Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ КІБЕРСПОРТИВНИХ МАТЧІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF TOOLS FOR PREDICTING ESPORTS MATCH OUTCOMES

Виконав: студент(ка) 2 курсу, групи КНТ-214м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

КОВЕРСУН О.С.
(ПРІЗВИЩЕ та ініціали)
Керівник ЛЕОЩЕНКО С.Д.
(ПРІЗВИЩЕ та ініціали)
Рецензент ПОЛЯКОВ М.О.
(ПРІЗВИЩЕ та ініціали)

2025

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ

Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____Сергій СУББОТІН
"____" _____2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОВЕРСУНА Олександра Сергійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація засобів прогнозування результатів кіберспортивних матчів. Research and Software Implementation of Tools for Predicting Esports Match Outcomes

керівник проєкту (роботи) д-р.філос., доцент, ЛЕОЩЕНКО Сергій Дмитрович ,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від " 30 " вересня 2025 року № 447
2. Строк подання студентом проєкту (роботи) 8 грудня 2025 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Вибір і обґрунтування структури системи. 3. Основні рішення реалізації компонентів системи. 4. Експлуатація, тестування та дослідження.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЕОЩЕНКО С.Д., доцент		
Нормоконтролер	КАЛІНІНА М.В., асистент		

7. Дата видачі завдання « 30 » вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Вибір і обґрунтування структури системи.	4-5 тижні	Розділ 2
4	Основні рішення реалізації компонентів системи.	6-8 тижні	Розділ 3
5	Експлуатація, тестування та дослідження	9 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів	10-11 тижні	Додатки
	до неї.		
7	Нормоконтроль та рецензування.	12 тиждень	
8	Захист роботи.	12 тиждень	

Студент(ка) _____ Олександр КОВЕРСУН
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Сергій ЛЕОЩЕНКО
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра: 103 с., 6 табл., 40 рис., 3 дод., 32 джерела.

ПРОГНОЗУВАННЯ, КІБЕРСПОРТИВНІ МАТЧІ, PYTHON, STREAMLIT, POSTGRESQL, ВЕБЗАСТОСУНОК, РЕЗУЛЬТАТИ.

Об’єкт дослідження – процес прогнозування результатів кіберспортивних матчів.

Предмет дослідження – методи та засоби для прогнозування результатів кіберспортивних матчів.

Мета роботи – підвищення точності прогнозування результатів кіберспортивних матчів.

Матеріали, методи та технічні засоби: мова програмування Python, інтегроване середовище розробки PyCharm, фреймворк Streamlit, система керування базами даних PostgreSQL, персональний комп’ютер під управлінням операційної системи Microsoft Windows 11.

Результати. Створено вебзастосунок, який надає можливість користувачам швидко та зручно визначати результати кіберспортивних матчів.

Висновки. Розроблено програмне забезпечення для прогнозування результатів кіберспортивних матчів на основі аналізу статистичних даних. Застосунок дозволяє користувачу формувати прогноз на вибраний матч, отримувати ймовірності перемоги, переглядати статистику команд та гравців.

Галузь використання – кіберспорт та аналітика змагань. Програмне забезпечення може застосовуватись: аналітиками та тренерами кіберспортивних команд для підготовки до матчів, платформами статистики та кіберспортивними медіасервісами, користувачами, які цікавляться аналітикою матчів та хочуть отримувати об’єктивні прогнози на основі даних.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 103 pages, 6 tables, 40 figures, 3 appendixes, 32 sources.

PREDICTION, ESPORTS MATCH, PYTHON, STREAMLIT, POSTGRESQL, WEB APPLICATION, OUTCOMES

Object of research is the process of predicting esports match outcomes.

Subject of research is methods and tools for predicting esports match outcomes.

The purpose of the work is to enhance the accuracy of esports match outcomes prediction.

Materials, methods, and tools: programming language Python, integrated development environment PyCharm, Streamlit framework, database management system PostgreSQL, personal computer under the control of the Microsoft Windows 11 operating system.

Results. A web application has been developed that enables users to quickly and conveniently predict the results of esports matches.

Conclusions. Software has been created for predicting esports match outcomes based on the analysis of statistical data. The application allows the user to form a prediction for a selected match, obtain victory probabilities, and view team and player statistics.

The field of use is esports and competition analytics. The software can be used by esports analysts and team coaches for match preparation, by statistic platforms and esports media services, as well as by users interested in match analytics who want to receive objective, data-driven predictions.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ML – Machine Learning;

SQL – Structured Query Language;

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

У наш час кіберспорт перетворився на одну з найбільш динамічних і швидкозростаючих галузей цифрової індустрії. Щодня відбуваються сотні матчів, у яких беруть участь професійні команди та гравці з усього світу. Обсяг статистичних даних постійно збільшується: результати матчів, індивідуальні показники гравців, зміни складів команд, особливості карт та багато інших факторів. Через таку кількість інформації людині складно здійснювати об’єктивний аналіз та робити точні прогнози результатів майбутніх матчів без використання спеціалізованих програмних інструментів.

Актуальність даної теми полягає в потребі автоматизації процесу аналізу кіберспортивних даних та підвищення точності прогнозування результатів матчів. Прогнозування є важливим для широкого кола користувачів: аналітиків, тренерів, кіберспортивних організацій, медіасервісів та звичайних глядачів, які хочуть оцінити ймовірність перемоги тієї чи іншої команди, ґрунтуючись не на інтуїції, а на фактичних даних.

У цій дипломній роботі розглядається процес дослідження існуючих методів прогнозування результатів кіберспортивних матчів та розробка програмного забезпечення, яке реалізує можливість аналізу статистичних даних і формування прогнозів із застосуванням алгоритмів машинного навчання. Розроблене ПЗ забезпечує зручний доступ до даних, дозволяє робити прогнози на певні матчі та карти, переглядати статистику команд та гравців, яка вплинула на прогноз.

Для реалізації застосунку як основну мову програмування було обрано Python, що є однією з найпопулярніших мов у сфері аналізу даних та машинного навчання. Для створення користувацького інтерфейсу використовується фреймворк Streamlit, який дозволяє швидко та ефективно розробляти вебзастосунки, інтегруючи результати моделювання та візуалізацію даних у зручному вигляді.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття кіберспорту

Кіберспорт – це тип змагань з використанням відеоігор. Кіберспорт часто набуває форми організованих змагань з багатокористувацьких відеоігор, особливо між професійними гравцями, в які грають індивідуально або в командах. Багатокористувацькі змагання довгий час були частиною культури відеоігор, але здебільшого проходили між аматорами, до кінця 2000-х років, коли поява платформ онлайн-трансляцій, зокрема YouTube та Twitch, сприяла сплеску участі професійних геймерів та глядачів [1].

За останні кілька років спостерігається стрімкий зріст цього виду спортивних змагань. Його найбільші події бачили десятки мільйонів глядачів у прямому ефірі, учасники витримують години щоденних тренувань. Замість змагань грубої сили чи традиційних спортивних здібностей, ці змагання зосереджені на швидкості реакції, стратегічному розумінні та часто співпраці, оскільки вони проводяться через відеоігри – що й отримало назву кіберспорт. Індустрія, яка постійно зростала з початку тисячоліть, отримала ще більший поштовх ближче до мейнстріму під час нещодавньої пандемії Covid-19 [2]. На рис. 1.1 зображено діаграму найпопулярніших кіберспортивних подій за максимальною кількістю глядачів в минулому 2024 році.

Рисунок 1.1 – Діаграма перегляду кіберспортивних подій у 2024 році [3]

Для порівняння найпопулярніший спортивний турнір на даний момент був чемпіонат світу з футболу 2022 року, де фінал дивилося більше 5 мільярдів людей з усього світу [4]. Також можна подивитися на дуже популярний спорт американський футбол, середні перегляди крупних турнірів якого сягають близько 100 мільйонів [5].

Цифри перегляду у кіберспорті не настільки масштабні, але це зумовлено більш короткою історією існування та наприклад не досить розвиненою трансляцією на звичайному телебаченні. Основна аудиторія кіберспорту – молодь, яка рідше дивиться телебачення і не завжди враховується традиційними системами вимірювання рейтингів, також частина трансляцій не має глобального ліцензування, тому вони недоступні в деяких регіонах або доступні лише через VPN.

З фінансового боку кіберспорт програє традиційному спорту, сумарні доходи кіберспорту на 2025 більше одного мільярда доларів, в той час як дохід лише одних змагань Формула-1 за 2024 рік склав близько трьох мільярдів доларів [6]. Це зумовлюється тим, що більша частина доходу кіберспорту приходить через спонсорство та стрімінг, тоді як у традиційному спорті більше через телевізійні права, продаж квитків, стадіони. Проте, регулярні спортивні змагання також обов’язково залежать від величезних, дорогих арен як місць проведення, тоді як кіберспорт має унікальну перевагу в тому, що може існувати повністю децентралізовано. Також, менші прибутки зумовлено віковою категорією, за оцінками, 38% фанатів кіберспорту віком від 16 до 24 років, і з віком та початком роботи вони зможуть фінансово підтримувати галузь [2].

Але призові фонди турнірів не завжди уступають традиційному спорту, на рис. 1.2 можна побачити діаграму порівняння.

Рисунок 1.2 – Порівняння призових фондів [7]

1.2 Визнання кіберспорту як виду спорту

Сидіти перед комп'ютером і грати у відеоігри – це не той образ, який спадає на думку, коли людина думає про спортсмена. Натомість, зазвичай стереотипно асоціюється образ людини, яка може бути не в хорошій фізичній формі та не мати спортивних здібностей. З розвитком технологій, гравці у відеоігри, які змагаються, починають демонструвати ті ж атлетичні якості, що й традиційні спортсмени. Концепція відеоігор також змінилася. Замість того, щоб грати у відеоігри для розваги, люди починають грати у відеоігри на змаганнях, що дуже нагадують спортивні змагання [8].

Одна з найбільших дискусій щодо кіберспорту полягає в тому, чи можна визначити змагальні відеоігри як вид спорту. Визначення спорту намагалися дати багато разів, але універсального визначення так і не було визначено. Замість остаточного академічного визначення, люди посилаються на визначення з Оксфордського словника англійської мови: «Діяльність, що включає фізичні зусилля та майстерність, в якій окрема особа або команда змагається проти іншої особи або інших заради розваги» [8].

Під час гри змінюється базальний метаболічний рівень (МЕТ), що є науковим критерієм фізичної активності. Оскільки МЕТ може бути використаний для визначення навантаження, можливий зв’язок із тим, як МЕТ змінюється під час гри у відеоігри. Додатково, можуть бути використані рівні кисню (VO_2), помірна фізична активність матиме 40-60 % резерву VO_2 та/або 4-6 МЕТ [8].

Багато досліджень демонструють те що під час гри в людини збільшується рівень МЕТ та пульс, як і в звичайному спорті, також топові кіберспортмени можуть робити близько 500 дій за хвилину, що можна вважати фізичною активністю [9]-[10].

Інший ж термін у визначенні спорту – майстерність, очевидно присутній у кіберспорті. Професійні кіберспортсмени, так само як і спортсмени традиційних видів спорту, мають розклад тренувань, працюють з тренерами, аналітиками, спортивними психологами, дієтологами. Тренувальний процес включає перегляд вже зіграних матчів для пошуку помилок, або місць для покращення результатів, наприклад як у футболі, розбір тактики суперника, для підвищення ймовірності передбачити дії суперників та підвищити ймовірність перемоги, командні тренування та участь у турнірах. Разом із цим кіберспортсмени ведуть роботу над концентрацією, реакцією, стресостійкістю та навичками комунікації всередині команди.

Для багатьох професіоналів шлях починається ще в дитинстві або підлітковому віці, коли, подібно до юних футболістів чи тенісистів, вони прагнуть стати кращими в своїй грі. Роки наполегливих тренувань, участь у аматорських турнірах, пошук команди, безліч поразок і спроб – усе це формує майстерність. Гравець поступово підвищує індивідуальний рівень, вчиться працювати в команді, розвиває дисципліну та самоконтроль.

