МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Комп'ютерна система прогнозування результатів спортивних змагань**

Виконав: студент 4 курсу, групи 6.121-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
М. С. Чистіков
(ініціали та прізвище)
Керівник к.т.н., доцент Заяц В.І.
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дискус»
Р.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Тетяна КРИТСЬКА_
“_01_” ___березня__ 2025_року

З А В Д А Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Чистікову Максиму Сергійовичу
(прізвище, ім’я, по батькові)

1.Тема роботи __Комп’ютерна система прогнозування результатів спортивних змагань
керівник роботи ___Заяц Валерій Іванович, доцент, к.т.н._____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 26.12.2024 №2215-с

2. Строк подання студентом кваліфікаційної роботи_____07.06.2025_____
3. Вихідні дані бакалаврської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- створення програмного продукту та його опис;
- дослідження поставленої проблеми та розробка висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
_____15 слайдів презентації

6. Консультанти розділів бакалаврської роботи

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

7. Дата видачі завдання 01.03.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Аналіз предметної області спортивної аналітики та футбольної статистики	15.03.25	виконано
2	Формулювання задачі прогнозування результатів матчів та узгодження її з науковим керівником	17.03.25	виконано
3	Аналіз існуючих математичних моделей прогнозування (рейтингів, ймовірностей, машинного навчання)	18.03.25	виконано
4	Збір та підготовка історичних даних про футбольні матчі (результати, коефіцієнти, склади)	19.03–25.03.25	виконано
5	Узгодження структури моделі та способів обробки даних	30.03.25	виконано
6	Проєктування математичної моделі прогнозування результатів матчів	01.04–15.04.25	виконано
7	Розробка програмного забезпечення для побудови та перевірки моделі	16.04–30.04.25	виконано
8	Реалізація інтерфейсу для введення/виводу матчів та прогнозів	01.05–10.05.25	виконано
9	Апробація моделі: тестування на реальних даних, порівняння з фактичними результатами	11.05–20.05.25	виконано
10	Аналіз точності та стабільності результатів прогнозування	21.05–25.05.25	виконано
11	Оформлення пояснювальної записки	26.05–05.06.25	виконано
12	Підготовка презентації, отримання рецензії від наукового керівника та рецензента	06.06–10.06.25	виконано

Студент _____ Чистіков М.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Заяц В.І.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено
Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок — 66

Рисунків — 16

Джерел — 15

Чистіков М.С. Комп’ютерна система прогнозування результатів спортивних змагань: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник В. І. Заяц. — Запоріжжя: ЗНУ, 2025. 66 с.

Метою даного дослідження є створення математичної моделі для прогнозування результатів футбольних матчів із використанням методів статистики, машинного навчання та аналізу історичних даних. У процесі роботи проведено аналіз предметної області спортивної аналітики, вивчено методи прогнозування на основі ймовірнісних моделей (Poisson, Logit), а також застосовано елементи рейтингових систем.

Розроблено програмний прототип, що збирає футбольні дані, обробляє їх та генерує прогнози результатів. Значну увагу приділено точності моделі, налаштуванню гіперпараметрів та візуалізації результатів для кінцевого користувача.

Ключові слова: прогнозування, футбольний матч, статистична модель, машинне навчання, Python.

ABSTRACT

Pages — 66

Drawings — 16

Sources — 15

Chystikov M.S. Computer system for predicting sports results: bachelor's thesis in specialty 121 "Software Engineering" / Scientific adviser V. I. Zaiats. — Zaporizhzhia: ZNU, 2025. 66 p.

The goal of this research is to develop a mathematical model for predicting football match outcomes using statistical techniques, machine learning, and historical match data analysis. The study covers modern approaches in sports analytics, including probabilistic models (Poisson, Logit) and rating-based prediction methods.

A prototype software system was implemented to collect, process and predict football match results. Particular focus was placed on accuracy, model tuning, and visualization of results for end users.

Keywords: prediction, football, statistical model, machine learning, Python.

ЗМІСТ

ВСТУП 8

1 АНАЛІЗ ПРОБЛЕМИ 11

1.1 Огляд літературних джерел 11

1.2 Аналіз існуючих аналогів 13

1.3 Постановка завдання 17

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ТА ТЕСТУВАННЯ 20

2.1 Середовища розробки для Python 20

2.2 Огляд середовищ та мов програмування для розробки програмного забезпечення 22

2.2.1 Python 22

2.2.2 Мова R 22

2.2.3 Система MATLAB 23

2.2.4 Google Colab / Jupyter Notebook 23

2.3 Аналіз вимог до апаратного та програмного забезпечення 23

3 ТЕОРЕТИЧНІ ОСНОВИ ШТУЧНОГО ІНТЕЛЕКТУ ТА ПОБУДОВИ МОДЕЛЕЙ ПРОГНОЗУВАННЯ 25

3.1 Поняття та класифікація штучного інтелекту 25

3.2 Основи машинного навчання 27

3.3 Основи статистичного моделювання у прогнозуванні 31

3.3.1 Роль статистичного моделювання в машинному навчанні 31

3.3.2 Значення статистики в машинному навчанні 32

3.3.3 Модель Random Forest: суть та авторство 32

3.3.4 Алгоритм Random Forest як основа моделі прогнозування... 33

3.3.5 Метрики оцінки ефективності моделі..... 34

3.4 Побудова моделі прогнозування результатів матчів 35

3.5 Інструменти розробки моделей ШІ..... 40

4. РОЗРОБКА СИСТЕМИ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ ФУТБОЛЬНИХ МАТЧІВ НА ОСНОВІ СТАТИСТИЧНОЇ МОДЕЛІ..... 43

4.1 Опис предметної області..... 43

4.2 Функціональні вимоги системи..... 44

4.2.1 Загальні вимоги..... 44

4.3 Архітектура системи..... 46

4.4 Функціонал системи прогнозування результатів матчів 48

4.5 Реалізація і тестування 49

4.5.1 Реалізація 49

4.5.2 Ключові переваги реалізованої системи..... 57

4.5.3 Апробація системи..... 57

4.6 Перевірка прогнозів моделі 62

4.7 Висновок з розділу 4..... 64

ВИСНОВКИ..... 65

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 66

ВСТУП

Актуальність теми

У сучасному світі аналітика спортивних подій, зокрема футбольних матчів, набуває все більшого значення як для професійних аналітиків, так і для уболівальників, букмекерських компаній і трейдерів на ринку спортивних ставок. В умовах стрімкого розвитку цифрових технологій та зростання обсягів історичних спортивних даних виникає необхідність у розробці точних математичних моделей, здатних здійснювати якісне прогнозування результатів матчів.

Особливої актуальності тема набуває з огляду на те, що більшість сучасних систем прогнозування ґрунтуються або на поверхневому аналізі коефіцієнтів букмекерів, або на евристичних підходах, що не враховують повною мірою структуру даних та ймовірнісну природу спортивних подій.

Використання статистичних моделей, таких як Poisson-розподіл, логістична регресія чи моделі машинного навчання, дозволяє формалізувати процес прогнозування і досягти вищої точності. Вони відкривають можливість адаптивного підходу до врахування форми команд, місця проведення матчу, історії особистих зустрічей та інших ключових факторів.

Таким чином, розробка і апробація статистичної моделі для прогнозування футбольних матчів є не лише теоретично значущим, а й практично важливим завданням, яке може знайти застосування в реальних системах підтримки прийняття рішень.

Мета дослідження

Метою роботи є розробка та оцінка ефективності статистичної (математичної) моделі, яка дозволяє прогнозувати результати футбольних матчів на основі історичних даних, з використанням методів математичної статистики та алгоритмів машинного навчання.

Завдання дослідження

1. Проаналізувати існуючі підходи до математичного прогнозування результатів спортивних подій.
2. Зібрати та підготувати вибірку історичних даних про футбольні матчі (результати, склади, коефіцієнти букмекерів, тощо).
3. Обрати та реалізувати математичні моделі (Poisson, логістична регресія, XGBoost або інші), що можуть використовуватись для прогнозування.
4. Навчити моделі на реальних даних, оцінити їхню точність та стабільність.
5. Реалізувати програмне забезпечення для інтеграції моделі з вхідними даними та виводу прогнозів.
6. Провести апробацію розробленої системи на прикладах реальних турнірів (наприклад, Чемпіонат України, Англійська Прем'єр-ліга).

Об’єкт дослідження

Процес прогнозування результатів футбольних матчів з використанням формалізованих математичних методів.

Предмет дослідження

Математичні моделі ймовірнісного прогнозування результатів футбольних матчів, засоби їх реалізації та перевірки якості.

Методи дослідження

- Методи математичної статистики (оцінка параметрів, ймовірнісне моделювання);
- Методи машинного навчання (регресія, класифікація, бустинг);
- Методи порівняльного аналізу точності моделей;
- Програмна реалізація за допомогою Python (pandas, scikit-learn, xgboost, matplotlib).

Практичне значення одержаних результатів

1. Автоматизоване прогнозування матчів.

Модель може бути використана як основа для створення сервісу прогнозування результатів матчів у режимі реального часу.

2. Інструмент підтримки рішень.

   Модель може допомагати аналітикам при прийнятті рішень щодо ставок, спортивного менеджменту або оцінки сили команд.

3. Освітня цінність.

   Модель і програмний інструмент можуть бути використані як навчальний кейс у курсах статистики, машинного навчання, прикладного програмування.

4. Можливість масштабування.

   Запропонований підхід може бути адаптований під інші командні види спорту або турнірні формати.

Глосарій

- Poisson-модель — імовірнісна модель, що передбачає кількість голів як подію, яка підкоряється розподілу Пуассона.
- Логістична регресія — метод статистичного моделювання для оцінки ймовірності події (наприклад, перемоги команди).
- Машинне навчання (ML) — підхід до аналізу даних, який полягає у створенні моделей, що навчаються на даних.
- Accuracy, Precision, Recall — метрики оцінки якості класифікаційних моделей.
- XGBoost — потужна бібліотека градієнтного бустингу для побудови моделей класифікації/регресії.

1 АНАЛІЗ ПРОБЛЕМИ

1.1 Огляд літературних джерел

У сучасних дослідженнях, присвячених прогнозуванню спортивних подій, зокрема результатів футбольних матчів, все ширше застосовуються методи математичної статистики, машинного навчання та штучного інтелекту. Побудова ефективної моделі прогнозування вимагає не лише доступу до великих обсягів історичних даних, а й глибокого розуміння теоретичних підходів, що лежать в основі процесу навчання моделей, обробки ознак, інтерпретації результатів та оцінки якості прогнозів.

Фундаментальне підґрунтя для розуміння ймовірнісних моделей і принципів класифікації надає праця Крістофера Бішопа *"Pattern Recognition and Machine Learning"* [1]. У книзі докладно розглядаються такі методи, як логістична регресія, наївний баєсівський класифікатор, метод максимізації очікування (EM-алгоритм), багатовимірні нормальні розподіли та нейронні мережі. Особливо актуальними в контексті прогнозування результатів матчів є ймовірнісні моделі, здатні обчислювати оцінки ймовірності перемоги, нічиєї або поразки, що дозволяє розглядати задачу як багатокласову класифікацію.

З практичної точки зору, реалізація моделей машинного навчання потребує володіння відповідними інструментами та бібліотеками. У цьому контексті цінним ресурсом є книга Орельєна Жерона *"Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow"* [2], що містить численні практичні приклади побудови моделей у середовищі Python. Автор надає чіткі рекомендації щодо попередньої обробки даних (очищення, нормалізація, масштабування), інженерії ознак, відбору моделей, крос-валідації та оцінки якості прогнозів. У книзі також розглядаються ансамблеві методи (Random Forest, Gradient Boosting, XGBoost), які продемонстрували високу ефективність у задачах прогнозування з великим числом нерелевантних або шумових ознак.

Ще одне авторитетне джерело – *"An Introduction to Statistical Learning"* [3] під редакцією Гарета Джеймса та співавторів – акцентує увагу на інтерпретованості моделей та базових концепціях навчання з учителем. Книга надає приклади використання логістичної регресії, дерева рішень та методів регуляризації на реальних наборах даних. З огляду на необхідність пояснюваності моделі у прогнозуванні футбольних результатів (особливо для використання серед нефахівців), застосування простих, але прозорих моделей має важливе значення.

У сфері практичного збору та аналізу даних ключову роль відіграють онлайн-ресурси. Зокрема, платформа **Kaggle** [4] надає відкриті набори даних про футбольні матчі, коефіцієнти букмекерів, склади команд, а також реалізації моделей від спільноти дослідників. Такі сервіси, як **FiveThirtyEight** [6] та **Forebet** [7], публікують власні прогнозні моделі, засновані на статистичних або гібридних підходах, що дозволяє проводити бенчмаркінг власної моделі та порівнювати її прогностичну здатність.

