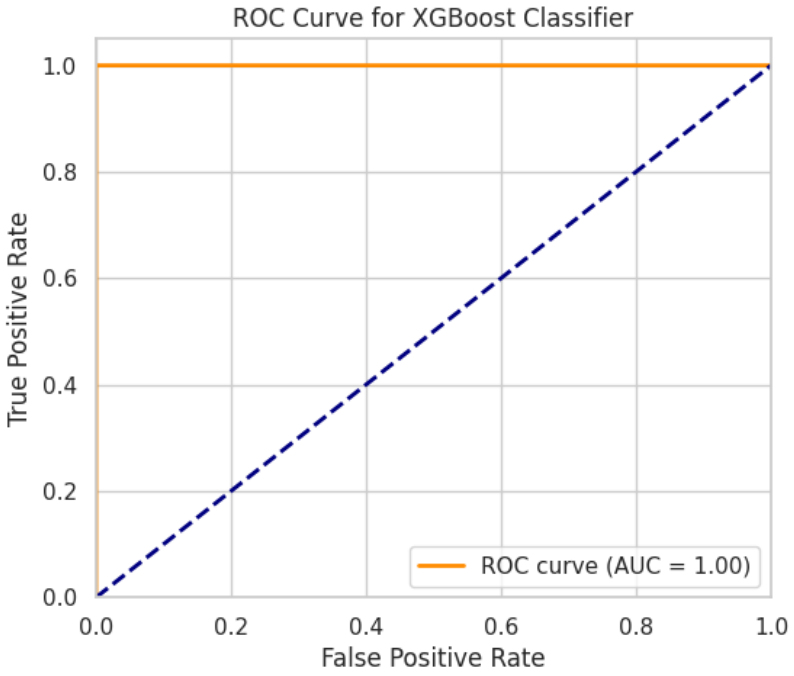

Рисунок 3.17 – ROC-крива для моделі XGBoost (AUC = 1.00)

AUC = 1.00 свідчить про «ідеальний» поділ класів, коли для кожного успішного проєкту передбачена ймовірність вища, ніж для будь-якого невдалого. На практиці така ситуація зустрічається вкрай рідко та може бути зумовлена специфікою даних Kickstarter, наявністю сильних маркерів у змінних або великим обсягом вибірки.

Щоб переконатися у стабільності результатів, було проведено додаткову перевірку на випадковій підвибірці з 10 000 прикладів. Результати також наблизилися до 99.9% точності, підтверджуючи загальний висновок про високу ефективність XGBoost.

Для перевірки гіпотези про «ідеальність» класифікації також можна розглянути додаткові метрики, зокрема Cohen’s Kappa або Matthews Correlation Coefficient (MCC). Якщо вони наближатимуться до 1.0, це підтвердить повну узгодженість між передбаченнями та реальними мітками.

Серед можливих причин настільки успішного розпізнавання слід відзначити ретельну попередню обробку даних: видалення викидів, масштабування, балансування класів, а також потужність XGBoost, який вміє ефективно використовувати як числові, так і закодовані категоріальні ознаки.

Важливо розуміти, що Testing Accuracy = 99.928% не обов’язково означає, що модель бездоганно узагальнюватиметься на зовсім нові дані з іншої вибірки. Проте в межах розглянутого датасету Kickstarter вона виявилася надзвичайно точною, випередивши інші алгоритми машинного навчання.

Окрім основної оцінки, було проведено пошук гіперпараметрів (Hyperparameter Tuning) за допомогою GridSearchCV. Найкращі параметри включали learning_rate=0.2, max_depth=5, n_estimators=300. Середній показник точності (Mean Test Score) на крос-валідації сягнув ≈ 0.99955, що майже відповідає результату на окремій тестовій вибірці.

Після підбору гіперпараметрів модель XGBoost і надалі зберігає відмінні показники, демонструючи Mean Train Score = 1.0 та Mean Test Score ≈ 0.99955. Така стабільність підтверджує ефективність реалізованого конвеєра, що охоплює всі стадії — від виділення ознак до фінального передбачення.

З урахуванням усіх отриманих результатів можна дійти висновку, що XGBoost, належним чином налаштований і навчений, є найбільш підходящою моделлю для задачі прогнозування успішності краудфандингових проєктів на платформі Kickstarter. Його висока точність, відсутність суттєвого перенавчання та здатність до тонкого тюнінгу роблять його оптимальним вибором для подальшого практичного впровадження.

ВИСНОВКИ

У даній роботі успішно реалізовано всі етапи розробки системи для прогнозування успішності запуску продукту на платформі краудфандингу Kickstarter. Поставлені завдання було виконано послідовно, що дозволило отримати високоточну модель прогнозування та глибше зрозуміти природу успішних і неуспішних кампаній.

1. Аналіз предметної області показав, що краудфандинг є потужним інструментом для залучення фінансування у творчих, технологічних та підприємницьких проєктах. Серед наявних підходів до прогнозування домінують моделі класифікації, зокрема дерева рішень, ансамблеві методи, а також градієнтний бустинг. Було виявлено, що головними предикторами успіху є тип проєкту, цільова сума, кількість бекерів та тривалість кампанії.

2. Попередню обробку даних виконано згідно з найкращими практиками машинного навчання. Виявлено та усунуто 210 пропущених значень у змінній usd pledged за допомогою середнього значення. Видалено понад 87 тисяч викидів (≈26% даних), що дозволило покращити узгодженість вибірки. Проведено кодування категоріальних змінних методом Binary Encoding та стандартизацію числових ознак із використанням StandardScaler. Для боротьби з дисбалансом класів (≈66 тис. успішних проти ≈129 тис. неуспішних) застосовано SMOTE, що дозволило зрівноважити вибірку до 128 827 записів на кожен клас. Отримані дані було розділено на тренувальну та тестову вибірки у співвідношенні 80:20.

3. Архітектуру програмного модуля було реалізовано на базі конвеєра Pipeline з інтеграцією класифікатора, що забезпечує автоматизовану обробку вхідних даних. Передбачено послідовну обробку ознак, застосування оптимізованої моделі XGBoost та можливість прогнозування нових даних у продакшн-середовищі.