В Україні кіберспорт визнали спортом у вересні 2020 року, для підтримки життєдіяльності, розвитку та популяризації цієї сфери у нашій

країні існує Федерація кіберспорту України, яка проводить змагання у різних дисциплінах в усіх містах України.

Також, в 2027 році планується проведення кіберспортивних олімпійських ігор, хоча традиційні кіберспортивні ігри, такі як Counter-Strike, Dota, League of Legends та Valorant, вважалися «занадто жорстокими» для включення до кіберспортивних змагань олімпійського типу [11].

1.3 Прогнозування результатів кіберспортивних матчів

Як і в звичайному спорті є люди, які зацікавлені в тому щоб спробувати дізнатися переможця матчів, або визначити ймовірності перемоги того чи іншого гравця або команди, до початку цих матчів. Це можуть бути як букмекерські контори, які вносять значний фінансовий вклад в індустрію кіберспорту, так і тренери, або аналітики команд, яким потрібно знайти слабкі місця суперників і зрозуміти приблизні шанси перемоги. Проблема прогнозування результатів кіберспортивних матчів є актуальною, оскільки кількість турнірів і обсяг статистичних даних стрімко зростає, а рішення часто впливають на фінансові ставки, роботу аналітиків та підготовку команд.

Проте, порівняно зі звичайним спортом, визначення результатів складніше, через те що:

– у відеоіграх регулярно випускаються патчі (нові версії гри), змінюються карти, баланс персонажів – це робить модель, натреновану на минулих даних, менш стійкою;

– у кіберспорті команди часто змінюються, гравці переходять з організації в організацію, команди реорганізуються, немає чітко визначених дат трансферних вікон;

– менше історичних даних порівняно з традиційним спортом, що зменшує об’єм тренувальних даних для моделі;

– нестабільність і фактори випадковості. Як і у традиційному спорті, в кіберспорті присутній фактор випадковості, але через множинність дрібних впливів він може бути вищим.

Також, можна визначити загальні, що не залежать від конкретної гри, показники за якими зазвичай розраховується результати матчів:

– історичні результати: перемоги/поразки, % перемог, історія зустрічей (head-to-head);

– поточна форма команди/гравців: останні N матчів, серія перемог/поразок;

– статистичні данні гравців: індивідуальні метрики ефективності в залежності від дисципліни;

– контекст гри: формат матчу (кількість карт), стадія: груповий етап / плей-оф / фінал, чи є матч на вибування, наявність глядачів;

– командна динаміка: зміни складу, тренера, психологічна форма;

– ін-матчеві фактори: перевага після стартових раундів / перших хвилин, економічні рішення (як команда розподіляє ресурси), психологічна стійкість після невдалих моментів, помилки.

Для визначення результатів матчів можна використовувати різні методи та моделі, усе залежить від обраної гри (дисципліни), неможна буде застосувати одну модель для різних дисциплін.

На рис. 1.3 зображено діаграму найбільш популярних ігор за піковою кількістю переглядів.

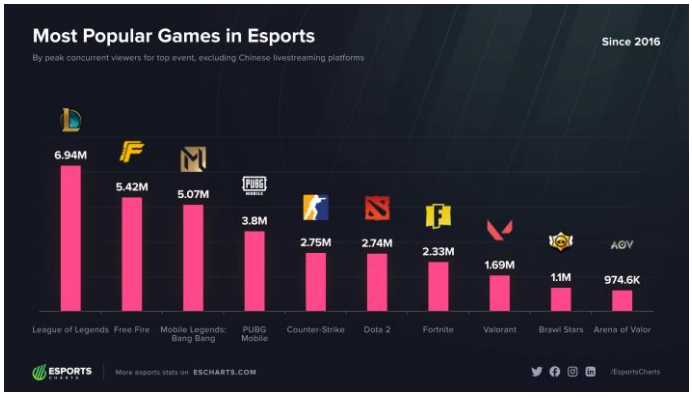


Рисунок 1.3 – Найпопулярніші дисципліни [12]

Пропонується розглянути які методи можна використовувати, та які параметри впливають на розрахунок ймовірностей для гри Counter Strike 2. Ця гра є найбільш популярною в Україні і в Європі, на даний момент поточна кількість гравців є найбільшою серед усіх ігор. Ця гра відноситься до жанру FPS, де основна частина ігрового процесу це знищення ворогів із різноманітної вогнепальної зброї. Характерною особливістю жанру є вид «з очей» головного героя [13]. Існує багато ігор такого жанру, тому деякі показники оцінки ймовірності можуть співпадати з різними іграми цього ж жанру.

Для задачі прогнозування результатів матчу (перемога/поразка, % ймовірності) Counter Strike 2 використовується класична задача бінарної класифікації. В табл. 1.1 показано порівняння груп методів які можна застосувати для вирішення цієї задачі.

Таблиця 1.1 – Порівняння методів

Критерій порівняння	Класичні ML-моделі	Ансамблеві моделі	Нейромережі	Кластеризація
1	2	3	4	5
Моделі	Logistic Regression, LDA, SVM	XGBoost, LightGBM, CatBoost, Random Forest	MLP, RNN, LSTM, CNN	K-Means, DBSCAN
Тип даних	Табличні (статистика матчів, показники гравців)	Табличні, змішані (числові + категоріальні ознаки)	Складні структури (послідовності, відео)	Табличні числові дані (для групування за схожістю)
Вимоги до обсягу даних	Невеликі й середні датасети	Середні й великі датасети	Дуже великі датасети, багато матчів	Невеликі й середні
Якість прогнозування	Базова модель, точність середня	Найвища точність у табличних даних	Ризик розміру ансамблів	Середня, лише групує

Продовження таблиці 1.1

1	2	3	4	5
Час навчання	Дуже швидкий	Швидкий / Середній	Повільний	Дуже швидкий
Інтерпретація моделі	Легка інтерпретація	Частково інтерпретована	Найгірша інтерпретація (чорний ящик)	Інтерпретована (центроїди кластерів)
Переваги	Простота, швидкість	Максимальна точність	Висока точність у складних даних	Виявлення патернів, сегментація команд/стилів
Недоліки	Гірше захоплюють складні нелінійні	Потребують налаштування моделей	Високі вимоги до обчислень, складність	Не прогнозує результат напряму

В дослідженні [14] показано приклад застосування машинного навчання для прогнозування результатів у CS:GO (яка наразі оновлена до CS2). Дані для моделі збирались шляхом вебскрейпінгу з платформи FACEIT [15], витягуючи інформацію з демо-файлів (повторів) матчів.

Спочатку модель враховувала лише ролі гравців у команді, але низька точність змусила збільшити кількість характеристик (до 50 фіч), серед яких:

– кількість вбивств/смертей;

– тип зброї;

– тип вбивства;

– екіпірування;

– позиція гравця;

– рух прицілу перед вбивством тощо.

В дослідженні використовувалися методи кластеризації K-середніх та еволюційні алгоритми – для кластеризації характеристик і нейрона мережа

(логістична та ReLU активація, бінарна крос-ентропія) – для прогнозування результату матчу. Найкраща точність прогнозу: 65.11% при 32 кластерах.

Дослідження [16] також аналізує CS:GO, але використовує інші методи.

Дані взяті з платформи hltv.org [17], яка є новинним вебсайтом і форумом, що висвітлює професійні новини, турніри та статистику кіберспорту Counter-Strike 2, де було зібрано 15 характеристик гравців + 2 загальні фічі (наприклад: тип команди, баланс грошей, поточна вартість екіпірування, рахунок тощо).

Використовувалися методи Random Forest, XGBoost, багатошаровий персептрон (MLP).

В результаті Random Forest – найкращий для прогнозування переможця раунду: 64.07%, а XGBoost – найкращий для прогнозування переможця матчу: 62.37%.

Після порівняння моделей та методів можна зробити висновок, що класичні алгоритми машинного навчання підходять у випадках, коли кількість параметрів невелика, а головною вимогою є зрозумілість та швидкість побудови прогнозу.

Ансамблеві моделі демонструють найкращі результати у задачах прогнозування кіберспортивних матчів, оскільки здатні працювати з великою кількістю ознак та враховувати складні взаємозв’язки між параметрами – саме ці моделі найчастіше використовуються в дослідженнях CS:GO/CS2.

Нейронні мережі доречні, коли доступний великий обсяг історичних даних, і система повинна навчатися прихованим патернам поведінки команд.

Методи кластеризації не прогнозують результат безпосередньо, проте допомагають структурувати дані (наприклад, групувати стилі гри команд) та покращують якість прогнозів інших моделей.

1.4 Огляд аналогів

Існує низка сервісів та інструментів, що працюють з аналітикою кіберспортивних матчів Counter-Strike 2: вони можуть відрізнятися доступністю, глибиною статистики, наявністю алгоритмів прогнозування та цільовою аудиторією (аналітики команд, букмекерські контори, звичайні користувачі). Більшість рішень або фокусуються на загальній статистиці та аналізі ігрових даних, або використовують власні закриті алгоритми без можливості гнучкого налаштування.

Перші сервіси можна віднести до категорії “Betting Analysis” сайти, тобто на цих сайтах є лише аналітика по певному матчу. На них немає можливості вибрати команди, в них немає інтерактивних інтерфейсів, в більшості випадків також немає відсотку ймовірностей, яка команда переможе. На таких сайтах вказано які фактори можуть вплинути на результат, та яка команда за цією аналітикою має виграти. Прикладом такого сайту є bo3.gg [18], також на сайті hltv.org є розділ з такою аналітикою. На рис. 1.4 та 1.5 можна побачити яким чином виглядає аналітика на відповідних сайтах.

Інформація на таких ресурсах є корисною, однак вона орієнтована радше на огляд матчу, а не на моделювання результату. Користувач не може змінювати вхідні параметри – наприклад, склади команд, карту чи останні результати, щоб подивитися, як це вплине на ймовірність перемоги. Таким чином, ці сервіси виконують роль “пасивної аналітики”: вони надають статистику, але не дозволяють будувати власні прогнози або експериментувати зі зміною факторів матчу. Це створює потребу в окремому інструменті, який би надавав активний прогноз із можливістю взаємодії користувача, та дозволяв би не лише дивитися дані, а й використовувати їх для отримання результатів.

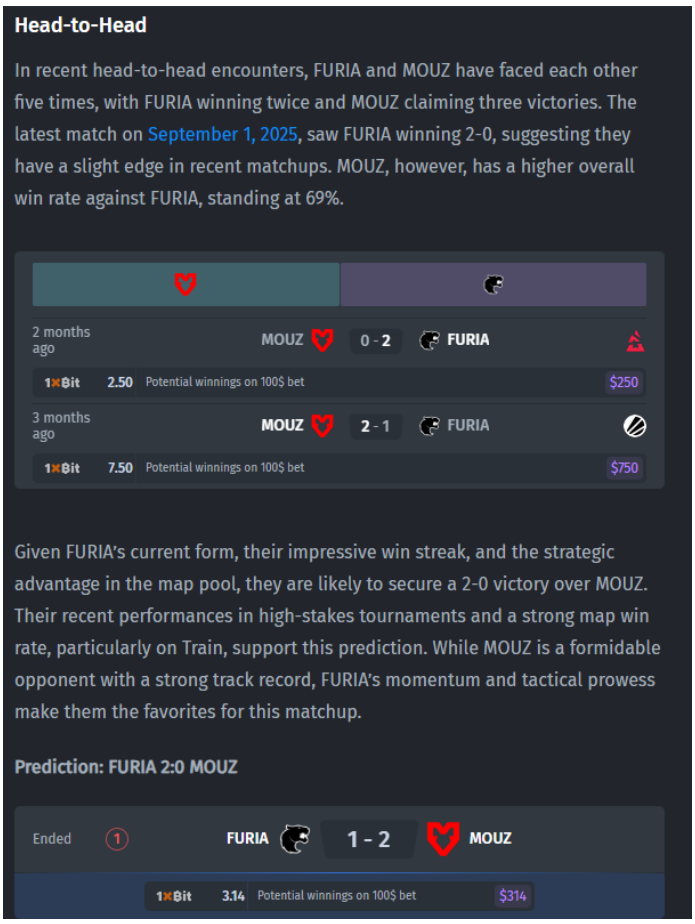


Рисунок 1.4 – Аналітика на сайті bo3.gg [18]