Додаткове практичне значення мають ресурси, орієнтовані на професійних аналітиків. Наприклад, **Wyscout** [10] пропонує доступ до розширеної аналітики матчів, дій гравців, трансферних даних, що може бути використано для побудови комплексних моделей з урахуванням якісних характеристик команди. Окрім того, на **GitHub** [8] доступні численні відкриті проєкти, які демонструють реалізацію моделей прогнозування футбольних результатів із використанням бібліотек XGBoost, LightGBM, SVM тощо.

З технічної точки зору, найбільш поширеним стеком інструментів у сфері прогнозування спортивних подій є мова програмування Python [11] та її наукові бібліотеки: scikit-learn [12] для реалізації класичних моделей машинного навчання, pandas [13] для обробки табличних даних, matplotlib [14] і seaborn [15] для візуалізації, xgboost для високоефективного градієнтного бустингу. Такий стек забезпечує повний цикл роботи з даними — від завантаження та передобробки до моделювання та аналізу результатів.

Таким чином, теоретичні джерела та прикладні керівництва, що були проаналізовані, формують надійну базу знань для побудови системи прогнозування результатів футбольних матчів. Їх поєднання дозволяє врахувати як фундаментальні статистичні принципи, так і найкращі практики реалізації моделей у прикладних умовах.

1.2 Аналіз існуючих аналогів

Прогнозування результатів футбольних матчів є складною багатофакторною задачею, яка поєднує методи статистичного аналізу, машинного навчання, експертного оцінювання та роботу з великими обсягами історичних і поточних даних. У зв’язку з підвищеним інтересом до аналітики у спорті та ставках, на ринку з’явилася низка платформ, що реалізують власні підходи до моделювання результатів футбольних подій. Їх аналіз дозволяє виділити як переваги сучасних підходів, так і існуючі обмеження, що є важливими для формування вимог до розробки власної системи.

Одним із найвідоміших прикладів є аналітична платформа FiveThirtyEight (рис. 1), яка широко застосовується для прогнозування спортивних подій, включаючи футбол. Її модель враховує множину чинників: силу команди (rating), результати останніх матчів, домашній/виїзний статус гри, форму гравців та інші параметри. Основною перевагою підходу є його багаторівнева ймовірнісна структура та наявність симуляцій турнірів із розрахунком шансів команди на вихід у наступний етап. Водночас математична модель є закритою і не підлягає інтеграції в сторонні рішення — доступ користувача обмежений лише інтерфейсом веб-сайту.

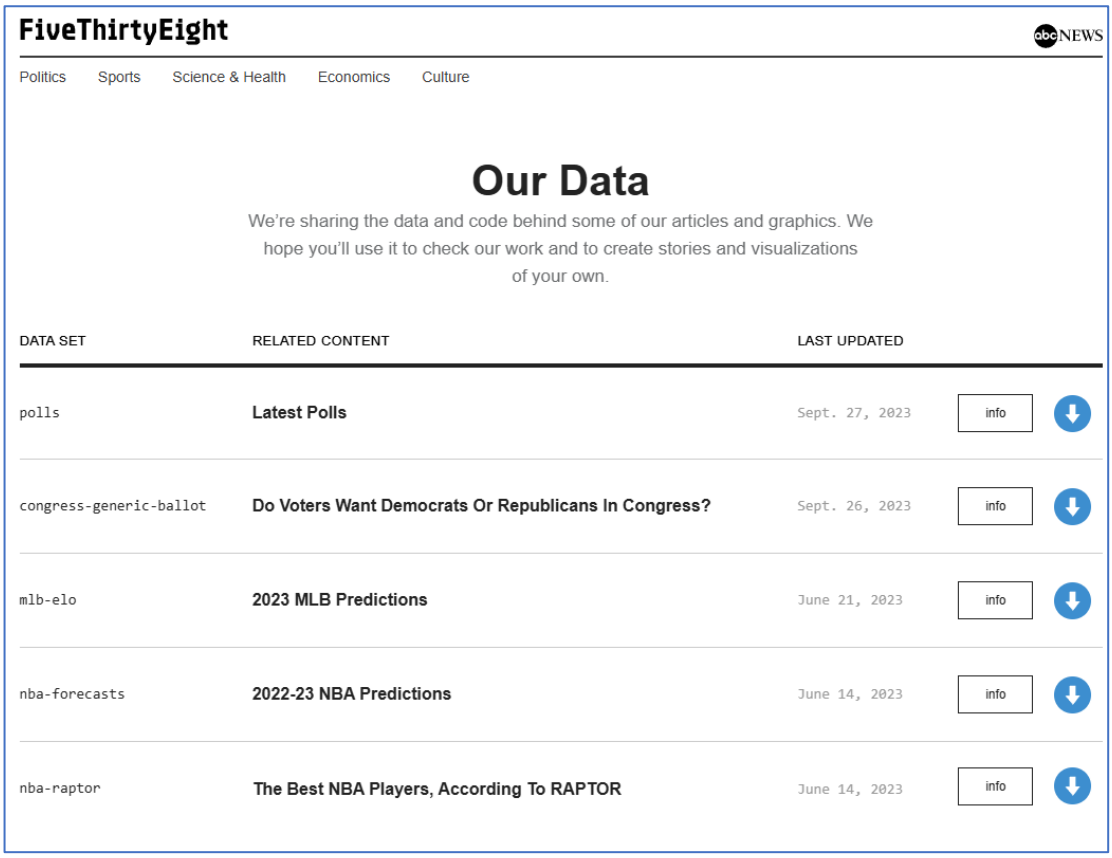


Рисунок *1 — Аналітична платформа FiveThirtyEight*

Інший приклад — платформа Forebet (рис. 2), яка генерує автоматизовані математичні прогнози на основі статистичного аналізу історичних результатів, тенденцій і розподілів. Ресурс пропонує ймовірності для кожного з можливих результатів матчу (перемога, нічия, поразка), а також прогнозований рахунок та інші параметри, як-от кількість голів чи імовірність "обох заб'ють". Хоча система є корисною для швидкого орієнтування у статистиці матчів, вона не надає користувачеві можливостей для адаптації чи інтеграції алгоритмів у власні моделі, що знижує її застосовність у наукових або прикладних розробках.

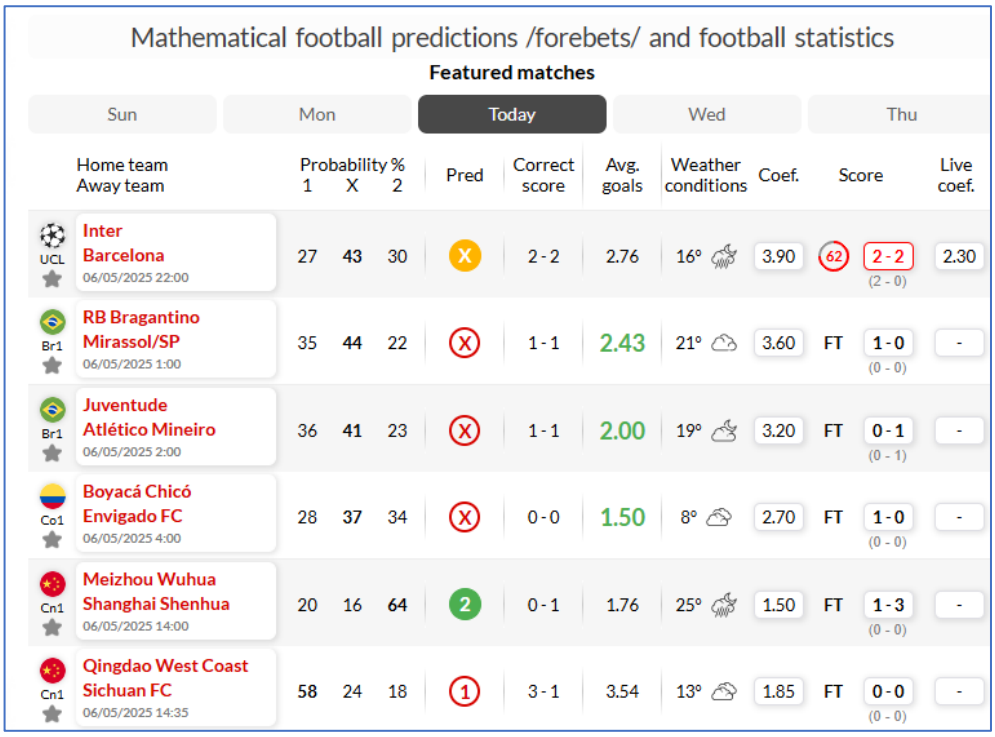


Рисунок *2* — *Аналітична платформа Forebet*

Платформа Betegy (рис. 3) поєднує статистичний аналіз з машинним навчанням для побудови прогнозів, орієнтуючись насамперед на ринок спортивних ставок. Вона враховує не лише історичні дані, а й динамічні фактори — травми ключових гравців, мотивацію команд, погодні умови, зміни у складі. Прогнози реалізуються у формі комерційного вебсервісу. Попри високу точність окремих прогнозів, система має обмежений рівень прозорості — методологія моделювання не є відкритою, а результати недоступні для перевірки або відтворення.

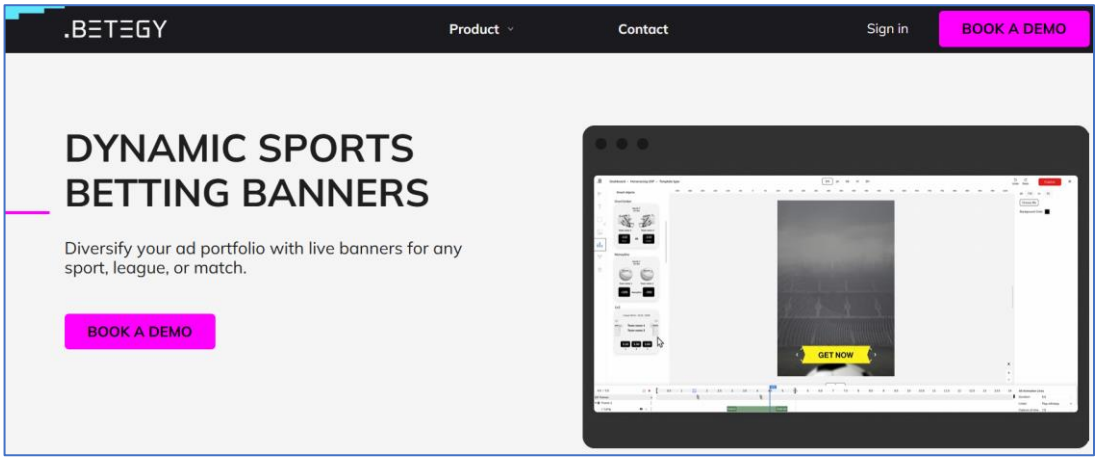


Рисунок *3* — *Аналітична платформа* Betegy

З-поміж новітніх проєктів варто відзначити Kickoff.ai (рис. 4), що застосовує штучний інтелект та методи машинного навчання, спеціалізуючись на матчах італійських чемпіонатів. Проєкт вирізняється тим, що надає відкриту документацію щодо джерел даних, попередньої обробки та основ алгоритмів, що створює умови для повторного використання чи вдосконалення у власних дослідженнях. Недоліком є обмежена географія охоплення (в основному Серія А та В Італії), а також обмежений функціонал щодо масового аналізу або порівняння прогнозів.

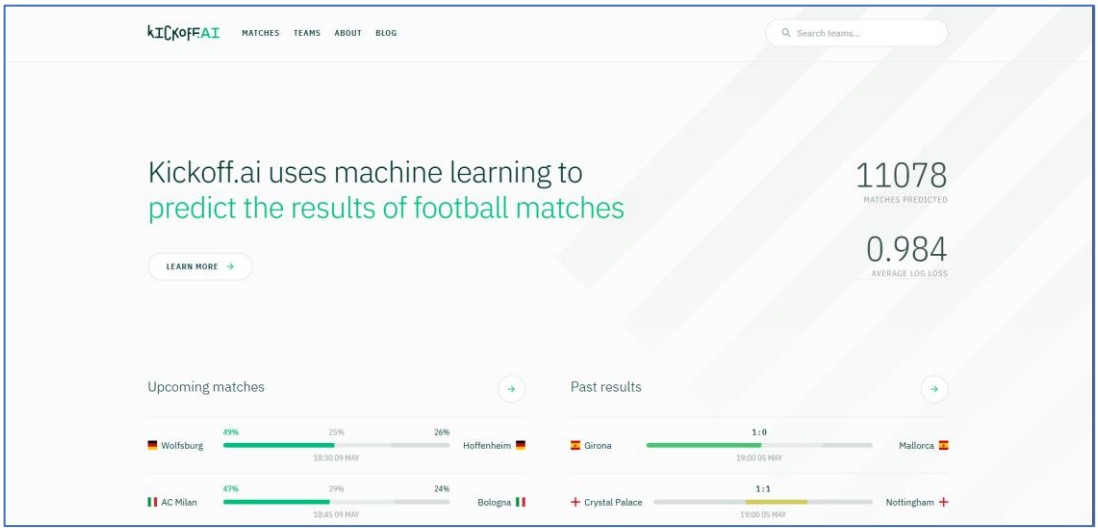


Рисунок *4 — Аналітична платформа* Kickoff.ai

Інші сервіси, як-от SoccerVista та PredictZ, надають спрощені прогнозні оцінки на основі історичної статистики та базових ймовірнісних розрахунків. Їх ключова перевага — простота використання та наочність, однак ці сервіси не мають глибокого моделювання або використання складних алгоритмів машинного навчання. Вони є радше довідковими інструментами для широкого загалу, ніж повноцінними системами прогнозування.