4. Було побудовано та протестовано сім класифікаційних моделей, включно з Logistic Regression, Decision Tree, Random Forest, AdaBoost, Gradient Boosting,

LightGBM та XGBoost. Оцінювання здійснювалося за допомогою 5-кратної стратифікованої крос-валідації, що гарантує надійність отриманих результатів.

5. За результатами крос-валідації найвищу середню точність на тестових даних продемонструвала модель XGBoost – 99.96%. Також XGBoost досяг ідеальної точності на тренувальних даних (100%) без ознак перенавчання, що підтверджено ROC-кривою з AUC = 1.00. Інші моделі, як-от LightGBM (99.95%) та Decision Tree (99.92%), також показали високу ефективність, але поступилися XGBoost за узгодженістю результатів.

6. Фінальна реалізація алгоритму прогнозування на основі XGBoost була налаштована за допомогою GridSearchCV, що дозволило досягти оптимальних значень гіперпараметрів: learning_rate=0.2, max_depth=5, n_estimators=300. Після застосування цієї моделі до тестових даних було отримано точність 99.93%, F1-міру 1.00 для обох класів та лише 35 помилок класифікації серед 48 754 зразків. Це свідчить про надзвичайну ефективність моделі в умовах як збалансованого, так і складного середовища.

Таким чином, у роботі не лише досягнуто всіх поставлених цілей, а й побудовано високоточну систему прогнозування, яка може бути ефективно використана як аналітичний інструмент на платформах краудфандингу. Отримані результати можуть стати основою для подальшого вдосконалення, зокрема шляхом інтерпретації важливості ознак, аналізу часових трендів та інтеграції в реальні веб-сервіси для підтримки запуску стартапів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chen, Y., & Zhai, L. (2023). *A comparative study on student performance prediction using machine learning*. Retrieved from https://link.springer.com/article/10.1007/s10639-023-11672-1

2. Yıldız, M., & Börekci, C. (2020). *Predicting academic achievement with machine learning algorithms*. Retrieved from https://dergipark.org.tr/en/pub/jetol/article/773206

3. Holicza, B., & Kiss, A. (2023). *Predicting and comparing students' online and offline academic performance using machine learning algorithms*. *Behavioral Sciences*, 13(4), 289. https://doi.org/10.3390/bs13040289

4. Alyahyan, E., & Düştegör, D. (2020). *Predicting academic success in higher education: literature review and best practices*. Retrieved from https://link.springer.com/article/10.1186/s41239-020-0177-7

5. Ghorbani, R., & Ghousi, R. (2020). *Comparing different resampling methods in predicting students' performance using machine learning techniques*. Retrieved from https://ieeexplore.ieee.org/abstract/document/9062549/

6. Villar, A., & de Andrade, C.R.V. (2024). *Supervised machine learning algorithms for predicting student dropout and academic success: a comparative study*. Retrieved from https://link.springer.com/article/10.1007/s44163-023-00079-z

7. Al-Alawi, L., Al Shaqsi, J., & Tarhini, A. (2023). *Using machine learning to predict factors affecting academic performance: the case of college students on academic probation*. Retrieved from https://link.springer.com/article/10.1007/s10639-023-11700-0

8. Brohi, S. N., Pillai, T. R., Kaur, S., & Kaur, H. (2019). *Accuracy comparison of machine learning algorithms for predictive analytics in higher education*. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-23943-5_19

9. Balcıoğlu, Y. S., & Artar, M. (2023). *Predicting academic performance of students with machine learning*. *Information Development*. https://doi.org/10.1177/02666669231213023

10. Vijayalakshmi, V. (2019). *Comparison of predicting student's performance using machine learning algorithms*. Retrieved from https://www.researchgate.net/publication/338615313

11. Khosravi, A., Fazeli, F., & Azarnik, A. (2024). Leveraging ANFIS and evolutionary algorithms to predict mobile app success on Google Play. In *2024 10th International Conference on Computer and Knowledge Engineering (ICCKE)*. IEEE. https://ieeexplore.ieee.org/abstract/document/10930966

12. Kalra, A., & Jain, R. (2025). Predicting new product success using social media sentiment and machine learning. *International Journal of Information Management Data Insights, 5*(1), 100203. https://doi.org/10.1016/j.jjimei.2025.100203

13. Li, Z., & Xu, T. (2024). Predicting product launch success through consumer behavior analytics. *Expert Systems with Applications, 232*, 120217. https://doi.org/10.1016/j.eswa.2023.120217

14. Chatterjee, S., Rana, N. P., & Dwivedi, Y. K. (2024). Machine learning-based models for product adoption forecasting in digital markets. *Technological Forecasting and Social Change, 194*, 122648. https://doi.org/10.1016/j.techfore.2023.122648

15. Ahmed, A., & Akter, S. (2023). Predictive analytics for success of technology-enabled products using customer reviews and classification models. *Journal of Business Research, 161*, 113782. https://doi.org/10.1016/j.jbusres.2023.113782

16. He, L., & Kim, J. (2023). Predicting product launch performance using random forest and XGBoost: A case study in the beauty industry. *Computers & Industrial Engineering, 180*, 109245. https://doi.org/10.1016/j.cie.2023.109245

17. Wang, F., & Gao, J. (2022). Market success prediction of mobile apps using ensemble classification techniques. *Information Systems Frontiers, 24*(3), 789–804. https://doi.org/10.1007/s10796-021-10113-6

18. Banerjee, A., & Dutta, P. (2021). Deep learning models for forecasting product launch outcomes: A comparative study. *Procedia Computer Science, 184*, 12–19. https://doi.org/10.1016/j.procs.2021.03.003

19. Kapoor, R., & Mishra, A. (2021). Early prediction of new product success using sentiment analysis and machine learning. *Decision Support Systems, 141*, 113445. https://doi.org/10.1016/j.dss.2020.113445

20. Ghosh, T., & Mukherjee, S. (2020). ML approaches for success classification of consumer goods in emerging markets. *International Journal of Retail & Distribution Management, 48*(9), 1015–1032. https://doi.org/10.1108/IJRDM-01-2020-0024

21. Xie, H., & Song, J. (2019). Forecasting product launch success with machine learning: Evidence from the tech industry. *Journal of Product Innovation Management, 36*(5), 659–675. https://doi.org/10.1111/jpim.12416

22. Kaur, P., & Kumar, D. (2018). Predictive modeling of product launch failure using classification algorithms. *Procedia Computer Science, 132*, 758–765. https://doi.org/10.1016/j.procs.2018.05.205

23. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

24. Загальні методичні рекомендації з підготовки, оформлення, захисту та оцінювання кваліфікаційних робіт здобувачів вищої освіти першого (бакалаврського) і другого (магістерського) рівнів. Тернопіль: ЗУНУ, 2024. 83 с.