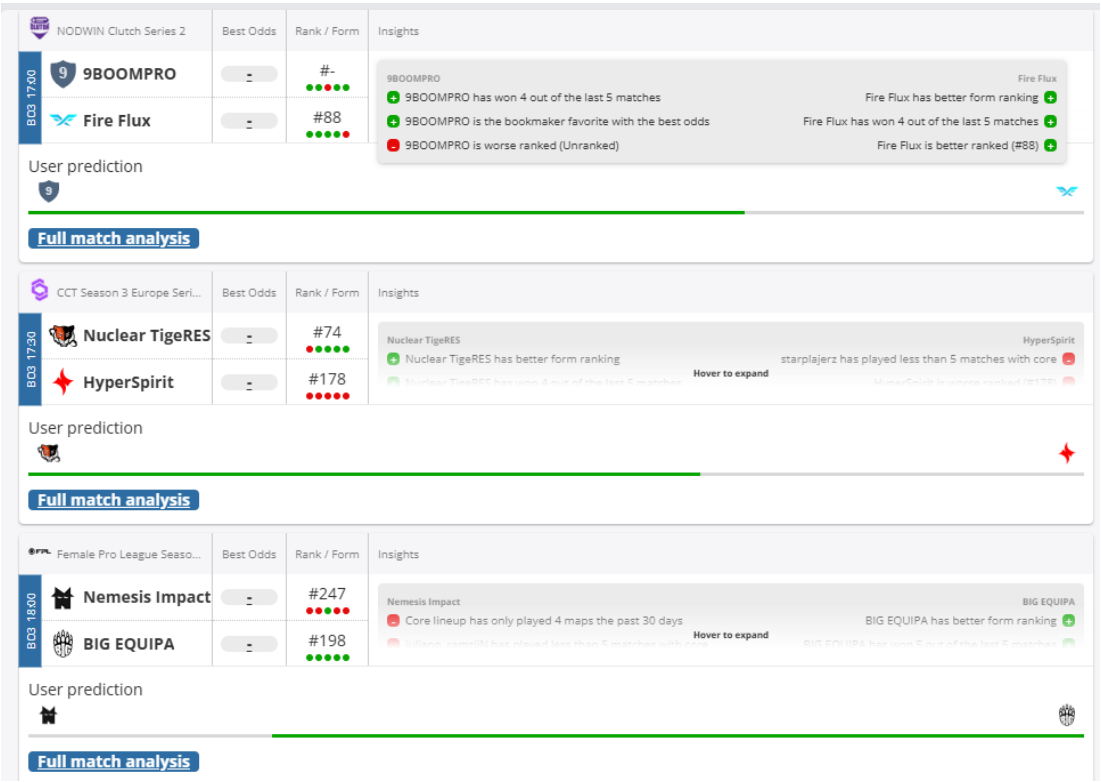


Рисунок 1.5 – Аналітичний розділ сайту hltv.org [17]

Другий сервіс є розширенням для платформи FACEIT, яка є кіберспортивною платформою здебільшого для звичайних гравців, тобто вона не орієнтована на проведення професійних масштабних турнірів, це розширення дозволяє подивитися ймовірність перемоги команди перед початком матчу, ґрунтуючись на статистиці гравців. Це розширення – тому інтерактивного інтерфейсу немає, воно орієнтовано на індивідуальні ігри звичайних гравців, а не для вибору певних професійних команд, також в тому інтерфейсі які додається розширенням вказано лише ймовірності перемоги на певних мапах, тобто функціонал дуже маленький.

Третій сервіс, який розглядається, це відкритий проєкт з github [19], який збирає дані з hltv.org, використовує машинне навчання (класичні та ансамблеві ML-моделі) для оцінки ймовірності перемоги команд. Містить код, але не має готового графічного інтерфейсу для кінцевого користувача і потребує технічного налаштування. Також, моделі навчалися більше ріку тому і навіть якщо зробити розгортання цього проєкту, дані з сьогоденних матчів будуть відрізнятися через заміни в складі, зміни відсотку перемог команд на певних картах й в цілому, зміни показників гравців. Також моделі навчалися лише на крупних турнірах і такі моделі не підійдуть для прогнозування менш відомих й менш сильних команд. Його інтерфейс з демонстрації автора можна побачити на рис. 1.6, а його прогноз на рис. 1.7.

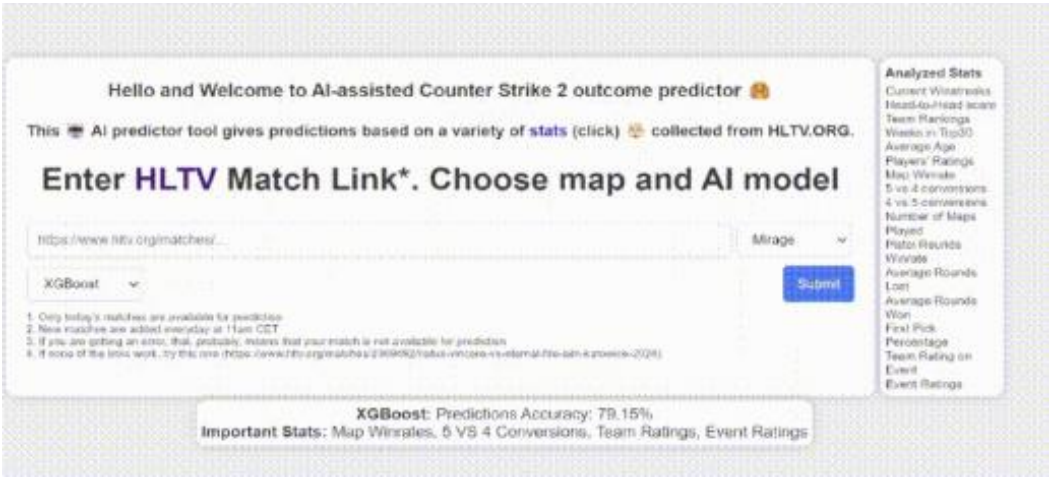


Рисунок 1.6 – Інтерфейс демонстрації відкритого проєкту [19]

В цьому застосунку більше функцій: можна обирати карту, модель машинного навчання, також демонструється список факторів які впливають на прогноз, вказується точність моделей. Демонструється яка команда виграє, проте немає % ймовірності виграшу.

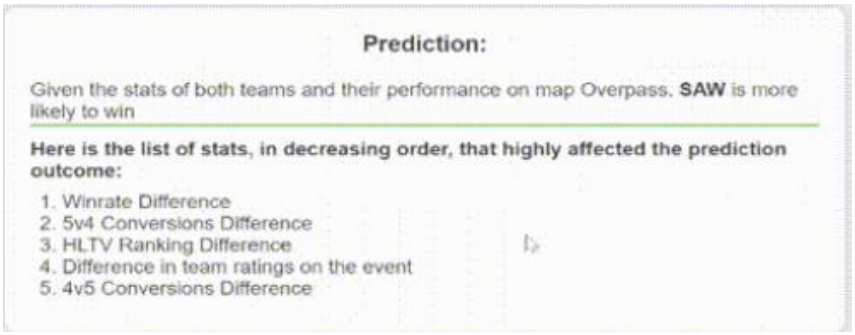


Рисунок 1.7 – Прогноз проєкту з github [19]

В табл. 1.2 вказано порівняння описаних аналогів.

Таблиця 1.2 – Порівняння аналогів

Критерії порівняння	Сайти з аналітикою (hltv.org, bo3.gg)	FACEIT CS2 Predictor	Point516 Github проєкт
Тип	Вебсайти	Браузер-розширення	Відкритий код
Наявність інтерактивного інтерфейсу	Немає	Немає	Потрібен окремий деплой, більше під розробників
Вибір матчів вручну	Немає	Немає	+-, можливо вставляти посилання на матч
Застосовані технології	Статистична аналітика та моделі	Логістична регресія	Алгоритми машинного навчання
Пояснення прогнозування	Лише в аналітичному вигляді	Немає	Є, прозоре
Автоматичний збір даних	Є, але дані недоступні користувачу	+	+
Вартість	Безкоштовно	Безкоштовно	Безкоштовно

1.5 Висновки за розділом 1

В даному розділі було досліджено предметну область, визначено ключові поняття, пов’язані з об’єктом дослідження.

В подробицях розібрано що таке кіберспорт, чим він відрізняється від традиційних видів спорту, чому його потрібно вважати видом спорту та підкреслено більшу складність в розрахунку результатів матчів порівняно з традиційним спортом.

Розібрано які методи машинного навчання можна використовувати для прогнозування результатів кіберспортивних матчів у дисципліні Counter Strike 2. Розібрано існуючі аналоги для цих цілей.

На основі дослідження предметної області та зважаючи на порівняльну таблицю аналогів виявлено, що для кінцевого користувача немає зручних безкоштовних додатків для прогнозування результатів кіберспортивних матчів у CS2 із можливістю вибору конкретних матчів та отримання прозорого пояснення прогнозу. Більшість рішень або є закритими аналітичними сервісами, або обмежені вузьким функціоналом або орієнтовані переважно на розробників і потребують складного розгортання.

Для розробки програмного забезпечення було використано ідею проєкту Point516, зокрема підхід до збору статистичних даних та прогнозування результатів матчів за допомогою моделі машинного навчання. Власна реалізація передбачає скрейпінг даних із hltv.org та формування прогнозів на основі статистики гравців і команд із можливістю виводу детальної статистики та зручного користувацького інтерфейсу для вибору майбутніх матчів.

В результаті аналізу предметної області було визначено вимоги до розроблюваного ПЗ та поставлено завдання на розробку (див. додаток А).

2 ВИБІР І ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ

2.1 Вибір засобів для проєктування та реалізації

Оскільки для формування точного прогнозу необхідні актуальні дані, які постійно оновлюються на сайті, доцільно реалізувати вебзастосунок. Це дозволяє користувачеві отримувати останню інформацію та результати прогнозів у режимі онлайн без необхідності завантажувати оновлення або локальні файли, що забезпечує зручність та оперативність у використанні.

Порівнюючи мови програмування для створення вебзастосунків з використанням машинного навчання, часто звертають увагу на такі параметри:

– швидкість виконання коду;

– зручність розробки;

– зручність інтеграції з базами даних;

– доступність бібліотек машинного навчання.

Порівняння, яке зроблено в табл. 2.1, показало, що кожна мова програмування має свої переваги та недоліки для розробки вебзастосунків із використанням машинного навчання. Наприклад, JavaScript добре підходить для створення інтерактивних інтерфейсів і забезпечує високу кросплатформеність, однак має обмежену підтримку готових бібліотек для глибинного аналізу даних і машинного навчання. Java і C# демонструють високу швидкість виконання коду та масштабованість, але для швидкої розробки прототипів та інтеграції з ML-бібліотеками вони потребують більшого часу на налаштування та вивчення специфічних фреймворків. Python, у свою чергу, поєднує простоту синтаксису, потужну екосистему бібліотек для машинного навчання та аналізу даних, а також достатню швидкість розробки вебзастосунків через високорівневі фреймворки.

Вибір Python для даного проєкту обумовлений передусім його універсальністю та швидкістю прототипування. Мова дозволяє легко

інтегрувати дані з вебсайтів та обробляти великі обсяги статистичної інформації про команди та гравців.

Структури даних високого рівня разом із динамічною семантикою та динамічним зв’язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. У 2023 році, найпопулярнішою мовою програмування серед IT-спеціалістів згідно рейтингу IEEE Spectrum стала Python. Рейтинг 2023 року охоплював 59 мов програмування [20].