Таким чином, аналіз існуючих аналогів дозволяє зробити кілька ключових висновків. По-перше, провідні сервіси активно використовують статистичні та ML-підходи, проте більшість із них є закритими або комерційними. По-друге, обмежений доступ до алгоритмів, даних і результатів ускладнює наукову валідацію та повторюваність досліджень. По-

третє, існує потреба у відкритих, гнучких і масштабованих рішеннях, які дозволяють не лише генерувати прогнози, а й проводити порівняльний аналіз моделей, адаптувати систему під нові турніри та змінні.

Результати цього аналізу є важливими для формування вимог до власної моделі прогнозування, яка має поєднувати точність, відкритість, гнучкість і практичну реалізованість.

1.3 Постановка завдання

Розробка ефективної системи прогнозування результатів футбольних матчів передбачає чітке формулювання цілей, задач, функціональних та технічних вимог до моделі та програмного забезпечення. Постановка завдання є ключовим етапом дослідження, що забезпечує цілісне бачення майбутньої системи, її призначення, логіку функціонування, критерії якості та очікувані результати. У цьому розділі окреслено основні компоненти проєкту, які визначають подальшу стратегію реалізації:

1.3.1 Опис системи:

• Визначити загальну мету системи — створення математичної моделі для прогнозування результатів матчів.

• Визначити цільову аудиторію — спортивні аналітики, любителі футболу, трейдери на ринку ставок.

• Окреслити ключові функціональні сценарії: завантаження історичних даних, запуск моделі, виведення прогнозу.

1.3.2 Функціональні вимоги:

• Забезпечити збір та обробку історичних даних (результати, коефіцієнти, склади, місце проведення).

• Реалізувати математичну модель прогнозування з можливістю навчання та перевірки точності.

• Створити інтерфейс для зручного введення нових матчів та виведення

прогнозу ймовірності перемоги/нічиєї/поразки.

• Передбачити можливість експорту результатів у зручному вигляді (CSV, таблиці, графіки).

1.3.3 Технічні вимоги:

• Мова реалізації — Python з використанням бібліотек scikit-learn, pandas, xgboost, matplotlib.

• Система має працювати в локальному середовищі (Jupyter Notebook, консольний застосунок або web-інтерфейс).

• Забезпечити масштабованість — можливість додавання нових турнірів, ознак, моделей.

• Забезпечити коректне логування, валідацію вхідних даних і обробку винятків.

1.3.4 Терміни виконання:

• Визначити покроковий план реалізації: збір даних, розробка моделі, тестування, оформлення звіту.\

1.3.5 Очікувані результати:

• Отримання статистичної моделі, що забезпечує прогнозування результатів з валідаційною точністю не нижче 60–70%.

• Функціональний програмний модуль або прототип із можливістю інтерактивної роботи.

• Аналітичне порівняння ефективності обраних моделей, візуалізація результатів.

• Рекомендації щодо подальшого вдосконалення.

1.3.6 Постановка завдання

Постановка завдання є ключовим етапом дослідження, оскільки вона формує чітке бачення вимог, цілей та очікувань, забезпечуючи основу для послідовного виконання проєкту.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ТА ТЕСТУВАННЯ

2.1 Середовища розробки для Python

Мова програмування Python є однією з провідних у галузях наукових обчислень, машинного навчання, аналізу даних та автоматизації. Ефективна робота з цією мовою передбачає використання спеціалізованих середовищ розробки, що забезпечують підтримку інструментів інтерактивного програмування, налагодження, візуалізації та управління залежностями. Нижче подано огляд найбільш популярних середовищ, які використовуються для розробки застосунків на Python.

У межах розробки системи прогнозування результати було досягнуто шляхом використання мови Python, яка забезпечила ефективну реалізацію повного циклу: від завантаження та обробки даних — до побудови моделі та створення графічного інтерфейсу. Серед використаних бібліотек: pandas, numpy, scikit-learn, matplotlib, seaborn, joblib, tkinter, а також rapidfuzz і deep_translator для обробки текстових даних та нормалізації назв команд.

1.PyCharm

PyCharm є потужним інтегрованим середовищем розробки (IDE), розробленим компанією JetBrains, спеціально для мови Python. Програма доступна у версіях Community (безкоштовна) та Professional (комерційна). Основні функціональні можливості включають:

- інтелектуальне автодоповнення коду, перевірку синтаксису, рефакторинг;
- інтеграцію з системами контролю версій (Git, Mercurial);
- підтримку юніт-тестування, статичного аналізу коду та інструментів деплойменту;

- розширення для Data Science, роботи з фреймворками Django, Flask та базами даних.

2. Visual Studio Code (VS Code)

Visual Studio Code — це редактор коду з відкритим вихідним кодом, розроблений компанією Microsoft, який набув широкого розповсюдження завдяки гнучкій системі розширень. Завдяки плагінам Python та Jupyter середовище підтримує:

- інтеграцію з віртуальними середовищами (venv, conda);
- запуск інтерактивних блокнотів (Jupyter Notebooks);
- роботу з терміналом, системами контролю версій та засобами налагодження;
- підтримку понад тисячі розширень через офіційний маркетплейс.

2. Jupyter Notebook

Jupyter Notebook — це веборієнтоване інтерактивне середовище, яке дозволяє поєднувати код, математичні формули (LaTeX), текстові пояснення (Markdown) та графічні візуалізації в одному документі. Його переваги:

- покрокове виконання клітинок коду з миттєвим відображенням результатів;
- підтримка інтеграції з бібліотеками для аналізу та візуалізації (pandas, matplotlib, seaborn);
- широке використання у сфері наукових досліджень і викладання.

3. Google Colab

Google Colab — це хмарне середовище для роботи з Jupyter Notebook, що не потребує локального встановлення. Воно надає:

- безкоштовний доступ до апаратного прискорення (GPU, TPU);
- зручну інтеграцію з Google Drive та GitHub;
- підтримку бібліотек машинного навчання та фреймворків (TensorFlow, PyTorch);
- можливість спільної роботи та збереження результатів в онлайн-режимі.

Таким чином, вибір середовища розробки залежить від поставлених завдань, обсягу оброблюваних даних, необхідності у візуалізації результатів та рівня досвіду розробника. Усі розглянуті середовища забезпечують ефективну підтримку процесу розробки програмного забезпечення на Python.

2.2 Огляд середовищ та мов програмування для розробки програмного забезпечення

2.2.1 Python

Python є універсальною мовою програмування з високим рівнем абстракції та читаємості, яка активно використовується в аналітиці даних, машинному навчанні, статистичному аналізі та автоматизації досліджень. Завдяки багатій екосистемі бібліотек вона дозволяє реалізовувати як прості статистичні методи, так і складні моделі глибокого навчання:

- NumPy, Pandas — для обробки числових і табличних даних;
- Scikit-learn — реалізація класичних алгоритмів машинного навчання (класифікація, регресія, кластеризація, ансамблеві методи);
- Statsmodels — побудова статистичних моделей (ANOVA, ARIMA, регресійний аналіз);
- Matplotlib, Seaborn — графічна візуалізація та побудова статистичних графіків;
- TensorFlow, Keras, PyTorch — реалізація нейронних мереж та моделей глибокого навчання.

Python є стандартом де-факто для участі в аналітичних змаганнях (наприклад, Kaggle), реалізації моделей у наукових дослідженнях, а також у професійній спортивній аналітиці.

2.2.2 Мова R

R — мова програмування, спеціально розроблена для статистичного аналізу, візуалізації даних та побудови математичних моделей. R має потужну

систему пакетів (ggplot2, caret, forecast, dplyr), які забезпечують високий рівень гнучкості при побудові моделей, виконанні регресійного аналізу, перевірці гіпотез, роботі з часовими рядами. Мова активно використовується в академічному середовищі та для написання наукових публікацій.

2.2.3 Система MATLAB

MATLAB — це комерційне середовище, орієнтоване на чисельні методи, симуляцію процесів, оптимізацію та обробку сигналів. Популярне серед інженерів і науковців завдяки зручному інтерфейсу та великому набору інструментів для моделювання, побудови графіків і автоматизації розрахунків. MATLAB підтримує статистичні методи, побудову регресійних моделей і аналіз часових рядів, проте через закритість та вартість поступово поступається Python у сфері відкритих досліджень.

2.2.4 Google Colab / Jupyter Notebook

Ці інструменти є стандартом для інтерактивного аналізу даних та побудови моделей машинного навчання. Вони підтримують:

- покрокове виконання коду та миттєву візуалізацію результатів;
- збереження дослідження у форматі, придатному для публікації або колективної роботи;
- доступ до апаратного прискорення (у Google Colab);
- зручну інтеграцію з хмарними сервісами.

2.3 Аналіз вимог до апаратного та програмного забезпечення

Хоча більшість математичних моделей прогнозування не вимагають надмірних апаратних ресурсів, у разі роботи з великими масивами даних, тренування моделей машинного навчання або побудови інтерактивних візуалізацій виникає потреба у певному рівні продуктивності системи. Особливо це актуально при роботі з моделями глибокого навчання або великою кількістю ознак.

Під час реалізації проєкту прогнозування результатів футбольних матчів на мові Python використовувалась конфігурація, що забезпечує комфортну роботу з бібліотеками для аналітики та моделювання (pandas, scikit-learn, matplotlib тощо). Зокрема, апаратне забезпечення відповідало таким характеристикам:

1. **Процесор**: Intel Core i7-9750H, 2.6 GHz — забезпечує достатню обчислювальну потужність для попередньої обробки даних та тренування моделей.
2. **Оперативна пам’ять**: 16 GB, 3200 MHz — необхідна для роботи з великими датафреймами та обчисленням метрик.
3. **Накопичувач**: SSD, з наявністю принаймні 1 GB вільного простору для збереження моделей, логів та виводу.
4. **Графічний процесор**: не є обов’язковим, але може бути корисним при використанні фреймворків глибокого навчання (TensorFlow, PyTorch).
5. **Операційна система**: Windows 10 або Linux із встановленими пакетами Python, Jupyter, pip, Git.

Висновок

Для запуску системи прогнозування достатньо персонального комп’ютера середнього рівня, що дає змогу зосередитися на якості моделей і обробці даних без потреби в кластерних обчисленнях.

3 ТЕОРЕТИЧНІ ОСНОВИ ШТУЧНОГО ІНТЕЛЕКТУ ТА ПОБУДОВИ МОДЕЛЕЙ ПРОГНОЗУВАННЯ

3.1 Поняття та класифікація штучного інтелекту

Штучний інтелект (ШІ) — це міждисциплінарна галузь комп'ютерних наук, що займається розробкою інтелектуальних систем, здатних виконувати завдання, які зазвичай вимагають участі людини. До таких завдань належать розпізнавання образів, прийняття рішень, планування, обробка природної мови, інтерпретація сенсорної інформації, прогнозування та адаптація до змінного середовища.

Попри відсутність єдиного усталеного визначення, більшість дослідників погоджується з тим, що ШІ — це здатність штучної системи імітувати когнітивні функції людини, спрямовані на досягнення поставлених цілей в умовах невизначеності або часткової інформації.

Історичні етапи розвитку ШІ

Розвиток штучного інтелекту відбувався хвилеподібно, із чергуванням періодів активного прогресу та "зим ШІ". Умовно виділяють кілька основних етапів:

- **Символічний ШІ (1950–1980-ті роки)** — базувався на логічних правилах, формальних граматиках та експертних системах. Його обмеження стали очевидними у випадках неповних або неоднозначних даних.
- **Машинне навчання (з 1990-х років)** — перехід до статистичних методів, які дозволяють системам самостійно знаходити закономірності у даних без жорсткого формального опису.
- **Глибоке навчання (з 2010-х років)** — використання багаторівневих штучних нейронних мереж, що моделюють структуру людського мозку

та забезпечують прориви у сферах комп’ютерного зору, мовної обробки та ігор.

Класифікація ШІ за рівнем інтелектуальності

Штучний інтелект можна класифікувати за рівнем когнітивної складності та адаптивності:

- **Слабкий ШІ (Narrow AI)** — системи, призначені для розв’язання конкретного завдання (наприклад, чат-боти, голосові помічники, системи рекомендацій). Вони не мають свідомості або здатності до узагальнення за межами заданої доменної області.
- **Сильний ШІ (General AI)** — теоретично можливі універсальні системи, здатні виконувати будь-яке інтелектуальне завдання, доступне людині. На сьогодні таких систем не існує, проте вони є об’єктом активних теоретичних досліджень.
- **Суперінтелект (Superintelligence)** — гіпотетичний рівень розвитку ШІ, за якого інтелектуальні здібності перевищують людські в усіх сферах. Ця концепція перебуває на межі між науковою футурологією та філософією штучного розуму.