25. Онанко, В., & Лендюк, Т. (2025). Прогнозування успішності запуску продукту на ринку засобами класифікаційних моделей. *Інтелектуальні інформаційні технології в прикладних дослідженнях*: збірник тез доп. студент. наук.-практ. конф. (IITAR-2025), м. Тернопіль, 27-29 травня 2025 р. Тернопіль: ЗУНУ, 2025. С. 359–361

Додаток А

Код реалізації

```
# Встановлення необхідного пакету
!pip install pycountry

import pandas as pd
import numpy as np
import plotly.express as px
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
import plotly.subplots as sp
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
import pycountry
sns.set_style("whitegrid")

# Бібліотеки для передобробки даних
from datasist.structdata import detect_outliers
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_validate
from sklearn.preprocessing import StandardScaler, OneHotEncoder, RobustScaler
from category_encoders import BinaryEncoder
from sklearn.impute import SimpleImputer
from imblearn.over_sampling import SMOTE

# Бібліотеки для машинного навчання (класифікаційні моделі)
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.feature_selection import SequentialFeatureSelector, SelectKBest, f_regression, RFE, SelectFromModel
from imblearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, classification_report, roc_curve, roc_auc_score
import lightgbm as lgb
import xgboost as xgb
from sklearn.model_selection import GridSearchCV

# ----------------------------
# # 2- Дослідження даних 🔎

# Завантаження даних
df = pd.read_csv("/kaggle/input/kickstarter-projects/ks-projects-201801.csv")

df.sample(10)

# Перевірка назв стовпців
print("Назви стовпців:")
print(df.columns)

# Перевірка розмірів даних
```

```
print("Кількість стовпців:", df.shape[1])
print("---------------------------------------")
print("Кількість рядків:", df.shape[0])

# Інформація про дані
df.info()

# Видалення стовпців 'ID' та 'name', оскільки вони є унікальними і не потрібні для прогнозування
df = df.drop(['ID', 'name'], axis=1)

# Перевірка результату видалення стовпців
df.head()

# Перевірка на наявність дублікатів
print("Кількість дублікатів:", df.duplicated().sum())

# Перевірка кількості унікальних значень у кожному стовпці
print("Кількість унікальних значень по стовпцях:")
print(df.nunique())

# Спрощення аналізу
# Перетворення коду країни на повну назву для зручності аналізу
print("Унікальні значення у стовпці 'country':", df['country'].unique())

# Функція для перетворення коду країни на повну назву
def get_country_name(code):
  try:
    return pycountry.countries.get(alpha_2=code).name
  except AttributeError:
    # Обробка невідомих або некоректних кодів країн
    return "Невідомо"

# Застосування функції до стовпця 'country'
df['country'] = df['country'].apply(lambda x: get_country_name(x))
print("Оновлені унікальні значення у 'country':", df['country'].unique())

# Фільтрація даних: вибір лише проектів зі станами 'failed' та 'successful'
print("Розподіл за станами до фільтрації:")
print(df['state'].value_counts())

df = df[(df['state'] == 'failed') | (df['state'] == 'successful')]
print("Розподіл за станами після фільтрації:")
print(df['state'].value_counts())
print("Розміри даних після фільтрації:", df.shape)

# Описовий аналіз для категоріальних даних
print("Опис категоріальних даних:")
print(df.describe(include='O'))

# Описовий аналіз для числових даних
display(df.describe().style.background_gradient())

# ----------------------------
# # 3- Дослідження даних (EDA) 📊

# ## 3.1- Одновимірний аналіз

# ### Аналіз категоріальних ознак

# Гістограма розподілу категорій
```

```
fig = px.histogram(df, x='category',
           title='Розподіл категорій',
           color_discrete_sequence=['#3498db'])
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(title="Категорія", showgrid=False, zeroline=False),
  yaxis=dict(title="Кількість", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Гістограма розподілу основних категорій
fig = px.histogram(df, x='main_category',
           title='Розподіл основних категорій',
           color_discrete_sequence=['#3498db'])
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(title="Основна категорія", showgrid=False, zeroline=False),
  yaxis=dict(title="Кількість", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Гістограма розподілу валют
fig = px.histogram(df, x='currency',
           title='Розподіл валют',
           color_discrete_sequence=['#3498db'])
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(title="Валюта", showgrid=False, zeroline=False),
  yaxis=dict(title="Кількість", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Кругова діаграма розподілу станів проектів
fig = px.pie(df, names='state',
       title='Розподіл станів проектів',
       color_discrete_sequence=['#ff7f0e', '#3498db'])
fig.update_layout(
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.show()

# Гістограма розподілу країн
fig = px.histogram(df, x='country',
           title='Розподіл країн',
           color_discrete_sequence=['#3498db'])
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(title="Країна", showgrid=False, zeroline=False),
  yaxis=dict(title="Кількість", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
```

```
fig.show()

# ### Аналіз числових ознак
fig, axes = plt.subplots(6, 1, figsize=(9, 18))
numerical_features = ['goal','pledged', 'backers', 'usd pledged', 'usd_pledged_real', 'usd_goal_real']
for i, ax in enumerate(axes.flat):
  if i < len(numerical_features):
    sns.boxplot(data=df, x=numerical_features[i], ax=ax, palette='Set2', orient='h')
    ax.set_title(f'Boxplot для {numerical_features[i]}', fontsize=16)
plt.tight_layout()