Таблиця 2.1 – Порівняння мов програмування

Критерій порівняння	Python	JavaScript	Java	C#
Швидкість виконання коду	Середня	Висока	Висока	Висока
Зручність розробки	Висока	Середня	Середня	Середня
Швидкість розробки веб	Дуже висока	Висока, але фронт окремо	Середня	Середня
Інтеграція з БД	Легка	Легка	Легка	Легка
Доступність бібліотек для ML	Велика різноманітність	Середня різноманітність	Середня різноманітність, небагато прикладів	Середня різноманітність, небагато прикладів
Кросплатформність	Повна	Повна	Повна	Часткова
Підтримка розробниками	Висока	Висока	Висока	Висока
Підтримка асинхронності	Висока	Висока	Середня	Висока

У сучасних проєктах з ML, особливо коли йдеться про вебінтерфейс для прогнозування, важливо підібрати фреймворк, який буде максимально ефективним як для бекенду, так і для взаємодії з моделями. У Python-екосистемі одним із таких варіантів є Streamlit, який створений спеціально для дата-сайєнс додатків, він дозволяє швидко організувати вебінтерфейс без необхідності писати складний фронтенд. Разом із тим, класичні веб-фреймворки як Flask та Django широко використовуються у веброзробці і мають великі можливості для кастомізації, масштабування та інтеграції з базами даних. В таблиці 2.2 наведено порівняння цих фреймворків.

Таблиця 2.2 – Порівняння фреймворків

Критерій порівняння	Streamlit	Flask	Django
Зручність використання	Висока	Середня	Висока
Зручність розробки	Висока	Середня	Середня
Актуальність	Висока	Висока	Висока
Потреба в бекенді	Не потрібен	Потрібно створювати сервер самостійно	Не потрібен, але UI та ML інтеграцію потрібно налаштовувати
Можливості UI	Середні	Середні	Високі
Розгортання	Дуже просте	Середнє	Просте, добре масштабується

Streamlit виділяється серед Python-фреймворків своєю простотою та швидкістю прототипування. Він дозволяє швидко створити вебінтерфейс для ML-моделей без необхідності писати фронтенд-код або окремий бекенд [21].

Для проєктів, які потребують швидких інтерактивних прототипів прогнозування, це значна перевага.

Flask і Django підходять більше для традиційних вебзастосунків, де важлива кастомізація і масштабування. Вони сумісні з Python-бібліотеками, але інтеграція ML-моделей потребує додаткових налаштувань, шаблонів і серверного коду. Django пропонує багатий функціонал для вебсайтів, але це ускладнює швидке створення ML-прототипів.

Тому для задачі прогнозування результатів кіберспортивних матчів, де важлива швидка інтеграція статистики та моделей, Streamlit забезпечує найзручніший і найбільш ефективний підхід.

2.2 Визначення функціональних вимог

Після того як було визначено основні інструменти розробки ПЗ, було виділено функціональні вимоги яким має відповідати програма:

– користувач повинен мати можливість відкрити вебзастосунок і бачити його вміст;

– користувач повинен мати можливість обирати між відображенням матчів які йдуть у реальному часі та майбутніх матчів;

– користувач повинен мати можливість бачити матчі, що відбуваються у режимі реального часу;

– користувач повинен мати можливість бачити майбутні матчі протягом найближчого тижня;

– користувач повинен мати можливість переходити за гіперпосиланнями на сторінку матчу з hltv.org;

– користувач повинен мати можливість вибирати тип прогнозу: на весь матч або на конкретну карту (якщо вона відома);

– користувач повинен мати можливість робити прогноз на матч, коли карти ще не визначені;

– користувач повинен мати можливість робити прогноз на матч, коли карти визначені;

– користувач повинен мати можливість вибирати карту для прогнозу зі списку доступних;

– користувач повинен отримувати повідомлення про помилку, якщо намагається зробити прогноз на карту, коли вони ще невідомі;

– користувач повинен мати можливість обирати додаткові ML-моделі для прогнозу результату матчу;

– користувач повинен мати можливість бачити прогноз результату обраної ML-моделі у відсотках для обох команд;

– користувач повинен мати можливість бачити форму команд (останні 5 матчів) при прогнозі на матч;

– користувач повинен мати можливість бачити історію зустрічей команд (head-to-head) при прогнозі на матч;

– користувач повинен мати можливість бачити середній вік гравців команди, якщо він доступний на hltv.org;

– користувач повинен мати можливість бачити позицію команд у топі hltv.org;

– користувач повинен мати можливість бачити загальний відсоток перемог команд (для актуального складу команди);

– користувач повинен мати можливість бачити середні статистики гравців за різними параметрами;

– користувач повинен мати можливість бачити окремі таблиці зі статистикою гравців для обох команд;

– користувач повинен мати можливість бачити окремі таблиці з відсотком перемог команд по різних картах (для актуального складу команди);

– користувач повинен мати можливість шукати матчі за назвою команди;

– користувач повинен мати можливість скасувати фільтр пошуку, якщо текст у полі пошуку видалено;

– система повинна парсити та зберігати історичні матчі у БД, які будуть використані для основної та додаткових моделей, і для формування вхідних ознак команд (форма та head-to-head);

– система повинна парсити інформацію зі сторінки матчу, а саме показники гравців та статистичні данні команд;

– система повинна брати інформацію про матчі на які можливо зробити прогноз з бази, якщо дані актуальні, або якщо парсинг неможливий;

– система повинна завантажувати статистичні дані гравців та команд у відповідні таблиці в БД та оновлювати їх, якщо минуло більше 24 годин з моменту останнього запису або нової спроби прогнозу;

– система повинна завантажувати збережені моделі при запуску;

– система повинна формувати вхідні ознаки для моделей на основі історії команд, їх форми та статистики гравців;

– система повинна робити прогноз на весь матч та на конкретну карту за наявності даних;

– система повинна відображати відсоток команд по вибраній карті при прогнозі на карту;

– система не повинна враховувати форму команд та h2h при прогнозі на карту;

– система повинна зберігати прогнози у БД окремо для матчів та карт;

– система повинна відображати дату та час матчу поруч із назвою матчу;

– система повинна зберігати та оновлювати моделі після навчання на історичних даних;

– система повинна мати можливість донавчати моделі після поповнення вибірки історичних матчів;

– система повинна видавати повідомлення про помилки у разі некоректного введення або відсутності даних;

– система повинна відображати результати прогнозів додаткових моделей для порівняння з основною моделлю.

На основі зазначених функціональних вимог були сформовані й описані можливі сценарії взаємодії користувача з програмним забезпеченням, що відображено на діаграмі прецедентів (рис. 2.1).

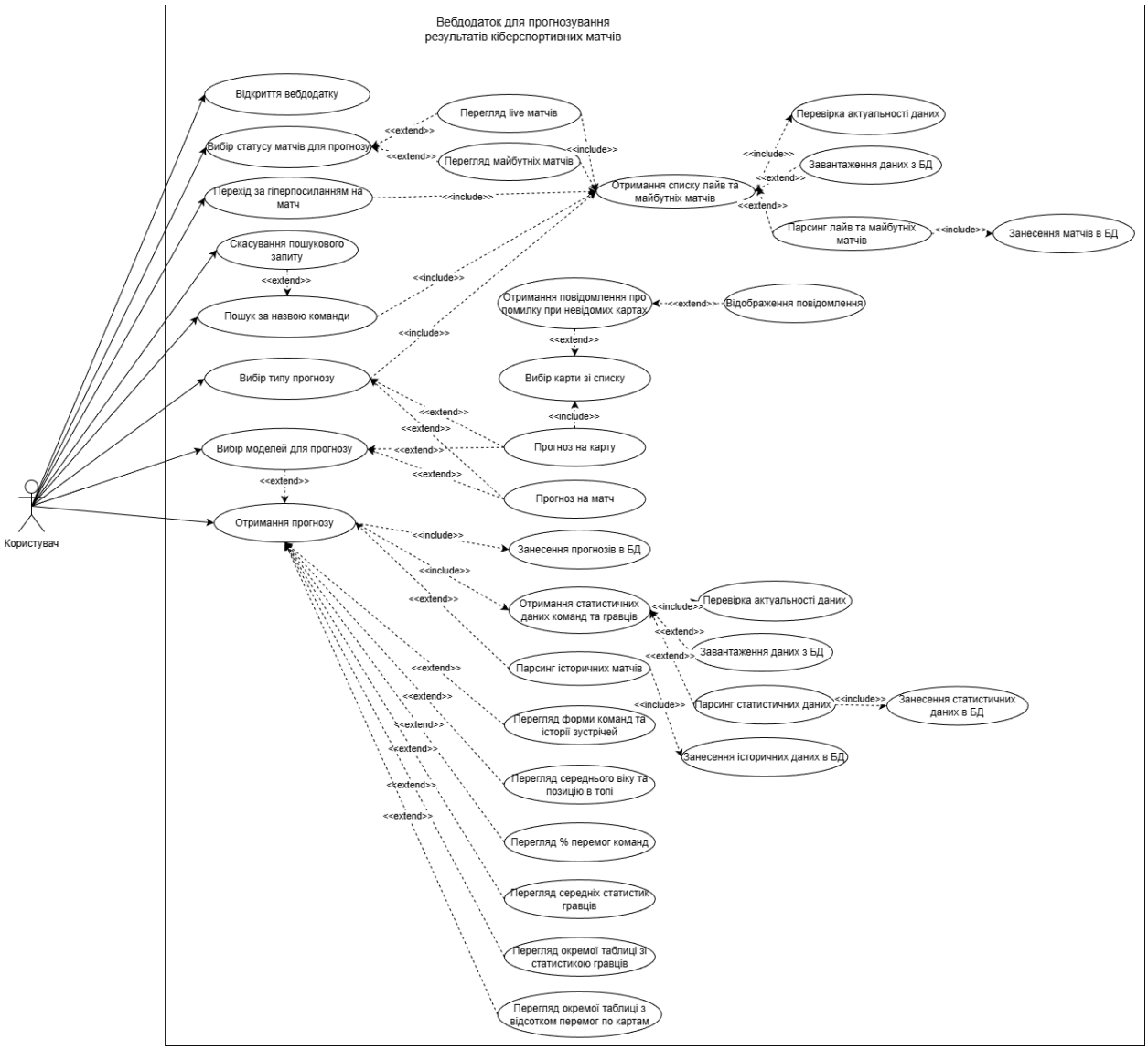


Рисунок 2.1 – Діаграма взаємодії користувача і вебзастосунку

2.3 Вибір СКБД та проєктування

Після формування функціональних вимог та побудови діаграми прецедентів постає задача вибору системи керування базами даних, яка забезпечить коректне зберігання та обробку інформації. У межах проєкту передбачається робота з декількома типами даних: історичними результатами матчів, статистикою гравців і команд, поточними показниками, а також прогнозами, що формуються під час обчислень. Частина цих даних оновлюється рідко (наприклад, історичні матчі), інша – регулярно (поточна форма команд, статистика гравців, список поточних матчів). Це вимагає від СКБД надійного механізму транзакцій, можливостей для виконання складних вибірок із фільтрацією та агрегацією, а також ефективної роботи з великими обсягами даних у режимі переважання читання над записом.

Система також повинна підтримувати зберігання структурованих даних (таблиці матчів, гравців, турнірів), а в подальшому – можливість роботи з напівструктурованими даними, наприклад з JSON-відповідями при парсингу поточної статистики. Важливим є і питання масштабованості: із розширенням покриття (додаткові турніри, десятки тисяч матчів) БД має залишатися продуктивною при обробці складних SELECT-запитів, зокрема запитів із JOIN та агрегаціями.

Для вирішення цих задач можливе використання як реляційних СКБД, так і представників NoSQL-підходу. Реляційні бази (MySQL, MariaDB, PostgreSQL) забезпечують чітку структурованість, цілісність даних та ефективну роботу зі складними аналітичними запитами. NoSQL-рішення, зокрема MongoDB, більше орієнтовані на зберігання JSON-подібних структур, швидке горизонтальне масштабування та роботу з нерегулярними або динамічними даними. Однак у рамках даного проєкту дані мають стабільну структуру й використовуються переважно у вигляді таблиць, а не вкладених документів. Тому застосування NoSQL не дало б суттєвих переваг, тоді як реляційний підхід спрощує організацію, пошук і подальшу аналітичну

обробку даних. У табл. 2.3 наведено порівняння основних СКБД, які можуть бути використані для реалізації цих вимог.