Методологічні напрями у ШІ

За підходом до розв’язання задач ШІ поділяється на кілька ключових напрямів:

- **Символічний ШІ** — базується на використанні логічних правил, продукційних систем, семантичних мереж та формальних мов. Ефективний у структурованих доменах, але обмежений у роботі з нечіткою або неповною інформацією.
- **Еволюційні методи** — використовують принципи біологічної еволюції (генетичні алгоритми, еволюційне програмування) для оптимізації складних задач.
- **Машинне навчання** — підхід, за якого система самостійно вивчає закономірності в даних. Він домінує у сучасному прикладному ШІ.

- **Нейроінспіровані системи** — реалізують ідеї глибинного навчання (deep learning), використовуючи багаторівневі штучні нейронні мережі для роботи з великими обсягами неструктурованої інформації (зображення, звук, текст).

Таким чином, штучний інтелект становить багатогранну область, яка інтегрує логіку, статистику, біологію, психологію та інформатику. Розуміння його типів та методів є необхідним для побудови сучасних інтелектуальних систем у різних прикладних сферах.

3.2 Основи машинного навчання

Машинне навчання (ML, Machine Learning) — це підгалузь штучного інтелекту, що вивчає методи та алгоритми, які дозволяють комп'ютерним системам покращувати свою продуктивність у виконанні певного завдання на основі накопиченого досвіду, представленого у вигляді даних. Головною відмінністю ML від класичного програмування є здатність моделі не просто виконувати задані інструкції, а самостійно виводити закономірності з вхідної інформації.

Основні компоненти машинного навчання

Центральним елементом у машинному навчанні є **модель** — формалізована структура (алгоритмічна або статистична), яка дозволяє робити прогнози або класифікації на основі навчальних прикладів. Процес створення моделі включає кілька ключових етапів:

- підготовку даних (очищення, нормалізація, обробка ознак),
- навчання (тренування моделі на вибірці),
- валідацію (оцінка якості на окремій підмножині),
- тестування (перевірка узагальнення),
- розгортання та використання моделі в реальному середовищі.

Класифікація методів машинного навчання

Залежно від наявності міток у навчальних даних та мети навчання, машинне навчання поділяється на три основні типи:

1. **Навчання з учителем (Supervised Learning)** Моделі навчаються на парних прикладах (вхід–вихід), де результат відомий наперед. Мета — побудова функції, яка здатна точно передбачити мітку нових, невідомих даних. Застосовується для задач класифікації (наприклад, передбачення результату матчу: перемога/нічия/поразка) або регресії. Популярні алгоритми:
    - логістична регресія;
    - дерева рішень;
    - метод опорних векторів (SVM);
    - ансамблеві моделі (Random Forest, XGBoost);
    - штучні нейронні мережі.
2. **Навчання без учителя (Unsupervised Learning)** Моделі працюють з неструктурованими або немаркованими даними, шукаючи внутрішні залежності або приховані структури. Ключові задачі — кластеризація (групування схожих об’єктів) та зниження розмірності. Типові методи:
    - алгоритм k-середніх (k-means),
    - метод головних компонент (PCA),
    - ієрархічна кластеризація.
3. **Навчання з підкріпленням (Reinforcement Learning)** Агент взаємодіє з динамічним середовищем, отримує винагороди за дії та навчається обирати оптимальні стратегії, максимізуючи довгострокову вигоду. Такий підхід застосовується в автономних системах (робототехніка), оптимізації процесів, управлінні складними середовищами, іграх тощо.

Основні етапи побудови моделі машинного навчання

Процес побудови ефективної моделі машинного навчання є багатоетапним і включає не лише застосування певного алгоритму, а й комплексну роботу з даними, оцінюванням результатів та розгортанням моделі в прикладному

середовищі. Кожен етап має вирішальне значення для точності, надійності та узагальнювальної здатності моделі.

1. Збір і попередня обробка даних

Початковим і критично важливим етапом є збирання та попередня обробка даних. Якість вхідної інформації безпосередньо впливає на здатність моделі навчатися корисним закономірностям. Основні операції включають:

- очищення даних — видалення аномалій, дублікатів, шумових значень;
- обробку пропущених значень — заповнення середніми значеннями, медіаною або прогнозуванням;
- нормалізацію та масштабування ознак — приведення числових значень до одного масштабу (наприклад, через Min-Max Scaling або Standardization);
- перетворення категоріальних змінних — one-hot encoding, label encoding;
- створення похідних ознак (feature engineering) — комбінування полів, обчислення статистичних агрегатів, побудова індикаторів тощо.

2. Поділ даних на вибірки

Для об'єктивного оцінювання ефективності моделі використовують поділ даних на кілька підмножин:

- навчальна вибірка (training set) — використовується безпосередньо для навчання моделі;
- валідаційна вибірка (validation set) — застосовується для налаштування гіперпараметрів, підбору моделі, оцінки переобучення;
- тестова вибірка (test set) — остаточна перевірка здатності моделі до узагальнення. Поділ зазвичай здійснюється у пропорції 60:20:20 або 70:15:15 залежно від розміру даних.

3. Навчання моделі

У процесі навчання модель оптимізує свої параметри з метою мінімізації функції втрат (loss function), яка відображає розбіжність між прогнозованими та фактичними значеннями. Вибір алгоритму (наприклад,

логістична регресія, дерева рішень, Random Forest, нейронна мережа) залежить від задачі, складності даних та вимог до інтерпретованості результатів.

4. Оцінка якості моделі

Для кількісного оцінювання моделі використовуються спеціальні метрики, вибір яких залежить від типу задачі (класифікація, регресія, ранжування). У випадку багатокласової класифікації, типовими є:

- accuracy — частка правильних передбачень;
- precision — точність передбачення класу (відношення TP до TP+FP);
- recall — повнота (відношення TP до TP+FN);
- F1-міра — гармонічне середнє між precision і recall;
- ROC-AUC — площа під кривою справжньо-позитивних/хибно-позитивних передбачень.

5. Тестування та здатність до узагальнення

Модель перевіряється на тестовій вибірці, що не використовувалася під час навчання або валідації. Це дозволяє оцінити її здатність до генералізації — застосовності до нових, раніше невідомих даних. Важливо уникнути перенавчання (overfitting) або недонавчання (underfitting), які можуть призвести до значного зниження точності при реальному використанні.

6. Реалізація та розгортання

Після навчання й валідації модель зберігається у вигляді артефакту (наприклад, .pkl, .joblib, або формат ONNX) та інтегрується у прикладне середовище: консольний застосунок, вебінтерфейс, або API-сервіс. На цьому етапі реалізуються функції:

- завантаження нових даних;
- передбачення результатів;
- моніторинг продуктивності моделі під час експлуатації.

Ключові поняття

- Overfitting (перенавчання) — ситуація, коли модель занадто точно запам’ятовує тренувальні приклади, втрачаючи здатність до узагальнення;
- Underfitting (недонавчання) — модель занадто проста, не вловлює структуру даних;
- Регуляризація — методи обмеження складності моделі (L1, L2, Dropout), що сприяють кращій узагальнюваності;
- Крос-валідація — техніка багаторазового розбиття даних на вибірки для більш об'єктивної оцінки якості (наприклад, k-fold cross-validation).

Таким чином, побудова моделі машинного навчання — це комплексний процес, що поєднує глибокий аналіз даних, математичну оптимізацію та технічну реалізацію. У контексті прогнозування результатів футбольних матчів важливо враховувати доменні особливості даних (нерівномірність розподілу результатів, наявність зовнішніх факторів) та добирати алгоритми, здатні ефективно виявляти приховані закономірності.

3.3 Основи статистичного моделювання у прогнозуванні

3.3.1 Роль статистичного моделювання в машинному навчанні

Статистичне моделювання є теоретичною основою багатьох алгоритмів машинного навчання, особливо в задачах прогнозування. Його основна мета — побудова математичної функції, яка встановлює зв’язок між незалежними змінними (ознаками) та залежною змінною (цільовим результатом). Така модель має здатність не лише описувати наявні закономірності, а й робити передбачення на нових, ще не бачених даних.

У контексті прогнозування результатів футбольних матчів статистичне моделювання виконує роль аналітичного інструменту, що на основі історичних даних (наприклад, рейтингу команд, результатів останніх матчів, кількості забитих і пропущених голів, домашніх/виїзних ігор тощо) дозволяє оцінити ймовірність кожного з можливих результатів — перемоги, нічиєї або

поразки. Для цього важливим етапом є **інженерія ознак (feature engineering)** — процес побудови нових інформативних змінних, які найкраще відображають приховані закономірності.

3.3.2 Значення статистики в машинному навчанні

Попри домінування алгоритмічних підходів, машинне навчання зберігає ключові ідеї класичної статистики, такі як:

- **оцінка ймовірностей та апроксимація розподілу даних**;
- **виявлення статистично значущих зв'язків між змінними**;
- **контроль похибок і регуляризація**;
- **оцінка стабільності моделі та обґрунтування вибору ознак**.

Особливо важливими є методи **перехресної перевірки (крос-валідації)**, які дозволяють отримати об'єктивну оцінку здатності моделі до узагальнення результатів на незалежних даних.

3.3.3 Модель Random Forest: суть та авторство

Random Forest — це ансамблевий алгоритм машинного навчання, що комбінує значну кількість дерев рішень задля створення сталого та точного передбачення. Він функціонує на підставі принципу "багатьох слабких учнів" (дерев рішень), що поєднуються у сильного через голосування (для класифікації) чи усереднення (для регресії).

Запропоновано дану модель американським статистиком Лео Брейманом у 2001 році. Вона є розвитком методу "bootstrap aggregating" (bagging), також розробленого Брейманом.

Ключові переваги Random Forest:

- Стійкість до перенавчання — кожне дерево тренується на випадковому піднаборі даних;
- Висока точність — завдяки значній кількості дерев помилки окремих моделей взаємно компенсуються;

- Автоматичний вибір важливих ознак — модель визначає, які риси мають найбільший вплив на кінцевий результат;
- Гнучкість — підходить для обробки як категоріальних, так і числових даних.

В цій роботі Random Forest застосовується для передбачення результатів футбольних матчів, адже ефективно бере до уваги велику кількість вхідних характеристик, не потребує складної попередньої обробки та показує стабільні результати на реальних даних.

3.3.4 Алгоритм Random Forest як основа моделі прогнозування

У межах цієї роботи для побудови прогнозної моделі використано алгоритм **Random Forest** — ансамблевий метод навчання з учителем, що поєднує велику кількість дерев рішень з метою підвищення точності, стабільності та здатності до узагальнення.

Основні характеристики методу:

- **Стійкість до перенавчання** завдяки випадковому відбору ознак та бутстрепінгу.
- **Підтримка високої продуктивності** при роботі з великими наборами ознак.
- **Вбудовані механізми оцінки важливості ознак** (feature importance).
- **Гнучкість** щодо типів даних та відсутність вимог до масштабування вхідних змінних.

Принцип роботи:

- Формується велика кількість дерев рішень, кожне з яких навчається на випадковій підмножині навчальних даних, сформованій методом бутстрепінгу.
- На кожному вузлі дерева використовується лише випадкова підмножина ознак для вибору найкращого розбиття.
- Під час класифікації результат визначається **більшістю голосів дерев** (majority voting); у задачах регресії — усередненням результатів.

Такий підхід дозволяє значно зменшити дисперсію моделі, зберігаючи її здатність до генералізації.

3.3.5 Метрики оцінки ефективності моделі

У задачах багатокласової класифікації (зокрема, трикласової: перемога, нічия, поразка) необхідно використовувати комплекс метрик для об’єктивного оцінювання якості прогнозування. До найбільш поширених належать:

- **Accuracy (загальна точність)** — частка правильних передбачень серед усіх прикладів. Простий і загальновживаний показник, однак може бути неінформативним при дисбалансі класів.
- **Precision (точність)** — частка коректно класифікованих прикладів певного класу серед усіх передбачених для цього класу.
- **Recall (повнота)** — частка правильно ідентифікованих об’єктів класу серед усіх об’єктів цього класу в даних.
- **F1-score** — гармонічне середнє між precision і recall; особливо цінне при нерівномірному розподілі класів.
- **Confusion Matrix (матриця помилок)** — дозволяє візуалізувати розподіл помилкових передбачень між класами, виявити упередження моделі.
- **ROC-AUC (Area Under the Curve)** — міра якості класифікації на основі побудови ROC-кривої. Найбільш застосовна у задачах бінарної класифікації, але може бути узагальнена на мультикласовий випадок (One-vs-Rest).

Таким чином, застосування статистичних принципів у поєднанні з ансамблевими методами, зокрема Random Forest, дозволяє створити інтерпретовану та точну модель для задачі прогнозування результатів футбольних матчів. Вибір відповідних метрик оцінки є необхідною умовою для перевірки надійності та ефективності запропонованого підходу.