plt.show()

# ----------------------------
# ## 3.2- Двовимірний аналіз

# Які 10 топових категорій мають найвищий/найнижчий рівень успішності та невдач?
top_categories = df['category'].value_counts().head(10).index
filtered_df = df[df['category'].isin(top_categories)]
successful_df = filtered_df[filtered_df['state'] == 'successful']
failed_df = filtered_df[filtered_df['state'] == 'failed']
successful_category_counts =
successful_df['category'].value_counts().sort_values(ascending=False)
failed_category_counts = failed_df['category'].value_counts().sort_values(ascending=False)

fig = make_subplots(rows=1, cols=2, subplot_titles=['Успішні проекти', 'Невдалі проекти'])
fig.add_trace(go.Bar(x=successful_category_counts.index, y=successful_category_counts.values,
        marker_color='#3498db', name='Успішні проекти'), row=1, col=1)
fig.add_trace(go.Bar(x=failed_category_counts.index, y=failed_category_counts.values,
        marker_color='#ff7f0e', name='Невдалі проекти'), row=1, col=2)
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(title_text='Розподіл топ-10 категорій за станом проектів',
        xaxis=dict(showgrid=False, zeroline=False),
        yaxis=dict(showgrid=True, zeroline=False),
        paper_bgcolor='rgb(233,233,233)',
        plot_bgcolor='rgb(233,233,233)',
        showlegend=False)
fig.show()

# Які основні категорії мають найвищий/найнижчий рівень успішності та невдач?
successful_df = df[df['state'] == 'successful']
failed_df = df[df['state'] == 'failed']
successful_category_counts =
successful_df['main_category'].value_counts().sort_values(ascending=False)
failed_category_counts = failed_df['main_category'].value_counts().sort_values(ascending=False)

fig = make_subplots(rows=1, cols=2, subplot_titles=['Успішні проекти', 'Невдалі проекти'])
fig.add_trace(go.Bar(x=successful_category_counts.index, y=successful_category_counts.values,
        marker_color='#3498db', name='Успішні проекти'), row=1, col=1)
fig.add_trace(go.Bar(x=failed_category_counts.index, y=failed_category_counts.values,
        marker_color='#ff7f0e', name='Невдалі проекти'), row=1, col=2)
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(title_text='Розподіл основних категорій за станом проектів',
        xaxis=dict(showgrid=False, zeroline=False),
        yaxis=dict(showgrid=True, zeroline=False),
        paper_bgcolor='rgb(233,233,233)',
        plot_bgcolor='rgb(233,233,233)',
        showlegend=False)
fig.show()

# Чи впливає мета фінансування на результат проекту?
```

```
limited_df = df.sample(min(10000, len(df)))
fig = px.box(limited_df, x='state', y='goal', color='state',
        title='Зв'язок між станом проекту та метою фінансування (обмежено 10 000 рядками)',
        color_discrete_sequence=['#ff7f0e', '#3498db'],
        labels={'state': 'Стан проекту', 'goal': 'Мета фінансування'},
        template='plotly_white')
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(showgrid=False, zeroline=False),
  yaxis=dict(zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Чи сприяє більша кількість бекерів успіху проекту?
fig = px.histogram(df, x='state', y='backers', color='state',
            title='Розподіл кількості бекерів за станом проекту',
            labels={'state': 'Стан проекту', 'backers': 'Кількість бекерів'},
            color_discrete_sequence=['#ff7f0e', '#3498db'])
fig.update_layout(
  bargap=0.1,
  xaxis_title='Стан проекту',
  yaxis_title='Кількість бекерів'
)
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(showgrid=False, zeroline=False),
  yaxis=dict(zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Чи впливає кількість бекерів на суму, залучену у USD?
fig = px.scatter(df, x='backers', y='usd_pledged_real',
          color_discrete_sequence=['#3498db'])
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(title="Кількість бекерів", showgrid=False, zeroline=False),
  yaxis=dict(title="Сума, залучена (USD)", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Чи впливає країна на успіх чи невдачу проекту?
fig = px.histogram(df, x='country', color='state',
            title='Рівень успішності за країнами',
            labels={'country': 'Країна', 'state': 'Стан проекту'},
            template='plotly_white', barmode='group',
            color_discrete_sequence=['#ff7f0e', '#3498db'])
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(showgrid=False, zeroline=False),
  yaxis=dict(title="Кількість", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()
```

```
# Групування даних за країною та станом, розрахунок рівня успішності
country_state_counts = df.groupby(['country', 'state']).size().unstack(fill_value=0)
country_state_counts['success_rate'] = country_state_counts['successful'] / (
  country_state_counts['successful'] + country_state_counts['failed'])

# Створення карти хороплет для рівня успішності за країнами
fig = px.choropleth(country_state_counts,
            locations=country_state_counts.index,
            locationmode="country names",
            color='success_rate',
            hover_name=country_state_counts.index,
            color_continuous_scale=px.colors.sequential.Plasma,
            title='Рівень успішності проектів за країнами',
            labels={'success_rate': 'Рівень успішності'})
fig.show()

# --------------------------
## 4- Передобробка даних ⚒️

### 4.1- Витяг ознак

# Перетворення стовпців 'launched' та 'deadline' у формат datetime
df['launched'] = pd.to_datetime(df['launched'])
df['deadline'] = pd.to_datetime(df['deadline'])

# Додавання нових стовпців: день, місяць та рік запуску
df = df.assign(day_launched = df.launched.dt.day,
        month_launched = df.launched.dt.month,