Таблиця 2.3 – Порівняння PostgreSQL, MariaDB, MongoDB

Параметр	MongoDB	MariaDB/MySQL	PostgreSQL
Підхід	Документована NoSQL	Реляційна	Реляційна
Модель даних	Документи JSON, гнучка структура	Таблична модель, оптимізована для CRUD	Чіткі схеми, сильні JOIN
Ефективність для складних SELECT-запитів	Обмежена	Середня	Висока
Робота з напівструктурованими даними	Основний формат	Обмежена	Підтримка JSONB та індексації
Продуктивність	Швидка з невеликими обсягами даних	Швидка	Зменшується при великій кількості процесів
Масштабованість	Горизонтальна	Вертикальна	Масштабується вертикально та горизонтально
Транзакції	Обмежені (на рівні документа)	ACID	Підтримує ACID, реплікацію
Придатність до ML/аналітики	Гнучка структура даних, підходить для швидкої агрегації	Середня	Добра інтеграція з Python, великі набори функцій
Застосування в проєкті	У рамках цього проєкту не дає суттєвої переваги через стабільну структуру таблиць.	Можна використовувати, але гірше працює з аналітикою	Оптимально для історичних даних, складних вибірок, прогнозів

У рамках проєкту було обрано PostgreSQL, оскільки ця СКБД поєднує високу продуктивність при виконанні складних аналітичних операцій, підтримку транзакцій та розширену роботу з JSON-даними, що є оптимальним для змішаного характеру інформації, яку необхідно обробляти в системі. PostgreSQL має тип JSONB, який зберігає JSON в декомпонованому бінарному форматі і підтримує індексацію (наприклад, GIN-індекси), що робить запити по JSON-полям ефективнішими [22]. Це дає змогу зберігати дані зі статистики гравців або матчів у JSON-подібному форматі, але при цьому швидко виконувати фільтрацію й агрегацію за окремими полями. PostgreSQL використовує MVCC (Multi-Version Concurrency Control), що дозволяє одночасно читати та записувати дані без блокувань, забезпечуючи консистентність і високу продуктивність при паралельних транзакціях [23]. Якщо система буде активно оновлювати статистику або зберігати нові прогнози, PostgreSQL гарантовано витримає навантаження і забезпечить коректну роботу з даними.

Після визначення СКБД наступним кроком було спроєктувати структуру бази даних, яка забезпечить інтеграцію з ПЗ, зберігання всіх необхідних даних та їх подальший запис і обробку.

Було створені головні таблиці в базі даних для поповнення і зберігання потрібної інформації.

Таблиця matches включає поля:

– id (integer) – первинний ключ;

– date (date) – дата проведення матчу;

– team1 (text) – назва першої команди;

– team2 (text) – назва другої команди;

– match_format (text) – формат матчу;

– score1 (integer) – рахунок першої команди;

– score (integer) – рахунок другої команди;

– tournament (text) – назва турніру.

Таблиця необхідна для зберігання історичних даних матчів, які використовуються для розрахунку форми команд, аналізу історії зустрічей (head-to-head) та навчання додаткових ML-моделей.

Таблиця players_stats:

– hltv_id (integer) – первинний ключ, унікальний ідентифікатор гравця на hltv.org;

– nickname (varchar(50)) – нікнейм гравця;

– rating (float) – рейтинг гравця який відображає наскільки показники гравця вище чи нижче середнього, де 1.00 – це середнє значення;

– round_swing (float) – показник, який показує наскільки в середньому гравець змінив шанси своєї команди на перемогу в раунді, залежно від економіки команди, сторони, вбивств смертей, урону;

– dpr (float) – показник смертей за раунд;

– kast (float) – відсоток раундів, у яких гравець вбив, допоміг у вбивстві, вижив або був розміняний після контакту з супротивником;

– multi_kill (float) – відсоток раундів, у яких гравець зробив 2 або більше вбивств;

– adr (float) – середній урон за раунд;

– kpr (float) – вбивста за раунд;

– last_update (timestamp) – дата останнього оновлення даних;

– team_id (integer) – зовнішній ключ, зв’язок з таблицею teams (зв’язок один до багатьох, одна команда може мати багато гравців), id поточної команди гравця.

Таблиця використовується як кеш статистики гравців для швидкого отримання даних під час прогнозування. Якщо last_update перевищує 24 години, система парсить актуальні дані і оновлює таблицю.

Таблиця teams включає такі поля:

– hltv_id (integer) – первинний ключ, ідентифікатор команди на hltv.org;

– name (text) – назва команди;

– world_rank (integer) – поточна позиція у світовому рейтингу;

– avg_age (double) – середній вік гравців команди;

– maps_wr (json) – статистика % перемог команди по картах;

– lineup_wr (real) – % перемог актуального складу команди;

– last_update (timestamp) – дата останнього оновлення даних.

Таблиця містить актуальні дані про команди, які оновлюються парсером при зміні інформації або якщо last_update перевищує 24 години.

Таблиця match_predictions:

– id (integer) – первинний ключ;

– match_id (text) – унікальний ідентифікатор матчу, один прогноз на кожен матч;

– team1 (text) – назва першої команди;

– team2 (text) – назва другої команди;

– predicted_prob1 (numeric) – ймовірність перемоги першої команди;

– predicted_prob2 (numeric) – ймовірність перемоги другої команди;

– predicted_prob1_map_avg (numeric) – ймовірність перемоги першої команди якщо карти на матч відомі, інакше NULL;

– predicted_prob2_map_avg (numeric) – ймовірність перемоги другої команди якщо карти на матч відомі, інакше NULL;

– actual_winner (text) – реальний переможець матчу;

– predicted_winner (text) – передбачений переможець моделлю;

– prediction_date (timestamp) – дата створення прогнозу.

Таблиця потрібна для зберігання прогнозів матчів моделі та подальшого аналізу їх точності.

Таблиця map_predictions:

– id (integer) – первинний ключ;

– match_id (text) – унікальний ідентифікатор матчу, один прогноз на кожен матч;

– map_name (text) – назва карти, має унікальне обмеження з match_id, щоб не було дублювання прогнозів для однієї карти конкретного матчу;

– team1 (text) – назва першої команди;

– team2 (text) – назва другої команди;

– predicted_prob1 (numeric) – ймовірність перемоги першої команди на карті;

– predicted_prob2 (numeric) – ймовірність перемоги другої команди на карті;

– actual_winner (text) – реальний переможець карти;

– predicted_winner (text) – передбачений переможець моделлю;

– prediction_date (timestamp) – дата створення прогнозу.

Таблиця потрібна для зберігання прогнозів моделі на конкретну карту та подальшого аналізу їх точності.

Таблиця matches_cache:

– id (integer) – первинний ключ;

– live (json) – список лайв матчів;

– upcoming (json) – список майбутніх матчів;

– last_update (timestamp) – дата оновлення матчів для прогнозу.

Таблиця потрібна для зберігання завантажених матчів на які користувач може отримати прогноз, потрібна на той випадок коли витягувати інформацію напряму зі сторінки неможливо.

Схему бази даних можна побачити на рис. 2.2. Також для того, щоб цю базу даних можна було використовувати не лише на локальній адресі, а й після деплою через Streamlit Cloud, цю базу потрібно загрузити наприклад в “хмару” для цього обрано платформу Railway, з мінусів цієї платформи що дається 1 місяць пробного користування – далі потрібно платити гроші щоб база продовжувала існування в хмарі (5 доларів на місяць).

Railway підтримує різні типи баз даних, які можна розгортати як сервіси в проєкті. Дані сервісі розгортаються в контейнерах (Docker), з підключеними томами для зберігання даних (Volumes). Можна використовувати TCP-проксі, щоб підключатися до баз даних ззовні частини приватної мережі проєкту [24].

Railway має шаблони для популярних баз: наприклад, Postgres.

Завдяки шаблонам й готовим конфігураціям, можна швидко підняти базу даних: не треба вручну налаштовувати сервер – все “в коробці” [25].

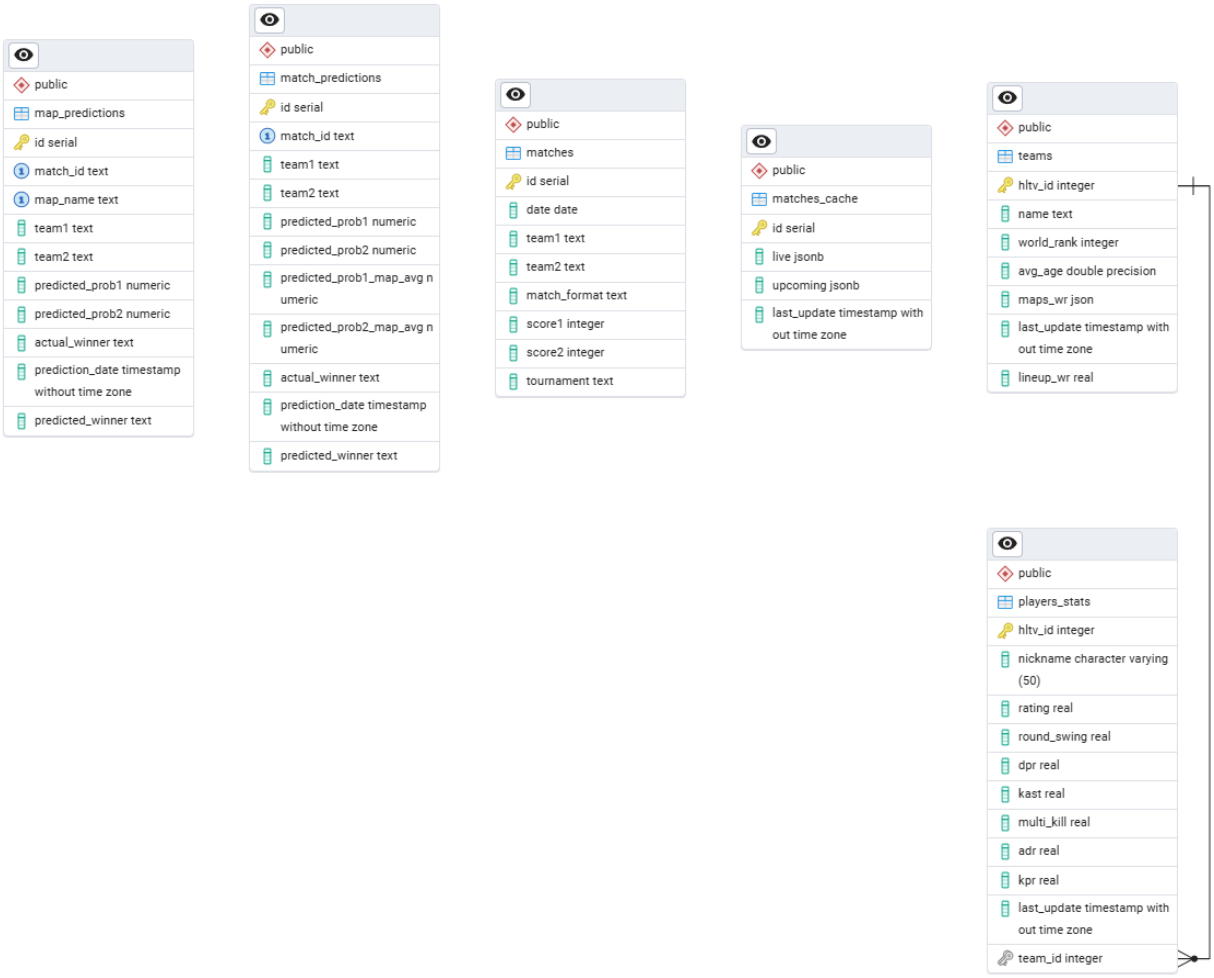


Рисунок 2.2 – Схема бази даних

2.4 Структура системи

За задумом користувач повинен відкрити вебзастосунок, дочекатися завантаження матчів що йдуть зараз та майбутніх матчів на тиждень уперед. Він повинен мати можливість оновлювати список цих матчів, переходити між категоріями поточних та майбутніх матчів, обирати за допомогою якої моделі він хоче отримати прогноз, обирати тип прогнозу: на весь матч, або на конкретну карту. Також він повинен мати можливість переходити на

сторінку матчу на сайті hltv.org, бачити дату та час матчу біля цього посилання. Після отримання прогнозу користувач повинен бачити ймовірності перемоги кожної з двох команд та їх статистичні дані, а саме: поточну форму (5 останніх матчів), історію зустрічей між двома командами (якщо вона наявна), середній вік команд (якщо на hltv.org вказаний цей вік), позицію в топі, загальний % перемог, % перемог по вказаній карті, якщо вибраний прогноз по карті, середні показники по командах та таблиці % перемог по усіх картах та окремо статистичні дані по кожному гравцю.