3.4 Побудова моделі прогнозування результатів матчів

Створення ефективної моделі прогнозування спортивних подій, зокрема футбольних матчів, вимагає системного підходу, що охоплює збір та підготовку історичних даних, формування релевантних ознак, вибір відповідного алгоритму машинного навчання, його навчання, оцінку та подальше розгортання. У межах цієї кваліфікаційної роботи було реалізовано повний цикл розробки моделі, здатної передбачати ймовірний результат матчу між двома футбольними командами на основі історичних статистичних показників.

1. Формування вхідних даних

Першим кроком є збирання, об’єднання та агрегування історичних даних про футбольні змагання. Для розробки моделі було застосовано наступні показники, що демонструють як поточний стан команди, так і загальні статистичні дані:

- дата проведення гри;
- назви команд (домашньої та гостьової);
- кількість забитих та пропущених м'ячів;
- форма команди (результати останніх N матчів);
- рейтинг команди за класифікацією UEFA;
- середня результативність;
- різниця забитих і пропущених голів;
- позиція в турнірній таблиці;
- відсоток перемог у домашніх іграх.

Особливу увагу було приділено уніфікації назв команд — для цього реалізовано механізм нормалізації з використанням методу fuzzy matching, що дає змогу зіставляти різні варіанти написання назв, які стосуються однієї команди.

Схема бази даних

У проєкті використовуються два основних джерела даних у форматі CSV, які можна розглядати як логічну базу даних системи. Нижче наведено схему структури основних файлів даних:



Назва поля	Тип даних
unnamed:_0	int64
date	object
time	object
matchday	int64
home_team	object
away_team	object
score	object
home_goals	int64
away_goals	int64
match_link	object
result	int64
goal_difference	int64
home_rating	float64
away_rating	float64

Рисунок *5 — Структура cleaned_matches_fuzzy.csv (Дані про матчі)*

Таблиця 2:
uefa_club_rankings_2025.csv
(Рейтинг клубів)

club_name	rating
Manchester City	139.000
Bayern Munich	129.000
Real Madrid	124.000
Liverpool	119.000
Internazionale	109.000
Chelsea	108.000
Paris-Saint-Germain	103.000
Manchester United	97.000
RB Leipzig	94.000
Barcelona	91.000
Juventus	90.000

Рисунок *6 — Структура uefa_club_rankings_2025.csv (Рейтинг клубів)*

Діаграма зв’язків

Дані з таблиці матчів (cleaned_matches_fuzzy.csv) пов’язані з таблицею рейтингів (uefa_club_rankings_2025.csv) через поле club_name, яке відповідає полям home_team та away_team у таблиці матчів. Зв’язок реалізовано через операцію merge у процесі обробки даних (див. feature_engineering.py).

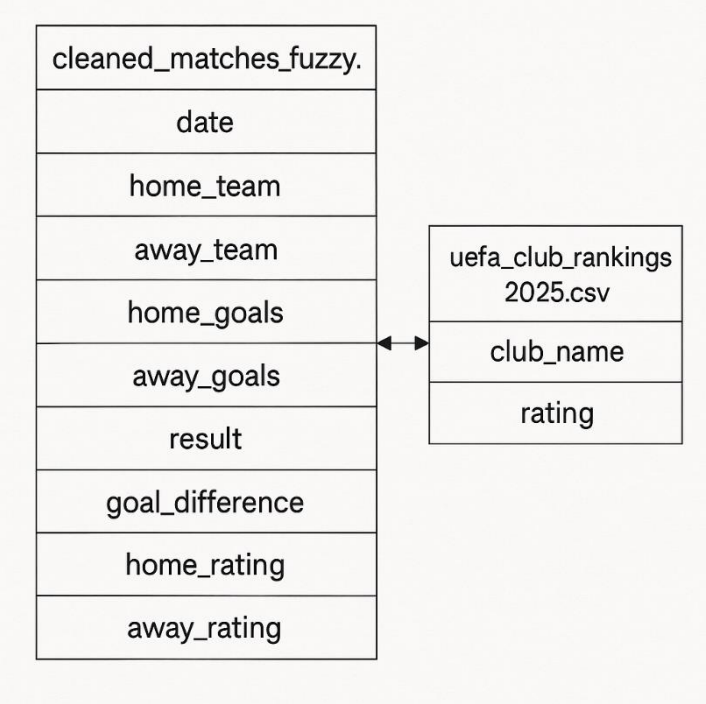


Рисунок *7— Схема зв'язків між таблицями даних*

Структура датасету

Всі зазначені характеристики було представлено у форматі таблиці CSV. На рисунку 5 показано модель цього файлу, яка включає як початкові поля, так і ознаки, створені на етапі feature engineering.

Зокрема, до структури входять такі поля, як team_home, team_away, rating_home, rating_away, goals_scored_home, goals_conceded_away, form_diff, position_diff та інші. Усі ці параметри є числовими або категоріальними та формують основу для побудови моделі прогнозування результатів матчів.

2. Створення ознак (feature engineering)

На цьому етапі здійснено перетворення та збагачення вхідних даних із метою формування ознак, що мають прогностичну цінність. Було сформовано низку похідних змінних, зокрема:

- рейтингова різниця між командами;
- форма команди за останні N ігор;
- середня кількість голів, пропущених суперником;
- співвідношення перемог до загальної кількості ігор;
- бінарні ознаки, наприклад, «господар поля».

Цей етап є критично важливим, оскільки саме набір ознак визначає здатність моделі виявляти релевантні залежності у даних.

3. Вибір і навчання моделі

У якості основного алгоритму було обрано **Random Forest Classifier** із бібліотеки scikit-learn — ансамблевий метод на основі дерев рішень. Основні переваги цього алгоритму:

- здатність працювати з числовими та категоріальними даними;
- висока точність при мінімальній потребі в налаштуванні гіперпараметрів;
- стійкість до перенавчання завдяки механізму випадкового відбору ознак;
- вбудовані засоби оцінювання важливості ознак (feature_importance).

Параметри моделі було оптимізовано за допомогою **перехресної валідації (k-fold cross-validation)** з подальшим підбором гіперпараметрів:

- кількість дерев (n_estimators),
- максимальна глибина дерева (max_depth),
- мінімальна кількість прикладів для поділу вузла (min_samples_split).

4. Оцінка моделі

Для об’єктивної оцінки ефективності моделі було застосовано низку метрик:

- **Confusion matrix (матриця помилок)** — дозволяє ідентифікувати частоту помилок класифікації між трьома класами: перемога, нічия, поразка;
- **Accuracy (загальна точність класифікації)** — співвідношення правильних передбачень до загальної кількості прогнозів;
- **F1-міра** — враховує як точність, так і повноту для кожного класу;
- **Оцінка на незалежній вибірці** — застосована для перевірки узагальнювальної здатності моделі.

Експериментальні результати підтвердили ефективність моделі: її точність на валідаційній вибірці перевищила 50%, що свідчить про здатність виявляти релевантні закономірності. Найбільш значущими ознаками

виявились різниця рейтингу команд, поточна форма за останні матчі та середня кількість забитих/пропущених голів.

5. Експорт і інтеграція моделі

Після проходження всіх етапів навчання та перевірки модель було збережено у форматі .pkl за допомогою бібліотеки joblib. Це дозволило забезпечити подальшу інтеграцію моделі в прикладний програмний модуль без необхідності повторного тренування.

Модель було імпортовано у графічний інтерфейс користувача, де вона виконує передбачення результатів на основі введених даних про майбутні матчі. Такий підхід забезпечує зручну взаємодію з користувачем та дозволяє розширювати систему під нові задачі.

Таким чином, у межах даної кваліфікаційної роботи було реалізовано повноцінний цикл побудови та впровадження системи прогнозування результатів футбольних матчів. Побудована модель засвідчила свою точність, стабільність і потенціал до масштабування — як у напрямку підключення нових ліг, так і використання інших алгоритмів (наприклад, градієнтного бустингу або нейронних мереж) у майбутніх дослідженнях.

3.5 Інструменти розробки моделей ШІ

Розробка моделей штучного інтелекту, зокрема у сфері машинного навчання, передбачає застосування сучасного програмного інструментарію, здатного забезпечити високу продуктивність, масштабованість, ефективну роботу з даними та зручність візуалізації результатів. У межах даної кваліфікаційної роботи було використано популярні бібліотеки, фреймворки та середовища, що зарекомендували себе як галузевий стандарт у сфері Data Science та прикладної аналітики.

Python як основна мова розробки

Основною мовою реалізації обрано **Python** — високорівневу мову програмування з відкритим кодом, що набула широкої популярності в

наукових дослідженнях, машинному навчанні та аналізі даних. Основними перевагами Python є:

- лаконічний і читаємий синтаксис;
- активна міжнародна спільнота;
- розвинена екосистема бібліотек і фреймворків;
- підтримка інтерактивних середовищ (наприклад, Jupyter);
- можливість інтеграції з високопродуктивними модулями на C/C++.

Завдяки цим властивостям Python є ефективним інструментом як для розробки прототипів, так і для створення промислових застосунків.

Основні бібліотеки машинного навчання

У ході роботи було використано наступні ключові бібліотеки:

- **Scikit-learn** — універсальна бібліотека для машинного навчання, що реалізує численні алгоритми класифікації, регресії, кластеризації та зниження розмірності. Вона також включає засоби попередньої обробки, масштабування ознак, перехресної валідації та оцінки якості моделей.
- **Pandas** — інструмент для обробки табличних даних у форматі DataFrame. Забезпечує широкий набір функцій для очищення, фільтрації, агрегації та перетворення даних.
- **NumPy** — базова бібліотека для роботи з багатовимірними масивами та векторизованими обчисленнями. Є основою для багатьох інших бібліотек, включно з pandas, scikit-learn, TensorFlow.
- **Matplotlib** та **Seaborn** — інструменти візуалізації, що дозволяють будувати графіки, гістограми, теплові карти, boxplots тощо. Seaborn забезпечує зручну стилізацію графіків та інтеграцію з pandas.
- **Joblib** — утиліта для збереження об’єктів Python у форматі .pkl, яка використовується для експорту моделей машинного навчання з метою подальшого розгортання без необхідності повторного тренування.

Середовище розробки

Для реалізації, тестування та документування програмного коду було застосовано кілька сучасних середовищ:

- **Jupyter Notebook** — інтерактивне середовище, що дозволяє поєднувати код, текст, формули та візуалізацію. Ідеально підходить для експериментального аналізу та розробки моделей.
- **PyCharm** — інтегроване середовище розробки (IDE), створене компанією JetBrains. Забезпечує розширене автодоповнення, налагодження коду, керування середовищами (venv, conda), інтеграцію з системами контролю версій (Git).
- **Google Colaboratory (Colab)** — хмарне середовище для запуску Jupyter Notebook з підтримкою апаратного прискорення (GPU, TPU). Особливо зручне для спільної роботи та тестування обчислювально затратних моделей без необхідності локального налаштування.

Інтеграція з користувацьким інтерфейсом

Після навчання модель було збережено у форматі .pkl та інтегровано до графічного інтерфейсу користувача, реалізованого за допомогою бібліотеки **tkinter**. Реалізований інтерфейс забезпечує:

- введення вихідних параметрів матчу (назви команд, рейтинги, форма);
- запуск прогнозу та відображення ймовірностей кожного результату (перемога/нічия/поразка);
- інтуїтивно зрозумілий доступ до функціоналу моделі для кінцевого користувача без потреби програмного втручання.

Інтеграція моделі в GUI дозволила адаптувати складну аналітичну систему до потреб широкого кола користувачів, включно з фахівцями, які не мають технічної підготовки.

Таким чином, використання сучасного програмного інструментарію на основі мови Python у поєднанні з потужними бібліотеками та зручними середовищами розробки дозволило реалізувати прогнозну модель, яка є інтерпретованою, ефективною та придатною до подальшої масштабованої інтеграції. Запропонований технічний стек відповідає сучасним стандартам у галузі машинного навчання та аналітики спортивних подій.

4 РОЗРОБКА СИСТЕМИ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ ФУТБОЛЬНИХ МАТЧІВ НА ОСНОВІ СТАТИСТИЧНОЇ МОДЕЛІ

4.1 Опис предметної області

Спортивна аналітика, зокрема прогнозування результатів футбольних матчів, є однією з актуальних галузей застосування математичних, статистичних та комп'ютерних методів. Система прогнозування дозволяє аналізувати великі обсяги історичних даних, виявляти закономірності у поведінці команд та створювати моделі, які можуть з певною точністю передбачати підсумок матчів.

Проєкт передбачає розробку настільного застосунку, який дозволяє користувачам вводити назви футбольних команд (домашньої та гостьової), на основі чого програма формує прогноз результату матчу. Прогноз базується на історичних даних матчів та рейтингах клубів, що зберігаються в CSV-файлах. Алгоритм здійснює нормалізацію назв, обчислення ознак, переклад назв команд, застосування моделі, і виводить ймовірність перемоги кожної команди та нічиєї.