        year_launched = df.launched.dt.year)

# Обчислення тривалості проекту (у днях)
df['project_duration_days'] = (df['deadline'] - df['launched']).dt.days

# Функція для визначення пори року за номером місяця
def get_season(month):
  if month in [12, 1, 2]:
    return 'Зима'
  elif month in [3, 4, 5]:
    return 'Весна'
  elif month in [6, 7, 8]:
    return 'Літо'
  elif month in [9, 10, 11]:
    return 'Осінь'
  else:
    return 'Некоректний місяць'

# Створення стовпця з сезонами
df['monthly_seasons'] = df['month_launched'].apply(get_season)

# Видалення стовпців 'launched' та 'deadline'
df = df.drop(['launched', 'deadline'], axis=1)
df.head()

### Чи змінювався рівень успішності проектів протягом років, місяців та днів?

# Рівень успішності за роками
yearly_state_counts = df.groupby(['year_launched', 'state']).size().unstack(fill_value=0)
yearly_state_counts['success_rate'] = yearly_state_counts['successful'] /
(yearly_state_counts['successful'] + yearly_state_counts['failed'])
```

```
fig = px.line(yearly_state_counts, x=yearly_state_counts.index, y='success_rate',
        title='Рівень успішності та невдач проектів протягом років',
        color_discrete_sequence=['#1f77b4'])
fig.update_layout(
  xaxis=dict(title="Рік запуску", showgrid=False, zeroline=False),
  yaxis=dict(title="Рівень успішності", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Рівень успішності за місяцями
monthly_state_counts = df.groupby(['month_launched', 'state']).size().unstack(fill_value=0)
monthly_state_counts['success_rate'] = monthly_state_counts['successful'] /
(monthly_state_counts['successful'] + monthly_state_counts['failed'])
fig = px.line(monthly_state_counts, x=monthly_state_counts.index, y='success_rate',
        title='Рівень успішності проектів протягом місяців',
        color_discrete_sequence=['#1f77b4'])
fig.update_layout(
  xaxis=dict(title="Місяць запуску", showgrid=False, zeroline=False),
  yaxis=dict(title="Рівень успішності", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Рівень успішності за днями
daily_state_counts = df.groupby(['day_launched', 'state']).size().unstack(fill_value=0)
daily_state_counts['success_rate'] = daily_state_counts['successful'] /
(daily_state_counts['successful'] + daily_state_counts['failed'])
fig = px.line(daily_state_counts, x=daily_state_counts.index, y='success_rate',
        title='Рівень успішності проектів за днями',
        color_discrete_sequence=['#1f77b4'])
fig.update_layout(
  xaxis=dict(title="День запуску", showgrid=False, zeroline=False),
  yaxis=dict(title="Рівень успішності", zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# Вплив тривалості проекту на рівень успішності
duration_success_rate = df.groupby('project_duration_days')['state'].apply(lambda x: (x ==
'successful').mean()).reset_index()
duration_success_rate.columns = ['project_duration_days', 'Рівень успішності']
fig = px.scatter(duration_success_rate, x='project_duration_days', y='Рівень успішності',
        title='Вплив тривалості проекту на рівень успішності',
        labels={'project_duration_days': 'Тривалість проекту (дні)', 'Рівень успішності': 'Рівень
успішності'},
        color_discrete_sequence=['#1f77b4'])
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(showgrid=False, zeroline=False),
  yaxis=dict(zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

# У якому сезоні бекери роблять найбільший внесок?
```

```
monthly_backers = df.groupby('monthly_seasons').agg({'backers': 'sum'}).sort_values(by='backers',
ascending=False)
fig = px.bar(monthly_backers, x=monthly_backers.index, y='backers',
        title="Внесок бекерів за сезонами",
        color_discrete_sequence=['#1f77b4'])
fig.update_layout(
  bargap=0.1,
  xaxis_title='Сезон',
  yaxis_title='Загальна кількість бекерів'
)
fig.update_traces(marker=dict(line=dict(width=2, color='DarkSlateGrey')))
fig.update_layout(
  xaxis=dict(showgrid=False, zeroline=False),
  yaxis=dict(zeroline=False, gridcolor='white'),
  paper_bgcolor='rgb(233,233,233)',
  plot_bgcolor='rgb(233,233,233)',
)
fig.show()

### 4.2- Обробка пропущених даних

print("Кількість пропущених значень по стовпцях:")
print(df.isna().sum())

# Заповнення пропущених значень у стовпці 'usd pledged' за допомогою середнього значення
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputed_column = imputer.fit_transform(df['usd pledged'].values.reshape(-1, 1))
df['usd pledged'] = imputed_column

print("Після заповнення пропущених значень:")
print(df.isna().sum())

### 4.3- Обробка викидів

outliers_indices = detect_outliers(df, features=numerical_features, n=0)
number_of_outliers = len(outliers_indices)
print(f"Кількість викидів: {number_of_outliers}")

df = df.drop(outliers_indices)
print(f"Розмір даних після видалення викидів: {df.shape}")

### 4.4- Обробка категоріальних даних

# Перетворення порядкового стовпця 'state'
targ_dic = {'failed': 0, 'successful': 1}
df['state'] = df['state'].map(targ_dic)

categorical_cols = ['category', 'main_category', 'currency', 'country', 'monthly_seasons']
encoder = BinaryEncoder()
transformed_df = encoder.fit_transform(df[categorical_cols])
df = pd.concat([df, transformed_df], axis=1)
df = df.drop(categorical_cols, axis=1)
df.head()

### 4.5- Розбиття даних на навчальний та тестовий набори

X = df.drop(['state'], axis=1)
y = df['state']
print("Розміри X та y:", X.shape, y.shape)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                    test_size=0.2,
                                    random_state=42,
                                    stratify=y)
print("Навчальний набір містить {} зразків.".format(X_train.shape[0]))
print("Тестовий набір містить {} зразків.".format(X_test.shape[0]))