Таким чином, необхідно розробити інтерфейс вебзастосунку, створити базу даних з необхідними таблицями, зробити парсер для історичних матчів якій можна буде запускати та поповнювати базу історичних матчів. Планується створити окремі файли: main.py для взаємодії з користувачем, відображення лайв та майбутніх матчів, отримання прогнозів і демонстрації статистики. Models_train.py планується реалізувати для тренування та оцінки кількох сторонніх ML-моделей на основі історичних даних: Logistic Regression, XGBoost та CatBoost. Після навчання моделі зберігатимуться у вигляді .pkl файлів разом із допоміжними об’єктами – LabelEncoder, історією команд, H2H та різницею по картах – для подальшого використання в інтерфейсі вебзастосунку. Model.py – для функцій обчислення форми команд, організації історії зустрічей, парсингу та запису нових даних у базу, а також формування прогнозів на матчі та карти.

В model.py планується реалізувати власну модель, яка буде працювати з максимально актуальною інформацією про команди та гравців. Через відсутність офіційного API hltv.org та захист Cloudflare, отримання цих даних можливе лише поступово, через це сторонні моделі будуть тренуватися виключно на історичних даних – форма команд, історія зустрічей та рахунки з бази – що дозволить оцінити їх точність та порівняти ефективність без потреби в реальному часі.

Розроблену структуру проєкту можна побачити на рис. 2.3.

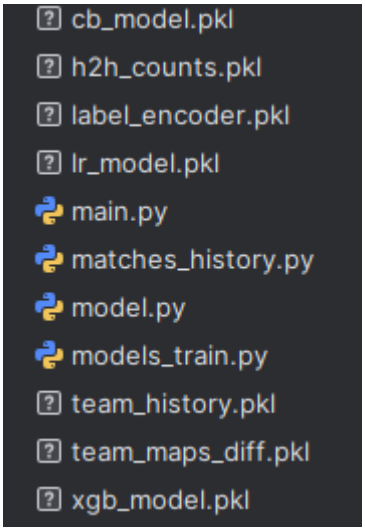


Рисунок 2.3 – Структура проєкту

2.5 Висновки за розділом 2

В цьому розділі визначено основний інструментарій, необхідний для розробки ПЗ. Проведено аналіз можливих мов програмування та бібліотек, обґрунтовано вибір Python як основного інструменту для реалізації вебзастосунку та роботи з ML-моделями, зокрема використання Streamlit для створення інтерфейсу та відображення лайв та майбутніх матчів, а також демонстрації прогнозів і статистики.

Визначено функціональні вимоги до системи, спроєктовано взаємодію користувача з ПЗ у вигляді діаграми прецедентів. Зважаючи на характер даних та потребу у їх структурованому зберіганні, проведено порівняння СКБД і обрано PostgreSQL.

Продумано загальну архітектуру проєкту, включаючи обробку історичних даних, статистики команд та гравців, а також можливість використання і порівняння різних ML-моделей для прогнозування результатів матчів. Прийнято рішення спробувати реалізувати власну модель, яка буде враховувати саме актуальні статистичні дані по гравцям та командам.

3 ОСНОВНІ РІШЕННЯ РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Алгоритм функціонування програми

За структурною схемою, описаною в розділі 2, було створено схему алгоритму функціонування програми, її можна побачити на рис. 3.1.

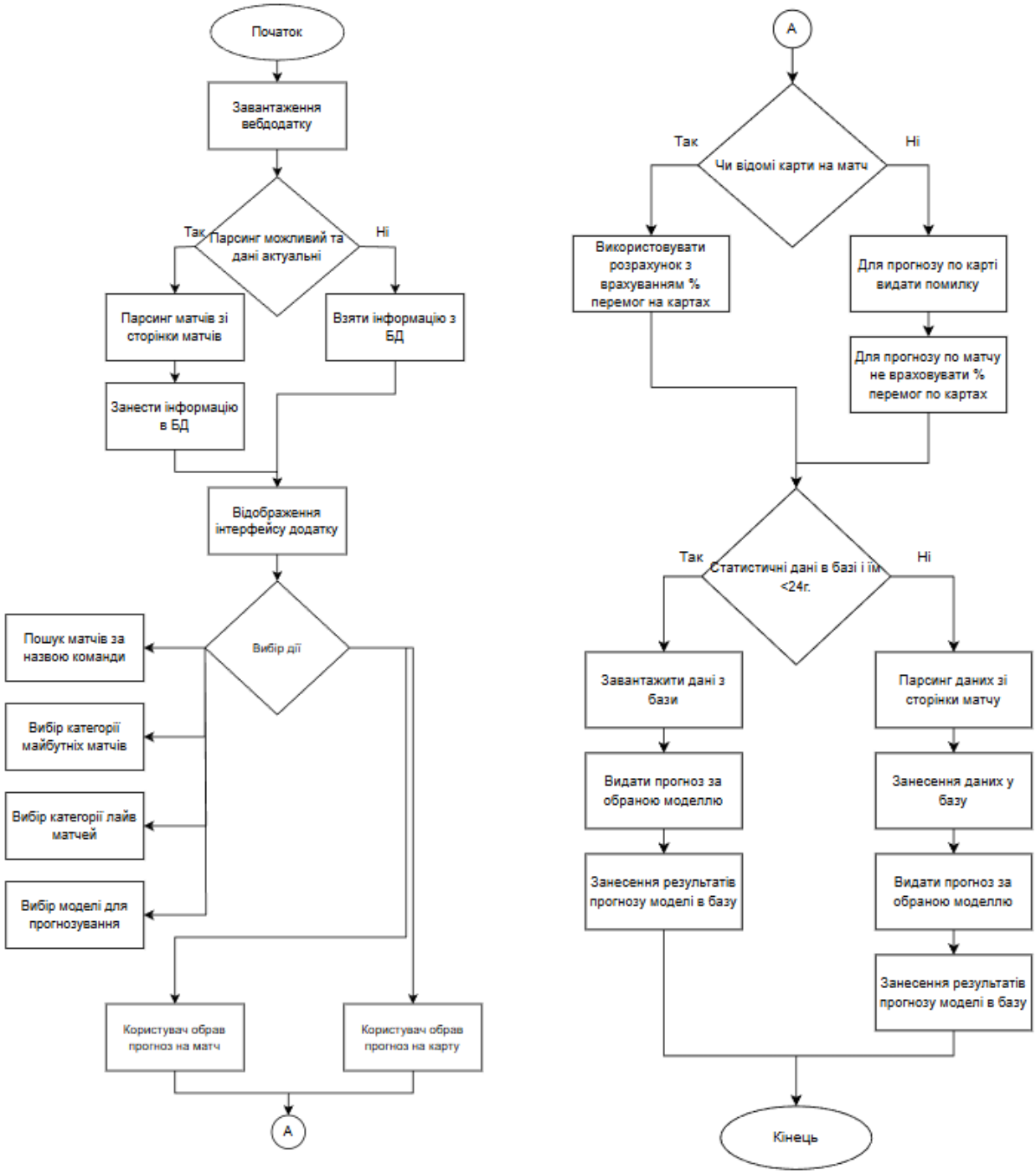


Рисунок 3.1 – Схема алгоритму програми

Розроблена схема алгоритму полегшує створення ПЗ, оскільки дозволяє наочно зрозуміти логіку програми, її структуру та потоки даних між компонентами. Схема також дозволяє оцінити ефективність алгоритму, виявити зайві або дублюючі кроки та оптимізувати логіку ще до початку програмування.

3.2 Опис розроблених компонентів

Компонент matches_history.py відповідальний за автоматизоване збирання, оновлення та підтримку історичних даних про матчі CS2. Його основна задача – виконувати роль парсера, який отримує результати матчів із сайту hltv.org, опрацьовує їх, очищує, структурує та зберігає в базу даних PostgreSQL. Ця історична вибірка використовується як навчальний датасет для моделей прогнозування, тому своєчасне оновлення даних є критично важливим. Компонент забезпечує повне формування бази даних, якщо вона порожня: у такому випадку файл послідовно обходить усі сторінки результатів hltv.org, збираючи повну історію матчів та інкрементальне оновлення: якщо база вже існує, скрипт автоматично визначає останній збережений матч і підвантажує лише нові, уникаючи дублювання. Розглянемо методи які присутні в цього компонента:

– clean_text(text: str) – очищає текстові дані (транслітерує символи та прибирає зайві пробіли для того щоб не зламати кодування у БД), використовується для назв команд та турнірів;

– parse_date(date_text: str) – переводить дату з hltv.org у формат date, забезпечує коректне порівняння та сортування матчів при оновленні бази;

– fetch_matches_from_page(offset) – offset це зміщення сторінок, тобто наприклад перша сторінка результатів матчів має offset 0, друга 100, тому що на сторінці по 100 матчів, і так далі, виконує запит, перевіряє наявність блоків матчів і виконує повторні запити при збої;

– parse_matches(blocks, stop_on_last=True) – blocks – HTML-блоки з матчами, stop_on_last – прапорець зупинки при знаходженні останнього матчу з БД, аналізує блоки та витягує інформацію про кожен матч (зчитує назви команд, турнір, рахунок, формат матчу та дату), порівнює з останнім записом у БД, формує список нових матчів у порядку від нових до старих (більш старі матчі з меншими індексами).

На даний момент парсер збирає інформацію за 2 роки (з моменту оновлення CS:GO до CS2), тобто це близько 20 тисяч матчів.

Компонент model.py є центральним компонентом системи прогнозування результатів кіберспортивних матчів. Він об’єднує набір аналітичних функцій, які працюють на основі історичних даних із бази даних, актуальної статистики гравців та команд отриманої з hltv.org.

У ньому зібрано повний набір методів, що відповідають за підготовку статистики, попередню обробку даних і формування ключових метрик, необхідних для побудови прогнозів. Файл поєднує роботу з історичними результатами команд, аналітикою гравців, внутрішніми коефіцієнтами моделі та системою масштабування, забезпечуючи стабільні, узгоджені та коректно підготовлені вхідні дані.

Модуль model.py використовується як ключовий інструмент для формування об’єктивних метрик, які надалі подаються у модель машинного навчання або комбінуються в аналітичні формули прогнозування. Його функції дозволяють отримати структуровану статистику в реальному часі, уникати дублювання запитів, використовувати кешовані дані, а також забезпечувати правильність та стабільність розрахунків.