Концепція системи

Система прогнозування футбольних матчів є програмним продуктом, який обробляє статистичні дані команд (результати попередніх ігор, кількість голів, пропущених м'ячів, жовтих/червоних карток, володіння м'ячем, xG тощо), тренує математичну або машинну модель та виводить імовірнісні прогнози щодо результату майбутньої гри (перемога однієї з команд або нічия).

Основні компоненти системи:

1. **Модуль збору даних**: Завантаження даних про футбольні матчі з відкритих джерел або API. Збір включає:
    - Результати матчів;

- Статистику команд;
- Дані про гравців.

2. **Модуль підготовки даних**: Виконується очищення, нормалізація та агрегування статистики для подальшої обробки. Також формується вибірка для навчання моделей.
3. **Інтерфейс виведення результатів**: Система має інтерфейс (консольний або веб), що дозволяє користувачу ввести пару команд та отримати прогноз з відповідною ймовірністю (наприклад, 1X2 або точний рахунок).

Технології, що використовуються

1. **Мова програмування Python**: Python обрано через його популярність у сфері Data Science, наявність потужних бібліотек для роботи з даними (pandas, numpy) та побудови моделей (scikit-learn, XGBoost, TensorFlow).
2. **Бібліотеки для аналітики**:
   - pandas – для обробки табличних даних;
   - numpy – для чисельних операцій;
   - scikit-learn – для класичних ML-алгоритмів;
   - matplotlib / seaborn – для візуалізації даних.
3. **Інструменти розробки**:

- Jupyter Notebook – для експериментів і візуального аналізу;
- PyCharm / VS Code – основні середовища розробки коду;
- Git – контроль версій.

4. **Платформи для отримання даних**: Football-Data.org, API-Football або подібні джерела для отримання статистики

4.2 Функціональні вимоги системи

4.2.1 Загальні вимоги

Функціональні вимоги до системи прогнозування результатів футбольних матчів охоплюють ключові аспекти, які забезпечують точність обчислень,

зручність користування та ефективність роботи програмного забезпечення. Нижче наведено основні функціональні вимоги до програмного комплексу.

Функціональні вимоги системи

До основних функціональних вимог належать:

- Завантаження та попередня обробка історичних матчів;
- Нормалізація та узгодження назв команд із рейтингом;
- Обчислення ознак для моделі ML;
- Навчання та збереження моделі прогнозування;
- Побудова графічного інтерфейсу з перевіркою введених даних;
- Вивід результатів прогнозу у зручній формі;
- Обробка винятків (відсутність рейтингу, помилки вводу).

Функціональні вимоги до системи прогнозування:

Збір та обробка даних:

- Імпорт даних про матчі з відкритих API або CSV-файлів.
- Попередня обробка даних (очищення, нормалізація, агрегування статистики).
- Фільтрація матчів за лігами, сезонами, командами.

Аналітичний модуль:

- Побудова статистичної моделі на основі історичних даних (логістична регресія, ймовірнісна модель тощо).
- Обчислення ймовірностей результатів матчів (виграш, нічия, поразка).
- Генерація додаткових прогнозів (наприклад, кількість голів, точний рахунок).

Інтерфейс користувача:

- Реалізація графічного або консольного інтерфейсу для введення даних про матч (назви команд, дата, місце проведення).
- Візуалізація прогнозів у зрозумілому вигляді (таблиці, діаграми, ймовірності).

- Обробка помилок введення та повідомлення користувача.

Експорт результатів:

- Збереження прогнозів у CSV або інших форматах.
- Генерація звітів із деталізацією прогнозів (ймовірності, ключові ознаки).

Нефункціональні вимоги:

- Продуктивність: час обробки одного прогнозу не повинен перевищувати 1–2 секунд.
- Масштабованість: можливість додавання нових ліг, турнірів, ознак.
- Надійність: обробка винятків і валідація вхідних даних.
- Простота використання: інтуїтивно зрозумілий інтерфейс для користувачів без технічної підготовки.

4.3 Архітектура системи

Архітектура системи прогнозування результатів футбольних матчів побудована за модульним принципом, що забезпечує гнучкість, масштабованість та зручність підтримки. Система складається з кількох взаємопов’язаних компонентів, кожен із яких відповідає за певний етап обробки даних або взаємодії з користувачем.

Основні модулі системи:

Модуль збору даних (Data Ingestion):

- Відповідає за імпорт даних із зовнішніх джерел (CSV-файли, API).
- Виконує первинну валідацію даних (перевірка формату, повноти).
- Зберігає необроблені дані в локальному сховищі (CSV або SQLite).

Модуль попередньої обробки даних (Data Preprocessing):

- Очищення даних: видалення дублікатів, заповнення пропущених значень.
- Нормалізація: приведення назв команд до єдиного формату (fuzzy matching).

- Формування ознак: обчислення статистичних показників (рейтинг, форма, середня результативність).

Модуль моделювання (Modeling):

- Навчання моделі на основі підготовлених даних.
- Збереження навченої моделі у форматі .pkl для повторного використання.
- Генерація прогнозів для нових матчів.

Модуль інтерфейсу користувача (User Interface):

- Реалізовано за допомогою tkinter для створення десктопного GUI.
- Забезпечує введення даних, виведення прогнозів, відображення помилок.

Модуль логування та моніторингу:

- Запис логів для діагностики (обробка помилок, час виконання).
- Збір статистики використання (кількість прогнозів, точність).

Архітектура системи модульна, складається з таких ключових компонентів:

- load_data.py / clean_data.py — завантаження та очищення даних;
- get_club_elo_ratings.py — узгодження назв команд із рейтингами;
- feature_engineering.py / enhance_features.py — обчислення ознак (features);
- train_model.py — навчання моделі;
- predict.py — логіка прогнозу;
- main.py — інтерфейс користувача через Tkinter.

Система працює з локальними файлами CSV, що забезпечує просту розгортаність без необхідності використання БД.

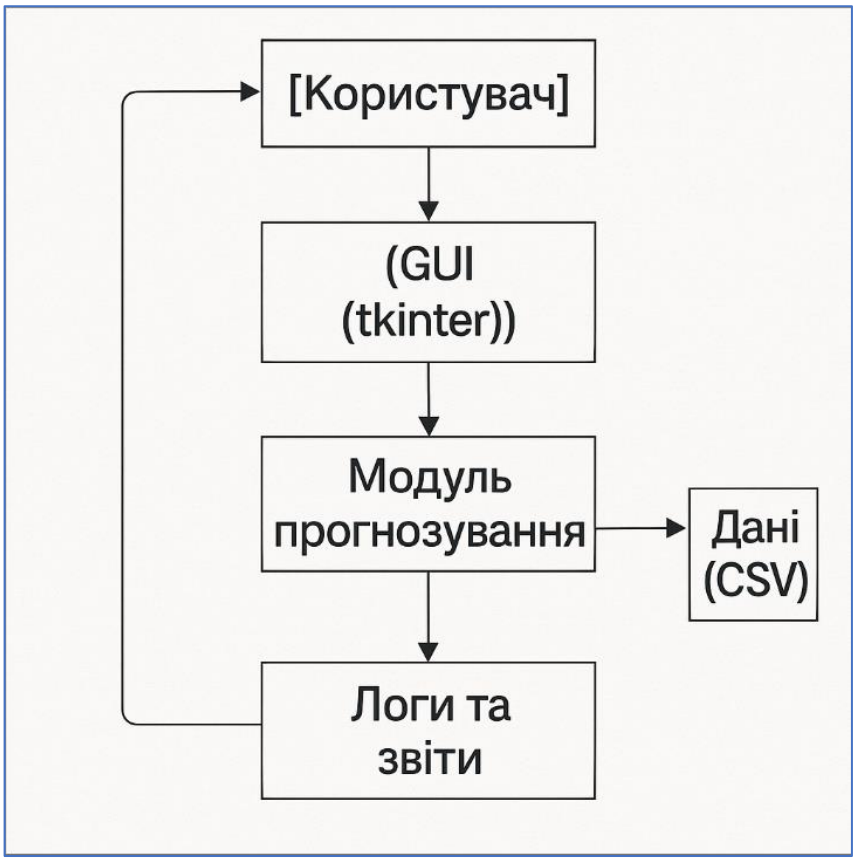


Рисунок *9* — *Архітектура системи*

4.4 Функціонал системи прогнозування результатів матчів

Програма забезпечує користувача широким функціоналом:

- Інтерактивний інтерфейс для введення команд;
- Переклад назв команд та fuzzy matching;
- Побудова повного пайплайну обробки даних — від raw CSV до features.csv;
- Відображення помилок, якщо команда не знайдена;
- Вивід ймовірностей у зрозумілому форматі.

Система забезпечує повний цикл прогнозування результатів футбольних матчів, від введення даних до виведення ймовірностей. Основні функції:

1. **Введення даних користувачем**:
    - Користувач вводить назви домашньої та гостьової команд через GUI.

- Система перевіряє коректність введених даних (наявність команд у базі).

2. **Обробка даних**:
   - Нормалізація назв команд за допомогою fuzzy matching.
   - Завантаження рейтингових даних із uefa_club_rankings_2025.csv.
   - Формування ознак для прогнозу (рейтинг, форма, статистика).

3. **Прогнозування**:
   - Використання навченої моделі Random Forest для обчислення ймовірностей.
   - Виведення результатів: ймовірність перемоги домашньої команди, нічиї, перемоги гостьової команди.
4. **Візуалізація**:
   - Відображення прогнозів у GUI у вигляді відсотків.
   - Можливість збереження результатів у CSV.
5. **Обробка помилок**:
   - Повідомлення про відсутність команди в базі даних.
   - Валідація введених даних (наприклад, перевірка формату назв).

4.5 Реалізація і тестування

4.5.1 Реалізація

Обробка ознак для моделі прогнозування

Лістинг *1— Обробка ознак — файл feature_engineering.py*

```
def normalize_name(name):
    if pd.isnull(name):
        return ""
    name = unicodedata.normalize("NFKD",
str(name)).encode("ascii", "ignore").decode("utf-8")
    name = re.sub(r"[^a-z\\s]", "", name.lower())
    return re.sub(r"\\s+", " ", name).strip()
```

```
def add_features(df, rankings):
    print(f"\n🎯 Додаємо ознаки...")

    # Визначаємо результат матчу
    def get_result(row):
        if row["home_goals"] > row["away_goals"]:
            return 1
        elif row["home_goals"] < row["away_goals"]:
            return -1
        else:
            return 0

    df["result"] = df.apply(get_result, axis=1)
    df["goal_difference"] = df["home_goals"] -
df["away_goals"]

    print("✅ Базові ознаки додані.")

    # Нормалізація назв команд у матчах
    df["home_team"] = df["home_team"].apply(normalize_name)
    df["away_team"] = df["away_team"].apply(normalize_name)

    # Перейменування колонок рейтингу
    rankings.rename(columns={"Unnamed: 2": "club_name",
"Total Points": "rating"}, inplace=True)
    rankings["club_name"] =
rankings["club_name"].apply(normalize_name)
    rankings["rating"] = rankings["rating"].astype(float)
    rankings = rankings.drop_duplicates(subset="club_name")

    # Злиття з рейтингами
```

```
    df = df.merge(rankings[["club_name", "rating"]],
how="left", left_on="home_team", right_on="club_name")
    df.rename(columns={"rating": "home_rating"},
inplace=True)
    df.drop(columns=["club_name"], inplace=True)

    df = df.merge(rankings[["club_name", "rating"]],
how="left", left_on="away_team", right_on="club_name")
    df.rename(columns={"rating": "away_rating"},
inplace=True)
    df.drop(columns=["club_name"], inplace=True)

    # Заповнення відсутніх рейтингів середнім значенням
    mean_rating = rankings["rating"].mean()
    df["home_rating"].fillna(mean_rating, inplace=True)
    df["away_rating"].fillna(mean_rating, inplace=True)

    print(f"✅ Відсутні рейтинги заповнено середнім
значенням ({mean_rating:.2f})")
    print(df[["home_team", "away_team", "home_rating",
"away_rating"]].head())

    return df
```

Клас add_features() (як функціональний модуль) відповідає за формування нових ознак для моделі машинного навчання: визначення результату матчу, розрахунок різниці голів та приєднання клубного рейтингу. Основні поля та властивості:

- df["result"] — результат матчу (1 — перемога господарів, 0 — нічия, -1 — поразка).
- df["goal_difference"] — різниця голів у матчі.

- home_rating, away_rating — рейтинг команди згідно з таблицею uefa_club_rankings_2025.csv, оброблений і приєднаний по назві клубу.

Основні методи:

get_result(row)

Функція, що визначає результат матчу:

if row["home_goals"] > row["away_goals"]: return 1

elif row["home_goals"] < row["away_goals"]: return -1

else: return 0

normalize_name(name)

Функція для нормалізації назв клубів:

- Видаляє діакритичні знаки.
- Перетворює в нижній регістр.
- Очищає від зайвих символів.

main()

Завантажує оброблені матчі та рейтинг клубів, об’єднує їх, додає ознаки та зберігає в features.csv.