### 4.6- Обробка незбалансованих даних

print("Розподіл класів у навчальному наборі до SMOTE:")
print(y_train.value_counts())
sm = SMOTE(random_state=42)
X_train, y_train = sm.fit_resample(X_train, y_train)
print("Розподіл класів у навчальному наборі після SMOTE:")
print(y_train.value_counts())

### 4.7- Масштабування ознак

numerical_features = ['goal','pledged', 'backers', 'usd pledged', 'usd_pledged_real',
          'usd_goal_real', 'project_duration_days']
scaler = StandardScaler()
scaler.fit(X_train[numerical_features])
X_train_cont_scaled = scaler.transform(X_train[numerical_features])
X_test_cont_scaled = scaler.transform(X_test[numerical_features])
X_train[numerical_features] = X_train_cont_scaled
X_test[numerical_features] = X_test_cont_scaled
X_train

# ----------------------------
### 5- Навчання та оцінка моделей ⚙

# Список класифікаторів для оцінювання
classifiers = [
  ("Логістична регресія", LogisticRegression(random_state=42, max_iter=1500, n_jobs=-1)),
  ("Дерево рішень", DecisionTreeClassifier(random_state=42)),
  ("Випадковий ліс", RandomForestClassifier(random_state=42, n_jobs=-1)),
  ("AdaBoost", AdaBoostClassifier(random_state=42)),
  ("Gradient Boosting", GradientBoostingClassifier(random_state=42)),
  ("LightGBM", lgb.LGBMClassifier(random_state=42, verbose=-1)),
  ("XGBoost", xgb.XGBClassifier(random_state=42, n_jobs=-1))
]

### 5.1- Оцінювання за допомогою крос-валідації (K-fold)

results = []
mean_test_accuracy_scores = []
classifier_names = []

for model_name, model in classifiers:
  print(f"Для {model_name}:")
  steps = []
  steps.append((model_name, model))
  pipeline = Pipeline(steps=steps)
  cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
  cv_results = cross_validate(pipeline, X_train, y_train, cv=cv, scoring='accuracy', n_jobs=-1,
return_train_score=True)
  print(f"Крос-валідація успішно завершена для {model_name}")
  print('*' * 50)
  results.append({
    "Назва моделі": model_name,
```

```
            "Середня точність на навчальному наборі": np.mean(cv_results['train_score']),
            "Середня точність на тестовому наборі": np.mean(cv_results['test_score'])
          })
          mean_test_accuracy_scores.append(np.mean(cv_results['test_score']))
          classifier_names.append(model_name)

        results_df = pd.DataFrame(results)
        display(results_df)

        # Графік середніх тестових точностей за класифікаторами
        data = pd.DataFrame({'Класифікатор': classifier_names, 'Точність тесту': mean_test_accuracy_scores})
        fig = px.bar(data, x='Точність тесту', y='Класифікатор', orientation='h', color='Точність тесту',
                     title='Середня точність тесту за класифікаторами', text='Точність тесту',
color_continuous_scale='viridis')
        fig.update_layout(
          xaxis_title='Точність тесту',
          yaxis_title='Класифікатор',
          xaxis=dict(range=[0, 1]),
          yaxis=dict(categoryorder='total ascending'),
          showlegend=False,
          height=500,
          width=900
        )
        fig.show()

        # ## 5.2- Навчання обраної моделі (XGBoost)

        pipeline = Pipeline(steps=[
          ("XGBoost", xgb.XGBClassifier(random_state=42, n_jobs=-1))
        ])
        pipeline.fit(X_train, y_train)
        y_pred = pipeline.predict(X_test)

        print(f"Модель: XGBoost")
        print(f"Точність на навчальному наборі: {accuracy_score(y_train, pipeline.predict(X_train))}")
        print(f"Точність на тестовому наборі: {accuracy_score(y_test, y_pred)}")
        print("----------------------------------------------------------------------")
        print("Матриця плутанини на тестовому наборі:")
        print(confusion_matrix(y_test, y_pred))
        print("----------------------------------------------------------------------")
        print("Звіт класифікації на тестовому наборі:")
        print(classification_report(y_test, y_pred))
        print("----------------------------------------------------------------------")

        # ---------------------------
        # # 6- Налаштування гіперпараметрів 🛠️

        param_grid = {
          'XGBoost__learning_rate': [0.01, 0.1, 0.2],
          'XGBoost__n_estimators': [100, 200, 300],
          'XGBoost__max_depth': [3, 4, 5],
        }

        steps = []
        steps.append(("XGBoost", xgb.XGBClassifier(random_state=42, n_jobs=-1)))
        pipeline = Pipeline(steps=steps)

        grid_search = GridSearchCV(estimator=pipeline, param_grid=param_grid, cv=5,
scoring='accuracy', n_jobs=-1,
                   return_train_score=True)
```

```
grid_search.fit(X, y)
best_params = grid_search.best_params_
best_score = grid_search.best_score_
print("Найкращі параметри:", best_params)
print("Найкращий результат:", best_score)

mean_test_score = grid_search.cv_results_['mean_test_score'][grid_search.best_index_]
mean_train_score = grid_search.cv_results_['mean_train_score'][grid_search.best_index_]
print("Середній тестовий бал:", mean_test_score)
print("Середній навчальний бал:", mean_train_score)

final_model = grid_search.best_estimator_
print("Кінцева модель:")
print(final_model)

# ## 6.1- ROC-крива для фінальної моделі (XGBoost)

y_probabilities = final_model.predict_proba(X)[:, 1]
fpr, tpr, thresholds = roc_curve(y, y_probabilities)
auc = roc_auc_score(y, y_probabilities)
sns.set(style='whitegrid')
plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC-крива (AUC = {auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Хибно позитивна частота')
plt.ylabel('Правдива позитивна частота')
plt.title('ROC-крива для XGBoost класифікатора')
plt.legend(loc='lower right')
plt.show()
```