Цей компонент реалізує такі методи:

– head_to_head(team1, team2, cur) – обчислює відсоток перемог команди в очних зустрічах (з таблиці matches витягуються усі матчі між team1 та team2, рахується кількість перемог team1, повертається частка перемог у відсотках), використовується як фактор переваги однієї команди

над іншою, cur це курсор бази даних (об’єкт через який виконуються SQL-запити), він використовується всюди де йде робота з БД;

– team_form(team_name, cur, last_n=5) – визначає форму команду за останні 5 матчів (отримує останні матчі за team_name, формує список з 1 (перемога) та 0 (поразка), повертає у порядку від найсвіжішого матчу до старішого), є метрикою поточного стану команди, необхідна для моделі прогнозування;

– team_form_string(team_name_cur, last_n=5) – формує html-представлення форми команди (перетворює список отриманий після відпрацювання team_form з нулів та одиниць на червону літеру “L” та зелену літеру “W”, потрібен для візуального відображення форми у вебінтерфейсі;

– head_to_head_scores(team1, team2, cur) – повертає повну інформацію про очні зустрічі (витягує всі матчі між командами, рахує перемоги обох сторін, формує список усіх очних результатів із турнірами на яких відбулися матчі, повертає підсумковий рядок та список матчів), потрібен для аналізу head-to-head та відображення в інтерфейсі;

– clean_text(text: str) – нормалізує й очищує текст, запобігає проблемам при занесенні назв команд та нікнеймів гравців в БД;

– team_player_stats(team_name, match_link, cur) – виконує початкове очищення нікнейму гравця, щоб уникнути помилок у SQL-запитах, отримує статистику гравців команди за матч (Завантажує сторінку матчу з hltv.org, знаходить склад команди, для кожного гравця витягує hltv_id, нікнейм, перевіряє чи є статистика для цього гравця в БД, якщо дані актуальні, або парсинг неможливий – бере з БД, інакше – парсить сторінку статистики цього гравця. Збирає метрики rating, round_swing, dpr, kast, multi_kill, adr, kpr. Оновлює або додає дані в таблицю players_stats, повертає список словників зі статистикою гравців). Це критично важлива функція для моделі, оскільки вона дозволяє використовувати актуальні персональні показники гравців, що суттєво підвищує точність прогнозування матчів. Вона також оптимізує запити за рахунок кешування, що зменшує навантаження на hltv.org;

– team_match_stats (team_name, match_link, cur) – виконує початкове очищення назви команди, щоб уникнути помилок у SQL-запитах, перевіряє локальний кеш у таблиці teams, якщо дані зберігались не більше 24 годин та містять усі ключові поля, функція повертає їх без парсингу, якщо даних немає або вони застарілі завантажує html-сторінку матчу. Визначає блок команди на сторінці, знаходить її id з посилання матчу hltv.org, завантажує профіль команди й отримує назву команди, рейтинг в топі, середній вік гравців, аналізує блоки статистики по певних картах і формує структуру % перемог. Визначає поточний склад команди, формує URL для статистики цього складу і отримує його загальний % перемог. Зберігає всі зібрані параметри у таблицю teams, повертає структурований словник з повною актуальною статистикою;

– predict_map() – дає прогноз для окремої карти (функція обчислює ймовірність перемоги кожної з команд на конкретній карті, використовуючи сукупність індивідуальних та командних статистичних параметрів), використовується функція weighted_metrics(players_stats), завантажує та використовує % перемог команди на конкретній карті, загальний % перемог складу, позицію в топі та середній вік (функції rank_factor та age_factor), формує підсумковий score у ймовірності (від 0 до 100%), записує прогноз у таблицю map_predictions, функція дозволяє отримати прогноз на окремій карті, що є критично важливим для матчів де більше одної карти в матчі, де результат сильно залежить від піку та історичної сили команд на мапі;

– weighted_metrics(players_stats) – зважена оцінка гравців, використовується в predict_map(), формує агреговані статистичні показники команди на основі індивідуальних метрик гравців (сортує гравців за рейтингом, надає певні ваги для п’яти гравців команди, формує зведені метрики гравця), дозволяє моделі враховувати реальну силу складу, підкреслюючи вплив ключових гравців;

– rank_factor(rank1, rank2) – фактор світового рейтингу, обчислює додатковий коефіцієнт сили команди, залежно від місця у світовому рейтингу

hltv.org (порівнює позиції команд, визначає ступінь “розриву сили”, нараховує бонус сильнішій команді зі збільшенням при значній різниці у рейтингу), враховує макрорівень сили команд, який не завжди видно в індивідуальних метриках. Також індивідуальні метрики можуть бути не дуже об’єктивними тому що наприклад гравців “набили” високі цифри на більш слабких суперниках;

– age_factor(age) – віковий коефіцієнт, оцінює відповідність середнього віку складу піковому значенню віка гравця, дає змогу моделі враховувати, чи команда ближче до “піку форми”, якщо на сторінці команди середній вік не вказаний, такій команді дається фіксоване стандартне значення в 80 пунктів (ідеал – 100);

– predict_match() – загальний прогноз матчу, формує прогноз результату матчу, використовуючи інформацію про всі мапи або виконує прогнозування враховуючи лише інші наявні статистичні дані (форма, head-to-head, загальний % перемог команд, позиція в топі, середній вік, статистичні дані гравців), якщо мапи відсутні викликає predict_map без мапи, коригує результат за формою та head-to-head, якщо мапи є виконує прогноз для кожної мапи і усереднює ймовірності з урахуванням ваги карти, зберігає результат у таблицю match_predictions. Наприкінці функції ймовірності коригуються з урахуванням форми та head-to-head (щоб поєднати мікро-статистику (гравці/мапи) з макро-статистикою (форма, h2h).

Компонент models_train.py призначений для побудови, навчання та тестування моделей машинного навчання, які прогнозують результати кіберспортивних матчів на основі історичних даних. Він виконує комплексне опрацювання даних із бази PostgreSQL, включно з сортуванням матчів за датою, формуванням цільової змінної (winner), кодуванням команд через LabelEncoder, а також обчисленням ключових ознак, таких як форма команди (серія останніх результатів), результати очних зустрічей (H2H) та різницю по картах (рахуються різниця між виграними та програними картами). Файл дозволяє тренувати кілька моделей (логістична регресія, XGBoost, CatBoost),

оцінювати їх точність, а також зберігати моделі й допоміжні об’єкти для подальшого використання у прогнозах. Крім того, він надає функцію для швидкого прогнозу результату конкретного матчу з урахуванням історії команд та розрахованих ознак.

В ньому є такі функції:

– predict_from_matches(team1_name, team2_name) – функція виконує прогноз результату матчу між двома командами на основі збережених моделей і попередньо обчислених ознак (перевіряє наявність команд у LabelEncoder, кодує команди у числові значення для моделей, витягує останню форму команд та результати очних зустрічей, формує DataFrame з ознаками, проганяє ознаки через всі моделі та повертає прогнозованого переможця та ймовірність його перемоги);

– get_winner(score1, score2) – визначає переможця матчу з історичної таблиці у БД, score1 – кількість виграних мап першою командою, score2 – другою, використовується для формування цільової змінної winner в DataFrame;

– форма команди обчислюється на основі результатів останніх 5 матчів: кожна перемога додає +1, поразка -1. Середнє цих значень дає числову характеристику поточної форми команди, яка використовується як ознака для моделей;

– результати очних зустрічей (H2H) формуються за допомогою словника h2h_counts, де для кожної пари команд зберігається кількість перемог кожної команди проти іншої. Різниця цих значень використовується як ознака h2h, що показує перевагу однієї команди над іншою в історії їхніх матчів;

– сумарна різниця карт (team_maps_diff) обчислюється для кожної команди як сума різниці виграних і програних карт у всіх попередніх матчах.

Це дозволяє моделі враховувати не лише перемогу чи поразку, а й наскільки впевнено команда вигравала матчі.

В цьому модулі формуються ознаки для моделі – x, цільова – y (1 – перемога першої команди, 0 – другої), моделі навчаються, 80% даних – навчання, 20% – тестування.

Логістична регресія – базова лінійна модель для класифікації, їй ставиться максимальна кількість ітерацій в 2000.

XGBoost – градієнтний бустинг дерев рішень, добре працює з табличними даними.

CatBoost – бустингова модель, оптимізована для категоріальних ознак, виставлено 300 ітерацій.

Для кожної моделі розраховується точність, функція втрат, та середнє квадратичне відхилення.

Останній модуль це main.py, він поєднує отримання даних з hltv.org, обробку інформації про команди та гравців, треновані моделі для прогнозування результатів матчів і карт, а також візуалізацію цієї інформації через Streamlit.

В ньому присутні такі функції:

– safe_load(path) – перевіряє чи існує файл, якщо існує завантажує його і повертає об’єкт, якщо файлу немає – повертає none, використовується для безпечного завантаження моделей та даних без помилок;

– get_matches() – завантажує сторінки з лайв та майбутніми матчами на тиждень уперед, парсить html та отримує інформацію про турнір, команди, посилання на матч, карти та їх статус (upcoming/played), дату та час матчу, для кожного матчу формує словник з усією інформацією, повертає список матчів для подальшої обробки і відображення, зберігає цю інформацію в БД для швидкого доступу, або для доступу коли парсинг неможливий;

– get_matches_cached(force_reload=False) – зберігає список матчів для швидкого доступу без повторного завантаження з hltv.org, повертає cached матчі з БД, або оновляє їх, якщо потрібно;

– build_model_features(team1, team2) – перевіряє чи є команди у LabelEncoder, перетворює назви команд у числові коди, обчислює середню

форму команд, різницю head-to-head, різницю по вигранним картам, формує датафрейм з ознаками для моделей щоб зробити прогноз;

– aggregate_stats(stats_list) – приймає список словників з параметрами гравців, обчислює середні або медіанні показники гравців команди, повертає агрегований словник з цими показниками для візуалізації;

– show_players_table(team_name, stats_list) – формує таблицю з ключовими показниками гравців, виводить її на інтерфейс, дозволяє користувача переглядати детальну статистику гравців кожної з команд.

Тобто в цьому модулі йде завантаження моделей та даних, йде ініціалізація інтерфейсу Streamlit, йде підключення до БД, інтерфейс відрисовується, після прогнозу йде відображення статистик та результатів моделей.

3.3 Огляд власної моделі

Розглянемо детальніше які ваги та фактори використано у моделі для прогнозу та чому.

Вплив найкращих гравців часто непропорційно більший: лідер може вирішувати ключові раунди, диктувати темп. Тому замість простого середнього ми робимо зважене середнє, де перші два гравці отримують суттєво більші ваги.

Рейтинг є ключовою метрикою у гравців це агрегована метрика на сайті hltv.org, він враховує велику кількість факторів, в програмі беруться 5 гравців з одної команди та сортуються за спаданням рейтингу і з іншої команди, для цих рейтингів встановлені певні ваги, а саме: [0.5, 0.3, 0.1, 0.07, 0.03], тобто перші гравці (найкращі гравці з двох команд) отримують 50% впливу, другі гравці 30% і т.д., це відображає реальну ситуацію коли найкращі гравці часто вирішують результат більшості раундів, це часто зумовлено роллю гравця в команді, частіше це гравці які мають роль

снайпера або “ентрі-фрагера” (гравець який повинен першим зайти на точку та спробувати знайти перевагу для своєї команди).

Окрім зваженого середнього також зберігається best_rating (рейтинг найкращого гравця), йому дається додаткове мультиплікативне значення у фінальному score (фінальне значення за яким розраховуються відсотки перемоги кожної з команд). Це робиться для того що, зірковий гравець може одноосібно виграти важливий раунд і це може вплинути на загальний результат карти або матчу.

Якщо для гравця відсутні дані метрик, в метрику підставляється консервативне значення rating = 1.0, а сумарна вага нормалізується або залишається незмінною при використанні тільки наявних гравців.

Метрики мають різні шкали: rating може коливатися приблизно від 0.5 до 2, adr приблизно від 50 до 150, round_swing приблизно від -1.5 до 4 тощо. Якщо сумувати їх напряму домінують ті, що мають більші абсолютні значення. В програмі застосовано множники, тобто ручне масштабування метрик до спільного порядку величин, наприклад rating * 60, round_swing * 50, dpr * 50, kpr * 50. Це дозволяє привести вклад кожної метрики до сумірного внеску у фінальний результат, ці значення підібрані так, щоб середній внесок усіх цих метрик мав порівнянний порядок.

Ваги у фінальному складанні визначені таким чином:

– rating: 0.3 – один з найважливіших індикаторів, отримає найбільшу вагу;

– round_swing: 0.2 – важливий показник, відображає вплив гравця на зміну шансу виграти раунд;

– map_wr: 0.12 – команда може погано зробити пік/бан карт і потрапити на сильну карту супротивника, навіть якщо команда слабіше по іншим метрикам вона може бути більше підготовленою на певній карті;

– team_wr: 0.1 – загальний % перемог актуального составу демонструє здатність перемагати матчі в цілому, а відповідно й різноманітні карти;

– rank: 0.2 – дуже важливий показник який демонструє те наскільки нещодавні результати команди хороші, або не дуже, чим вище команда в топі тим більше очок (виграних матчів/турнірів) вона має, відповідно команда вище рейтингом виграє частіше, грає на більш престижних турнірах і перемагає більш сильні колективи;

– avg_age: 0.01 – невеликий вплив, дає тонкий ефект;

– dpr: 0.03;

– multi_kill: 0.03;

– adr: 0.03;

– kpr: 0.03;

– kast: 0.06 – не залежить від ролі гравця, загальний вплив трохи вищий.