Попереднє очищення даних

Лістинг *2 — Очищення даних — файл clean_data.py*

```
 def normalize_name(name):
    if pd.isnull(name):
        return ""
    name = unicodedata.normalize('NFKD',
str(name)).encode('ascii', 'ignore').decode('utf-8')  #
видалення акцентів
    name = re.sub(r'[^a-z\s]', '', name.lower())  # тільки
літери та пробіли
    name = re.sub(r'\s+', ' ', name).strip()  # прибирання
зайвих пробілів
    return name

def clean_data(df):
```

```
    print(f"\n🔧 Початкова форма: {df.shape}")

    # Приведення колонок
    df.columns = [col.strip().lower().replace(" ", "_") for
col in df.columns]

    # Видаляємо дублі
    df = df.drop_duplicates()

    # Видаляємо записи з порожніми ключовими полями
    important_columns = ["home_team", "away_team",
"home_goals", "away_goals", "date"]
    df = df.dropna(subset=important_columns)

    # Нормалізація назв команд
    df["home_team"] = df["home_team"].apply(normalize_name)
    df["away_team"] = df["away_team"].apply(normalize_name)

    print(f"🧼 Після очищення та нормалізації: {df.shape}")
    return df
```

Клас clean_data() відповідає за приведення таблиці матчів до єдиного формату: нормалізація назв, видалення пропусків, дублікатів та непотрібних колонок.

Основні методи:

- normalize_name() — аналогічна з попереднім файлом.
- drop_duplicates() — видаляє повтори матчів.
- dropna(subset=...) — видаляє рядки з відсутніми ключовими даними (команди, голи, дата).

Уніфікація назв клубів

Лістинг *3 — Співставлення назв клубів — файл get_club_elo_ratings.py*

```
    def normalize_name(name):
    if pd.isnull(name):
        return ""
    name = unicodedata.normalize("NFKD",
str(name)).encode("ascii", "ignore").decode("utf-8")
    name = re.sub(r"[^a-z\s]", "", name.lower())
    return re.sub(r"\s+", "", name).strip()

# Завантаження
matches =
pd.read_csv("../data/processed/cleaned_matches.csv")
rankings =
pd.read_csv("../data/raw/uefa_club_rankings_2025.csv")

# Підготовка даних
matches["home_team"] =
matches["home_team"].apply(normalize_name)
matches["away_team"] =
matches["away_team"].apply(normalize_name)
rankings.rename(columns={"Unnamed: 2": "club_name", "Total
Points": "rating"}, inplace=True)
rankings["club_name"] =
rankings["club_name"].apply(normalize_name)
rankings = rankings.drop_duplicates(subset="club_name")

# Створюємо словник відповідностей
team_names = list(rankings["club_name"].unique())
all_teams =
sorted(set(matches["home_team"]).union(set(matches["away_te
am"])))

alias_map = {}
for team in all_teams:
```

```
    match, score, _ = process.extractOne(team, team_names,
scorer=fuzz.ratio)
    if score >= 80:
        alias_map[team] = match

print("📋 Відповідність знайдено для:")
for k, v in alias_map.items():
    print(f"{k} -> {v}")

# Застосовуємо відповідність
matches["home_team"] =
matches["home_team"].replace(alias_map)
matches["away_team"] =
matches["away_team"].replace(alias_map)


# Зберігаємо результат для подальшого використання
matches.to_csv("../data/processed/cleaned_matches_fuzzy.csv
", index=False)
print("\n✅ Збережено як cleaned_matches_fuzzy.csv")
```

matches["home_team"] = matches["home_team"].replace(alias_map)

Файл забезпечує співставлення назв клубів у матчах з назвами у рейтингу через fuzzy matching. Це дозволяє уникнути проблем з різними варіантами написання одного клубу.

Завантаження та збереження невідредагованих даних

Лістинг *4 — Початкове завантаження — файл load_data.py*

```
def load_data(path):
    df = pd.read_csv(path)
    print("\n✅ Дані успішно завантажені.")
```

```
    print(f"\n📦 Розмір: {df.shape[0]} рядків,
{df.shape[1]} колонок")
    print("\n📋 Назва колонок:")
    print(df.columns.tolist())
    print("\n🔍 Перші 5 рядків:")
    print(df.head())
    return df

def save_initial_copy(df, path):
    os.makedirs(os.path.dirname(path), exist_ok=True)
    df.to_csv(path, index=False)
    print(f"\n💾 Копія збережена в: {path}")
```

Файл load_data.py виконує перше завантаження матчів з CSV, друк метаінформації (розмір, колонки, перші рядки) і збереження копії у форматі initial_copy.csv.

Використання всіх модулів

Загалом, ці модулі реалізують повний пайплайн:

- load_data.py — виконує перше завантаження матчів з CSV. , тобто **сирих** (необроблених, початкових) даних, що ще не пройшли етапи очищення або нормалізації.
- clean_data.py — очищення та нормалізація.
- get_club_elo_ratings.py — нормалізація назв команд для злиття з рейтингом.
- feature_engineering.py — генерація ознак для навчання моделі.

Як CharacterEvents пов'язує логіку подій у грі, так ці файли формують повну логіку перед обробки футбольних матчів для подальшого прогнозування результату.

4.5.2 Ключові переваги реалізованої системи

Реалізована система має низку переваг:

- Відкритий код — забезпечує гнучкість та адаптацію до будь-яких турнірів або видів спорту;

- Інтуїтивно зрозумілий інтерфейс — придатний для нефахівців;

- Поєднання статистики та машинного навчання — дозволяє досягти високої точності;

- Масштабована архітектура — дає змогу інтегрувати live-дані, нові моделі, розробити API або мобільний застосунок.

4.5.3 Апробація системи

Для забезпечення коректної роботи моделі передбачення футбольних матчів була реалізована повноцінна система підготовки та обробки даних. Весь процес складається з кількох етапів: завантаження невідредагованих даних, очищення, нормалізація, відповідність назв команд, формування ознак та збереження результатів на кожному етапі.

Завантаження невідредагованих даних

Завантаження даних реалізовано у скрипті load_data.py. Дані зчитуються з файлу laliga_matches_2000-2023.csv, після чого зберігаються копією в initial_copy.csv. Це дає змогу зберігати оригінальний стан даних перед наступними модифікаціями.

Виводиться розмір таблиці, назви колонок та 5 перших рядків



Рисунок *10 —Завантаження даних та перегляд перших записів*

Очищення та нормалізація

Наступний крок — очищення даних (clean_data.py). Тут виконується:

- очищення дублікатів,
- видалення рядків з порожніми ключовими полями (команди, рахунок, дата),
- нормалізація назв команд (видалення символів, перетворення на нижній регістр).

Виводиться кількість рядків до і після очищення та підтвердження збереження до cleaned_matches.csv.

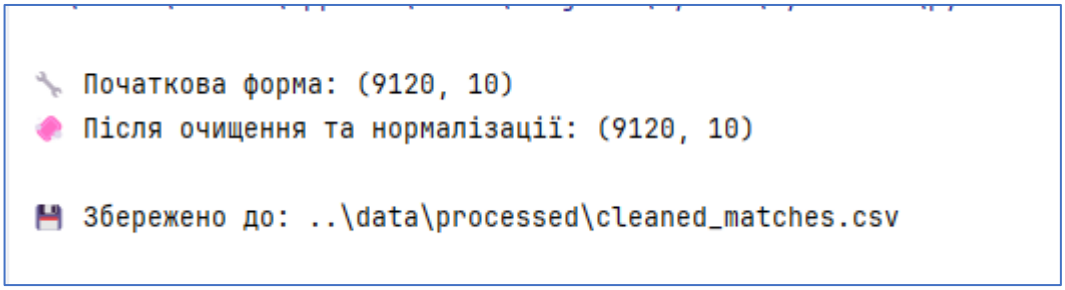


Рисунок *11 — Очищення даних*

Fuzzy Matching назв команд

Далі в get_club_elo_ratings.py реалізовано механізм приведення назв команд до рейтингової таблиці (UEFA club ranking 2025). Використовується

бібліотека rapidfuzz для fuzzy matching. Після співставлення створюється словник відповідностей, який дозволяє замінити **неуніфіковані** (тобто варіативні, неструктуровані) назви на стандартизовані згідно з таблицею рейтингів

Наприклад:

barcelona → fcbarcelona,

sevillafc → sevilla.

Збережений результат: cleaned_matches_fuzzy.csv

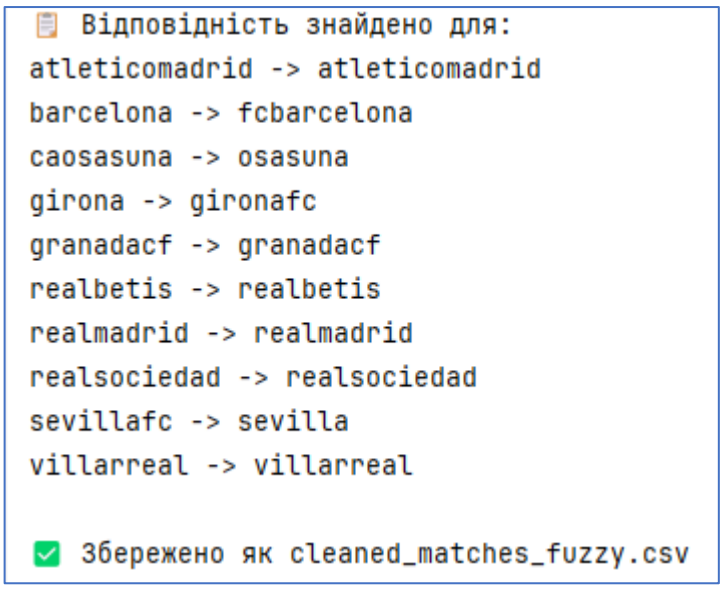


Рисунок *12 —Відповідність назв*

Додавання ознак

У скрипті feature_engineering.py реалізовано обчислення нових ознак:

- result — підсумок матчу (перемога, нічия, поразка),
- goal_difference — різниця голів,
- злиття з рейтингами (для home та away команд),
- заповнення відсутніх рейтингів середнім значенням.

Додаються нові стовпці: home_rating, away_rating.

Виводяться приклади перших 5 матчів після злиття з рейтингами

```
🎯 Додаємо ознаки...
✅ Базові ознаки додані.
✅ Відсутні рейтинги заповнено середнім значенням (15.95)
      home_team away_team  home_rating  away_rating
0  realsociedad    racing    65.000000    15.954214
1  realzaragoza  espanyol    15.954214    15.954214
2   fcbarcelona  malagacf   103.250000    15.954214
3   deplacoruna  athletic    15.954214    15.954214
4    realmadrid  valencia   143.500000    15.954214
```

Рисунок *13 —Додавання ознак до матчів*

У проєкті передбачення футбольного матчу на Python, реалізованому через бібліотеку **Tkinter**, важливу роль відіграє взаємодія користувача з графічним інтерфейсом. Користувач вводить назви команд (домашньої та гостьової), після чого запускається повний цикл обробки даних, перекладу назв, нормалізації, пошуку рейтингу та виклику моделі.

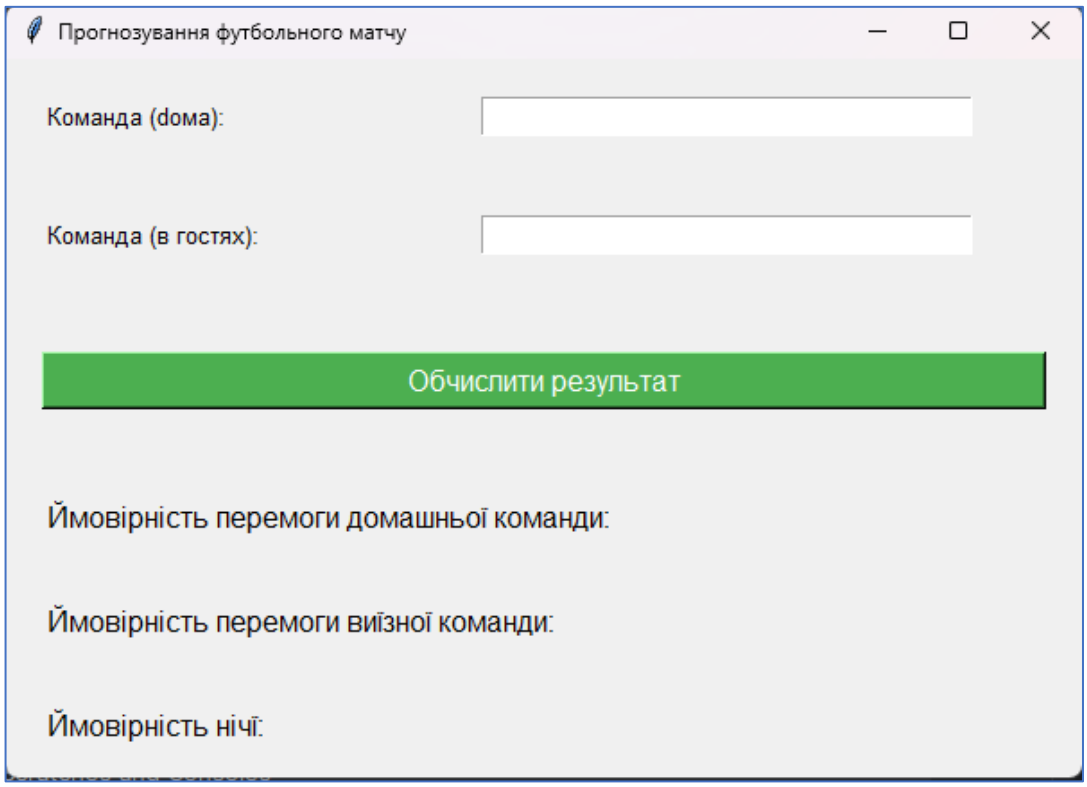


Рисунок *14 —Початковий екран*

Відображається форма введення з двома полями: домашня команда та гостьова команда. Кнопка "Обчислити результат" викликає головну функцію calculate_probabilities() із main.py, яка відповідає за запуск обробки.