Додаток Б

Ілюстративні матеріали до дослідження

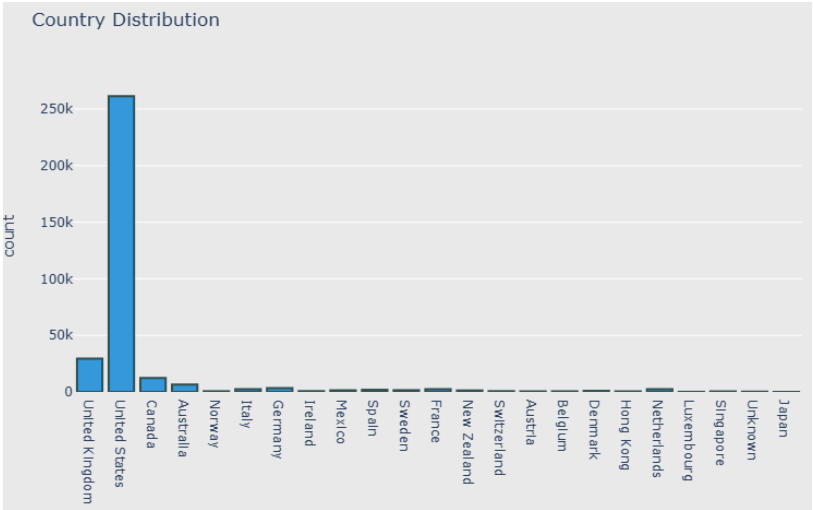

Рисунок Б.1 – Розподіл країн

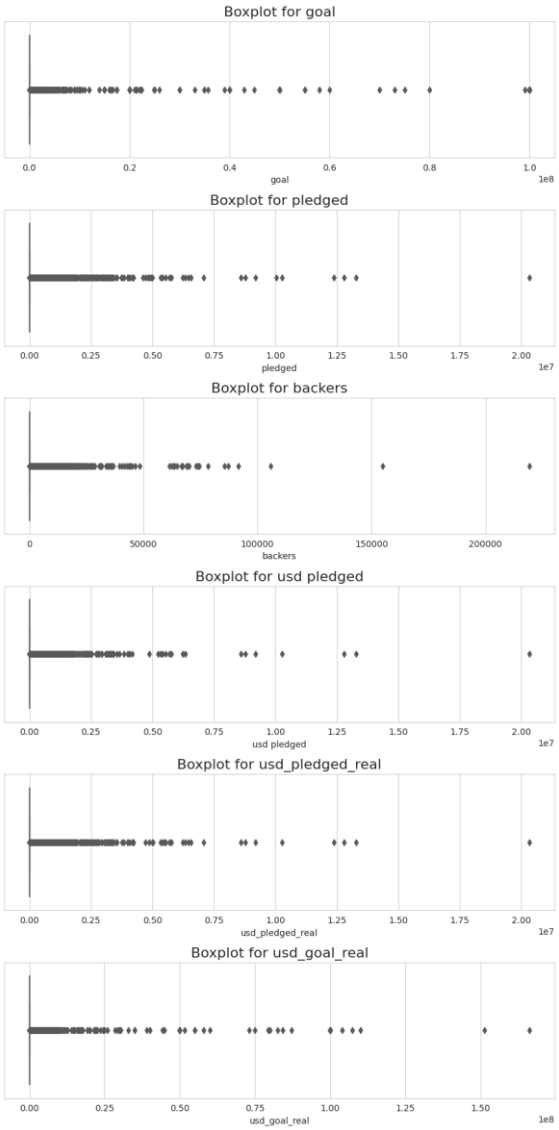

Рисунок Б.2 – Boxplot числових ознак

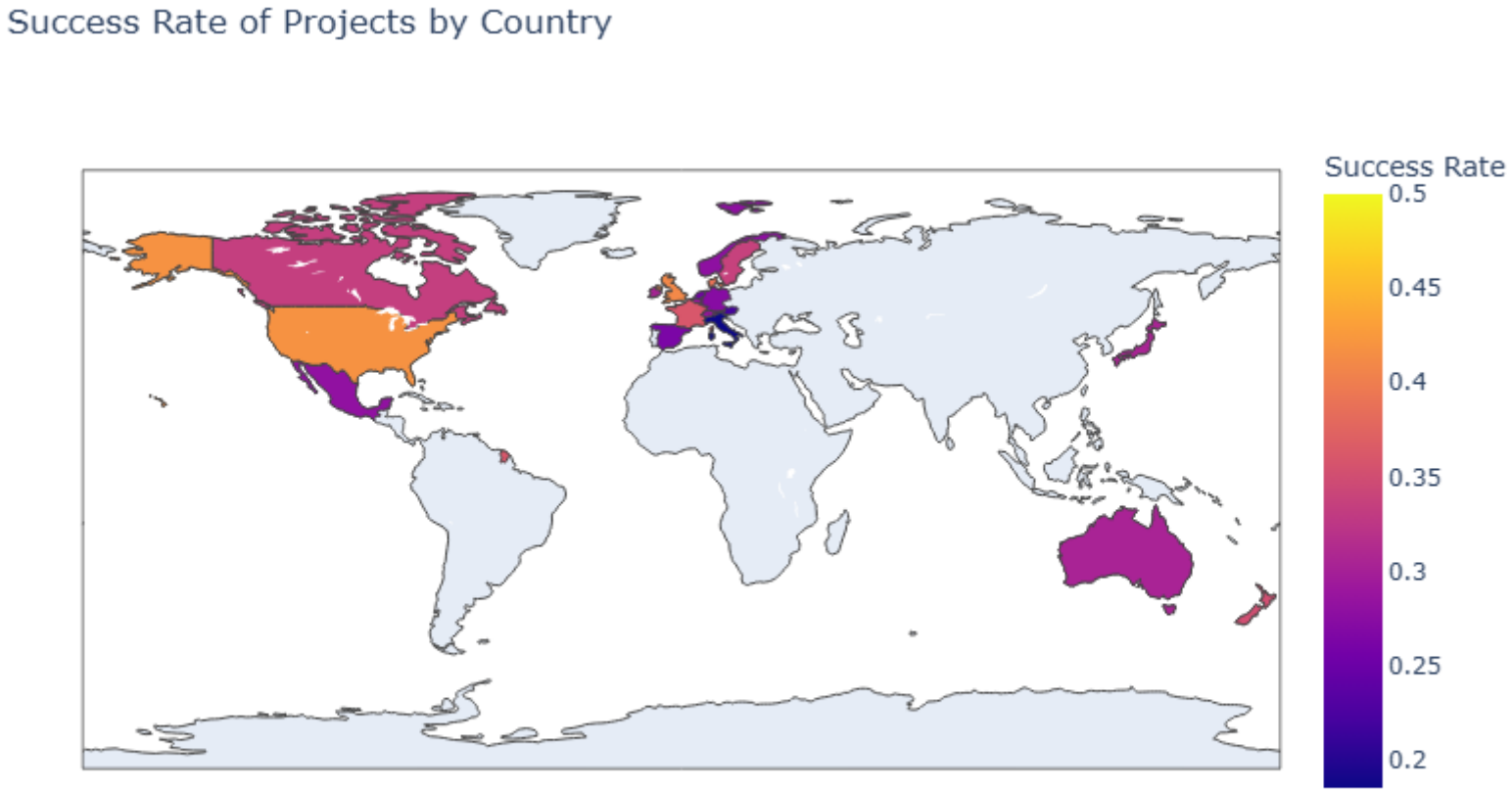

Рисунок Б.3 – Географічна карта успішності проєктів за країнами

Додаток В

Апробація отриманих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(IITAR-2025)

27-29 травня 2025 року

Тернопіль

2025

Онанко Вадим
студент групи КН-41
onanko2004@gmail.com
Лендюк Тарас
к.т.н., доцент
tl@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

ПРОГНОЗУВАННЯ УСПІШНОСТІ ЗАПУСКУ ПРОДУКТУ НА РИНКУ ЗАСОБАМИ КЛАСИФІКАЦІЙНИХ МОДЕЛЕЙ

Краудфандингові платформи типу Kickstarter дають стартапам демократичний доступ до капіталу, однак менш ніж половина кампаній досягає цільової суми. Тому виникає потреба у автоматизованому інструменті, який ще до старту кампанії оцінює її шанси на успіх та мінімізує фінансові ризики авторів і бекерів. У роботі запропоновано програмний модуль, що використовує методи машинного навчання для прогнозування бінарного результату successful / failed.