На рисунку 6 зображено результат якщо буде невірне вказання назв.

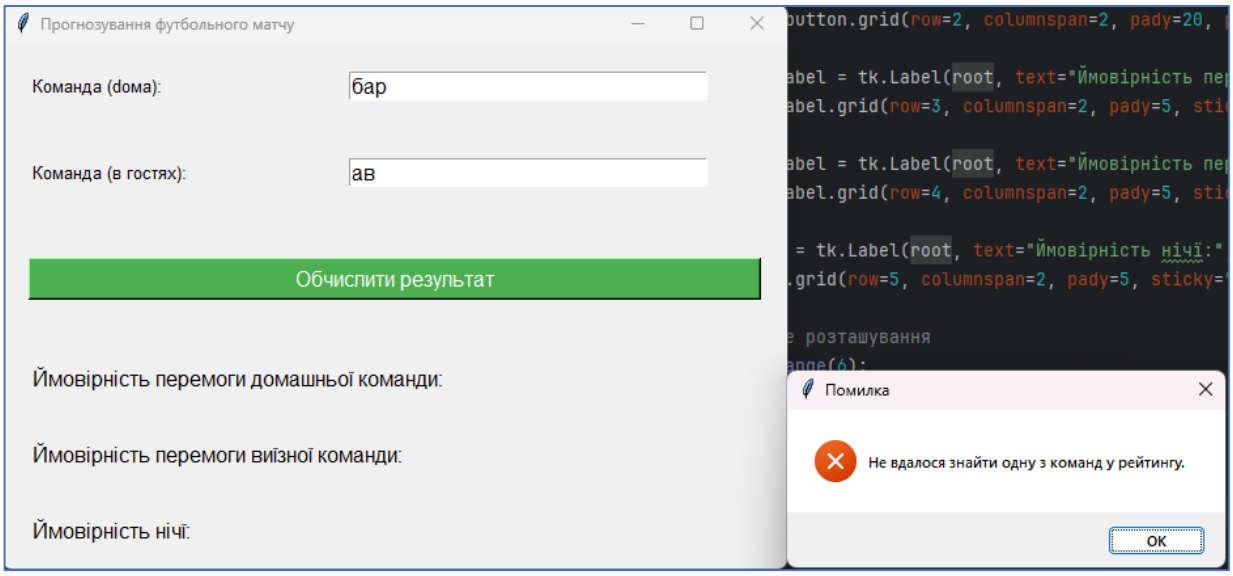


Рисунок *15 Перевірка вхідних даних*

Якщо користувач вводить неповні або неправильні назви команд (наприклад, "бар" і "ав"), відображається повідомлення про помилку: *"Не вдалося знайти одну з команд у рейтингу"*.

Ця перевірка реалізована у функції predict_match() в predict.py.

На рисунку 7 зображено успішне передбачення

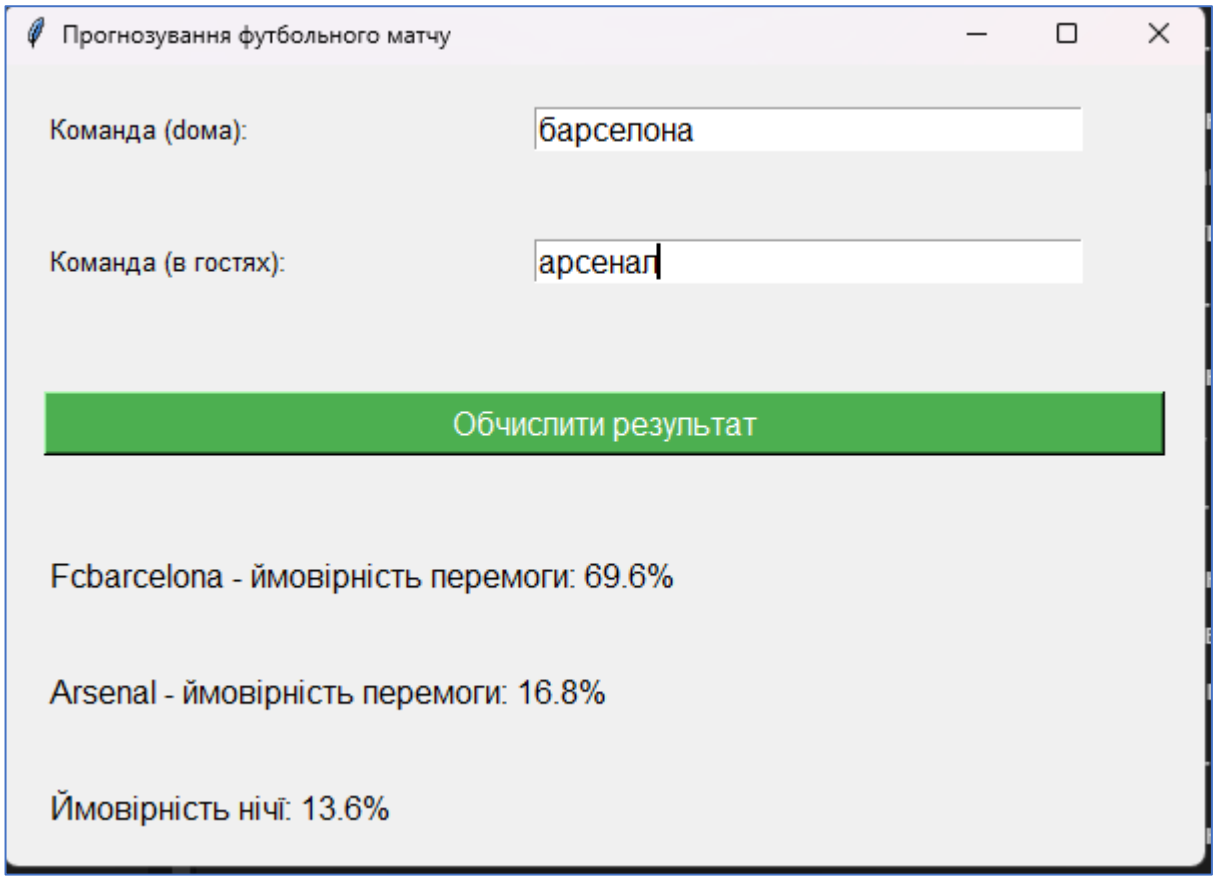


Рисунок *16 —Успішне передбачення*

Якщо команди знайдені (наприклад, "барселона" і "арсенал"), система автоматично перекладає їх, нормалізує, знаходить найближчий збіг у рейтингу та подає модель для обрахунку ймовірностей. Результати виводяться у форматі:

- Назва домашньої команди та ймовірність перемоги
- Назва гостьової команди та ймовірність перемоги
- Ймовірність нічиєї

4.6 Перевірка прогнозів моделі

У цьому розділі представлено демонстрацію тестування створеної системи на прикладі справжніх футбольних поєдинків 2025 року. Кожен матч аналізувався з формуванням прогнозу через користувацький інтерфейс, а

отримані результати порівнювалися з реальними підсумками, взятими з доступних відкритих джерел.

Приклад 1. Матч «Севілья» — «Барселона»

- **Реальний результат (Ла Ліга, 09.02.2025):**

  Інформація з інтернету: матч завершився з рахунком **Севілья 1 : 4 Барселона**, перемога гостей.

- **Прогноз моделі:**
  - **Fcbarcelona** — ймовірність перемоги: **72.2%**
  - **Sevilla** — ймовірність перемоги: **9.5%**
  - **Ймовірність нічиєї:** 18.3%
- **Оцінка результату:**

  Прогноз моделі повністю відповідає фактичному результату. Найбільша ймовірність була у перемоги «Барселони», що й сталося.

Приклад 2. Матч «Атлетіко Мадрид» — «Барселона»

- **Реальний результат (Кубок Іспанії):**

  Інформація з інтернету: матч завершився з рахунком **0 : 1** на користь «Барселони».

- **Прогноз моделі:**
  - **Atleticomadrid** — ймовірність перемоги: **51.3%**
  - **Fcbarcelona** — ймовірність перемоги: **27.9%**
  - **Ймовірність нічиєї:** 20.8%
- **Оцінка результату:**

У даному випадку прогноз не збігся з фактичним результатом. Модель надала перевагу господарям поля — «Атлетіко Мадрид», що логічно з огляду на домашній статус команди та статистичні дані. Проте матч завершився перемогою гостей. Важливо розуміти, що модель надає не гарантію, а ймовірнісну оцінку, і подібні ситуації не є помилками в класичному сенсі.

Висновок щодо точності:

Однією з причин неточностей у прогнозуванні може бути **якість та повнота даних, використаних для навчання моделі**. У відкритому доступі

відсутня узгоджена, повна та актуальна база даних з рейтингами команд, складами, формою тощо, тому при побудові навчального набору довелося:

- використовувати частково застарілі історичні дані;
- **доповнювати рейтинги команд вручну**, що могло вплинути на об'єктивність ознак.

У подальшому планується:

- **оновити базу даних** на основі більш актуального джерела;
- **автоматизувати збір даних через парсинг або API**, щоб забезпечити стабільну актуалізацію рейтингових та ігрових показників.

4.7 Висновок з розділу 4

У цьому розділі о детально розглянуто процес розробки програмного забезпечення для передбачення результатів футбольних матчів із використанням алгоритмів машинного навчання та мови програмування Python. Основні аспекти включають аналіз предметної області, функціональні вимоги, архітектуру системи, структуру обробки даних та реалізацію графічного інтерфейсу.

ВИСНОВКИ

1. У процесі роботи над кваліфікаційною роботою проведено аналіз літературних джерел, присвячених прогнозуванню результатів спортивних змагань.
2. Проведено аналіз існуючих аналогів, таких як Flashscore, Forebet та SoccerVista, що використовують власні алгоритми, але з прихованою методологією.
3. На основі аналізу для реалізації системи обрано мову програмування Python. Ця мова має широке застосування в Data Science, підтримує велику кількість бібліотек для машинного навчання, розробки інтерфейсів та перекладу тексту.
4. У рамках роботи розроблено повноцінну систему прогнозування результатів футбольних матчів, що охоплює всі ключові етапи — від завантаження вихідних даних у форматі CSV та їх попередньої обробки, до побудови ознак, навчання моделі машинного навчання та створення зручного інтерфейсу користувача.
5. Працездатність і ефективність системи перевірена на результатах реальних футбольних матчів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bishop C. M. Pattern Recognition and Machine Learning. Springer, 2006. 738 p.
2. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, 2022. 856 p.
3. James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning. Springer, 2013.
4. Kaggle. Платформа для змагань і відкритих датасетів з машинного навчання. 2025.
5. UEFA. Офіційний рейтинг клубів УЄФА за 2025 рік. 2025.
6. FiveThirtyEight. Статистичні прогнози спортивних подій. 2025.
7. Forebet. Математичні прогнози футбольних матчів. 2025.
8. Kyriacou G. Football Match Prediction using Machine Learning. GitHub. 2023.
9. Techibytes. Огляд AI-платформ для прогнозів футбольних матчів. 2024.
10. Wyscout. Професійна аналітика для скаутів і футбольних клубів. 2025.
11. Python Software Foundation. Офіційна документація мови програмування Python. 2025.
12. Pedregosa F. та ін. Scikit-learn: Machine Learning in Python. 2023.
13. McKinney W. Data Analysis with Python using Pandas. 2022.
14. Hunter J. D. Matplotlib: A 2D Graphics Environment. 2007.
15. Waskom M. L. Seaborn: Statistical Data Visualization. 2023.

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Чистіков Максим Сергійович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ, спеціальність 121 Інженерія програмного забезпечення освітньої програми "Програмне забезпечення систем", адреса електронної пошти se121-28@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «КОМП'ЮТЕРНА СИСТЕМА ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ СПОРТИВНИХ ЗМАГАНЬ» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 02.06.2025 _____ Чистіков Максим Сергійович
(підпис) (прізвище та ініціали) (керівник)

Дата 02.06.2025 _____ Заяц Валерій Іванович
(підпис) (прізвище та ініціали) (керівник)