Основою дослідження став відкритий набір ks-projects-201801.csv (≈ 331 тис. записів). Видалено неінформативні поля ID, name, після чого залишено лише дві релевантні мітки стану проєкту. Було виявлено 210 пропусків у стовпці usd pledged (заповнені середнім значенням) та ≈ 87 тис. викидів у числових показниках (goal, pledged, backers тощо), які прибрано для зменшення шуму . З часових міток launched / deadline сформовано тривалість кампанії, пору року та окремі компоненти дати, а 5 категоріальних ознак закодовано методом Binary Encoding.

Через дисбаланс (≈ 60 % невдалих кампаній) застосовано SMOTE, що зрівняв вибірку до 128 827 прикладів кожного класу. Числові ознаки стандартизовано StandardScaler, забезпечивши нульове середнє та одиничне σ, що критично для регуляризованих та бустингових моделей .

У конвеєрі Sklearn + Imbalanced-learn протестовано вісім класифікаторів: Logistic Regression, Decision Tree, Random Forest, AdaBoost, Gradient Boosting, LightGBM, XGBoost та базовий KNN. 5-кратна стратифікована CV показала, що ансамблеві методи істотно перевершують лінійні: XGBoost досяг середньої точності 0,9996, тоді як найближчий конкурент LightGBM—0,9995; AdaBoost був найслабшим (0,9903) .

Після тюнінгу XGBoost (learning_rate = 0,2; max_depth = 5; n_estimators = 300) забезпечив Testing Accuracy = 99,93 % та F1 = 1,00 для обох класів на відкладених 48 754 прикладах; лише 35 проєктів класифіковано помилково, що вказує на високу узагальнювальну здатність алгоритму .

Структура модуля — від завантаження даних до видачі прогнозу — реалізована як послідовний програмний конвеєр з чіткою декомпозицією на

блоки EDA → Pre-processing → Encoding/Scaling → SMOTE → Model Selection → Inference. Детальна схема подана на рисунку 1.

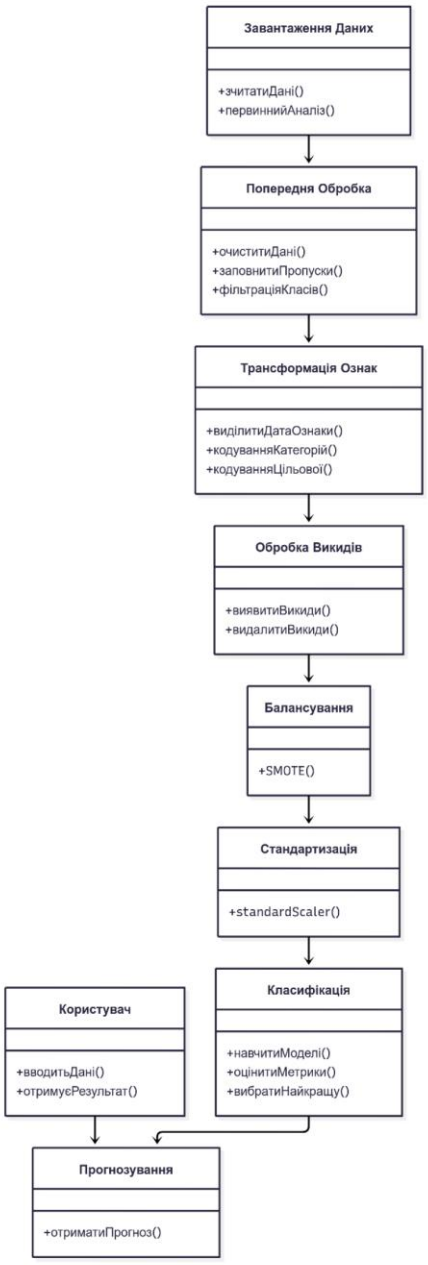

Рисунок 1 – Архітектура програмного модуля прогнозування успішності краудфандингових кампаній

Запропонована система може інтегруватися у веб-інтерфейс платформи та надавати авторам миттєвий аналітичний відгук про життєздатність їхньої ідеї, а інвесторам — незалежний індикатор ризику. Модуль масштабований, оскільки використовує кодування з лінійною складністю та дозволяє оновлювати модель при надходженні нових даних без повного перенавчання.

360

Планується розширення ознак поведінковими метриками, інтерпретація SHAP-значень для формування рекомендацій авторам, а також перевірка моделі на інших краудфандингових платформах (Indiegogo, GoFundMe) для оцінки переносимості.

Список використаних джерел

1. Chen, Y., & Zhai, L. (2023). *A comparative study on student performance prediction using machine learning*. Retrieved from https://link.springer.com/article/10.1007/s10639-023-11672-1
2. Yıldız, M., & Börekci, C. (2020). *Predicting academic achievement with machine learning algorithms*. Retrieved from https://dergipark.org.tr/en/pub/jetol/article/773206
3. Holicza, B., & Kiss, A. (2023). *Predicting and comparing students' online and offline academic performance using machine learning algorithms*. *Behavioral Sciences*, 13(4), 289. https://doi.org/10.3390/bs13040289
4. Alyahyan, E., & Düştegör, D. (2020). *Predicting academic success in higher education: literature review and best practices*. Retrieved from https://link.springer.com/article/10.1186/s41239-020-0177-7
5. Ghorbani, R., & Ghousi, R. (2020). *Comparing different resampling methods in predicting students' performance using machine learning techniques*. Retrieved from https://ieeexplore.ieee.org/abstract/document/9062549/