Метрики dpr, multi_kill, adr, kpr частіше залежать від ролі гравця, якщо гравець рідше бачить супротивників на карті то в нього ці показники будуть менше, тому вирішено не давати великі ваги.

Великий розрив у рангах означає суттєву різницю між командами.

Базові величини base1 = 120 / log(rank1+1) дозволяє отримати спадну функцію від рангу (чим кращий ранг – більша величина). Далі вводиться фактор, що підсилює бонус для кращої команди пропорційно розриву gap = rank2 - rank1. Використане підвищення factor = 1 + (gap/10)**1.5 – нелінійне зростання впливу при великих розривах (наприклад, gap=25 – помітний бонус). Якщо ранг відсутній, тоді цієї команди ще немає в рейтингу і відповідно їй дається значення 0 тому що вона ще нічого не виграла і не грала ні з ким.

Досвід і реакція балансується, в програмі вказано оптимальний середній вік складу, а саме – 24.25, відхилення від якого знижує ефективність. На рисунку 3.2 продемонстровано графік середнього віку для команд з топ-20 рейтингу hltv.org.

Вік не є ключовим фактором, але він також врахований в підсумку прогнозу.

Для прогнозу по картах не враховується форма та head-to-head тому що через обмеження доступу до даних, неможливо було отримати повну історію для колективів по конкретних мапах, і відповідно це можна застосовувати лише для прогнозу по матчу. Форма і head-to-head – це фактори, що не завжди корелюють з технічною силою.

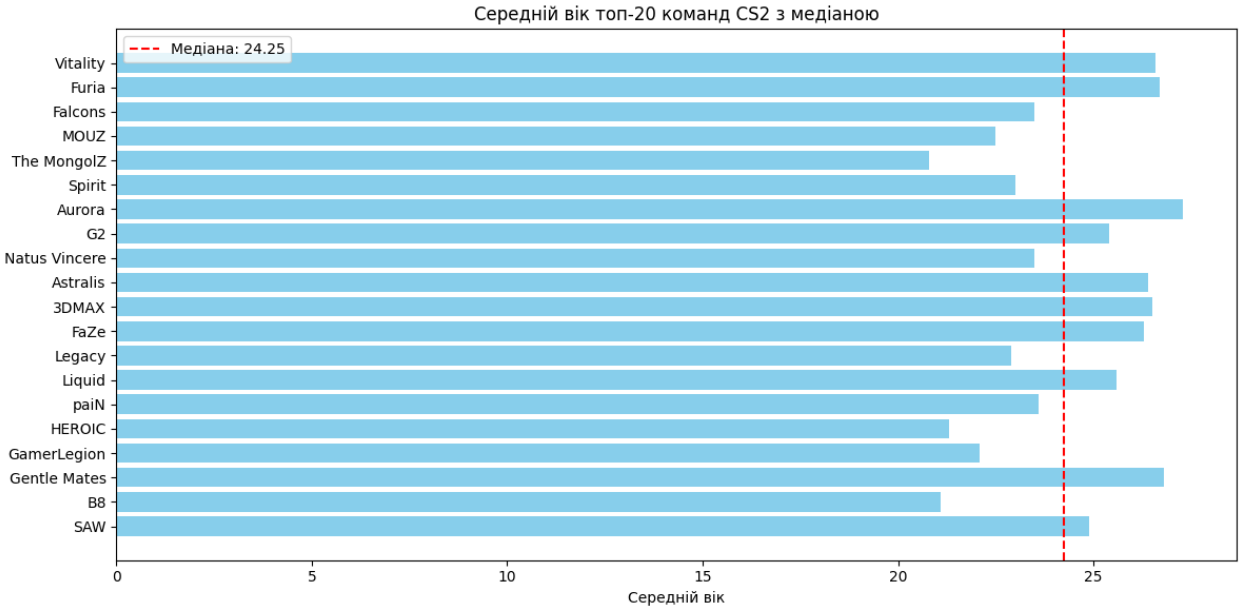


Рисунок 3.2 – Графік середнього віку

Form_diff (різниця форм між командами) та h2h_diff (різниця перемог в особистих зустрічах команд) помножуються на коефіцієнти 0.6 та 0.3 відповідно для того щоб подавляти ці фактори, матч не повинен «ламатися» лише через серію 5 перемог поспіль.

Якщо команда на серії поразок – вона може програти навіть сильнішій за класом, і навпаки. Для цих факторів дано обмеження від -3 до 3, якщо команда навіть має дику серію перемог її вплив на прогноз обмежується максимумом +3, якщо серія поразок жахлива - мінімумом -3. Це запобігає переоцінки тимчасової серії, різких стрибків, хибних піків ймовірності.

3.4 Опис інтерфейсу програми

Розглянемо з яких елементів з фреймворку Streamlit зроблено простий, але зручний інтерфейс вебзастосунку:

– заголовок: st.title(“CS2 Matches Predictor”) – розташований вгорі сторінки;

– інформаційний текст та стилі: st.markdown з CSS для налаштування шрифтів, ширини контейнера, відступів та форматування рядків матчів;

– кнопка оновлення матчів: st.button(“Reload matches”) – при натисканні оновлює список матчів через функцію get_matches_cached(force_reload=True);

– вибір статусу матчів: st.selectbox(“Choose match status”, [“live”, “upcoming”]) – фільтрує матчі по статусу;

– пошук по команді: st.text_input(“Search by team name (enter the name):”) – дозволяє ввести назву команди для фільтрації списку матчів;

– вибір моделі для прогнозу: st.selectbox(“Choose the model”, ...) – вибір моделі машинного навчання для прогнозу матчів;

– список матчів: кожен матч виводиться у вигляді HTML-рядка (st.markdown з класом match-row) з клікабельним посиланням на сторінку матчу та часом проведення;

– тип прогнозу: st.radio(“Choose prediction type”, [“Match prediction”, “Map prediction”]) – дозволяє вибрати прогноз на матч або на конкретну карту;

– вибір карти: st.selectbox (якщо обрано прогноз на карту та карти визначені) – для вибору карти конкретного матчу;

– кнопка для запуску прогнозу: st.button(f”Make prediction for {team1} vs {team2}”) – запускає обчислення прогнозу для обраного матчу;

– вивід прогнозу: st.info та st.success – відображення ймовірності перемоги команд від власної моделі та готових ML-моделей;

– агреговані статистичні дані команд: у вигляді HTML-блоку з середніми показниками гравців (rating, round_swing, dpr, kast, multi_kill, adr, kpr), % перемог команд та карт, місцем у HLTV рейтингу та середнім віком гравців;

– таблиці гравців: st.expander з таблицями st.dataframe для кожної команди, що показують статистику окремих гравців;

– таблиця % перемог по картах: st.expander з HTML-таблицею, що порівнює відсоток перемог команд на всіх картах.

3.5 Висновки за розділом 3

В цьому розділі було створено та показано схему алгоритму функціонування програми.

Також, було детально описано усі компоненти вебзастосунку з детальним описом їх призначення та функцій наявних в них. Проведено детальний аналіз власної моделі, пояснено вибір ваг для різних факторів, а також обґрунтовано застосування нормалізації та інших методів підготовки даних для підвищення точності прогнозів. Детально описано складові інтерфейсу та які методи викликаються при взаємодії користувача з ним. Використання мови програмування Python в поєднанні з фреймворком Streamlit дозволили легко поєднати ML-моделі, БД та фронтенд.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ДОСЛІДЖЕННЯ

4.1 Експлуатація програми

Для експлуатації розробленого ПЗ необхідно мати стабільне інтернет з’єднання та перейти за посиланням: https://cs2predictor-nuzp2025diploma.streamlit.app/.

Після запуску перед користувачем з’явиться вкладка з лайв матчами, якщо вони є, на рис. 4.1 можна побачити цю вкладку.

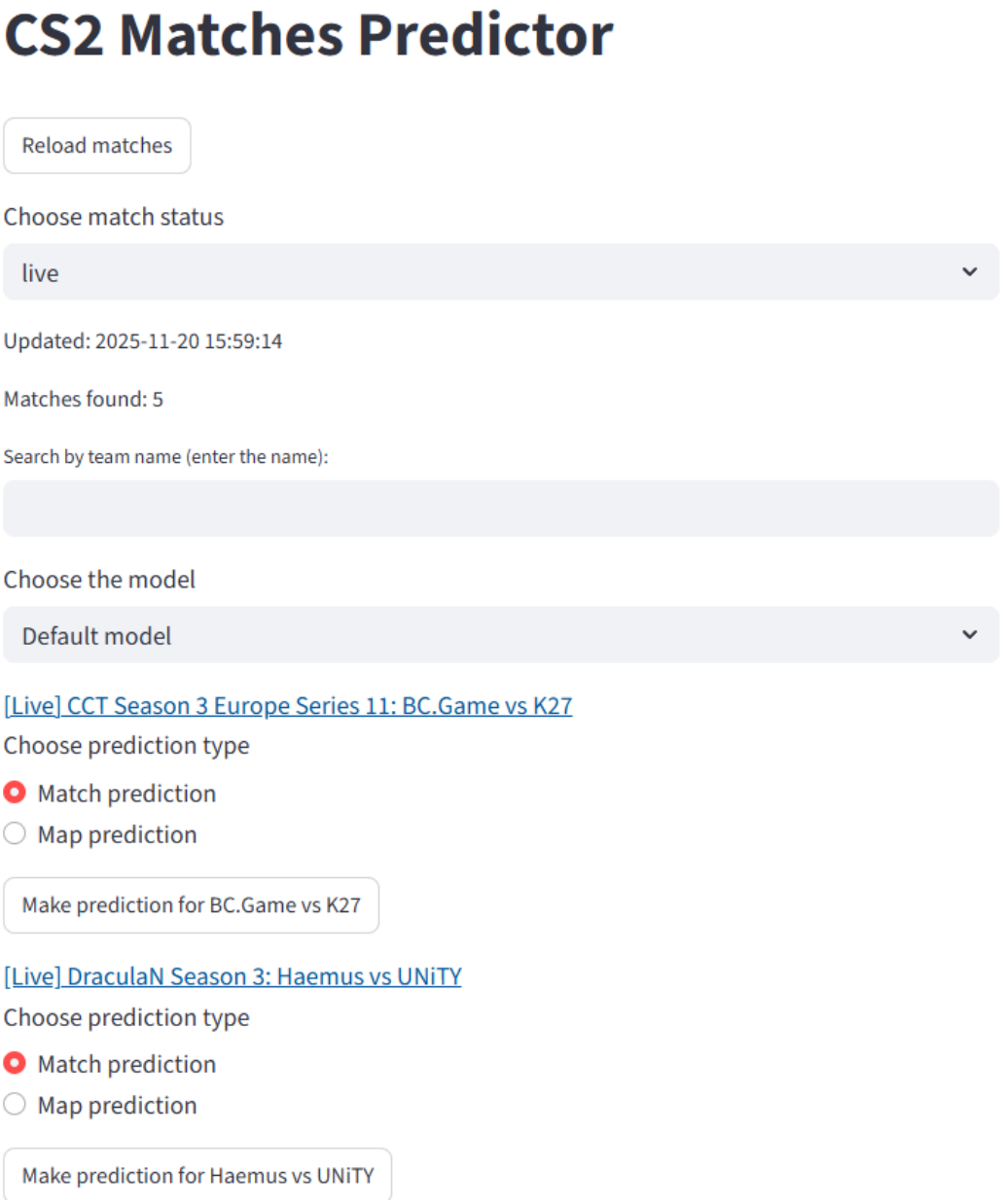


Рисунок 4.1 – Вкладка з лайв матчами

Користувач може потрапити до іншого розділу, а саме до секції майбутніх матчів, які пройдуть в найближчий тиждень, для цього йому потрібно в селекторі “Choose match status” обрати “upcoming”, цей розділ можна побачити на рис. 4.2.

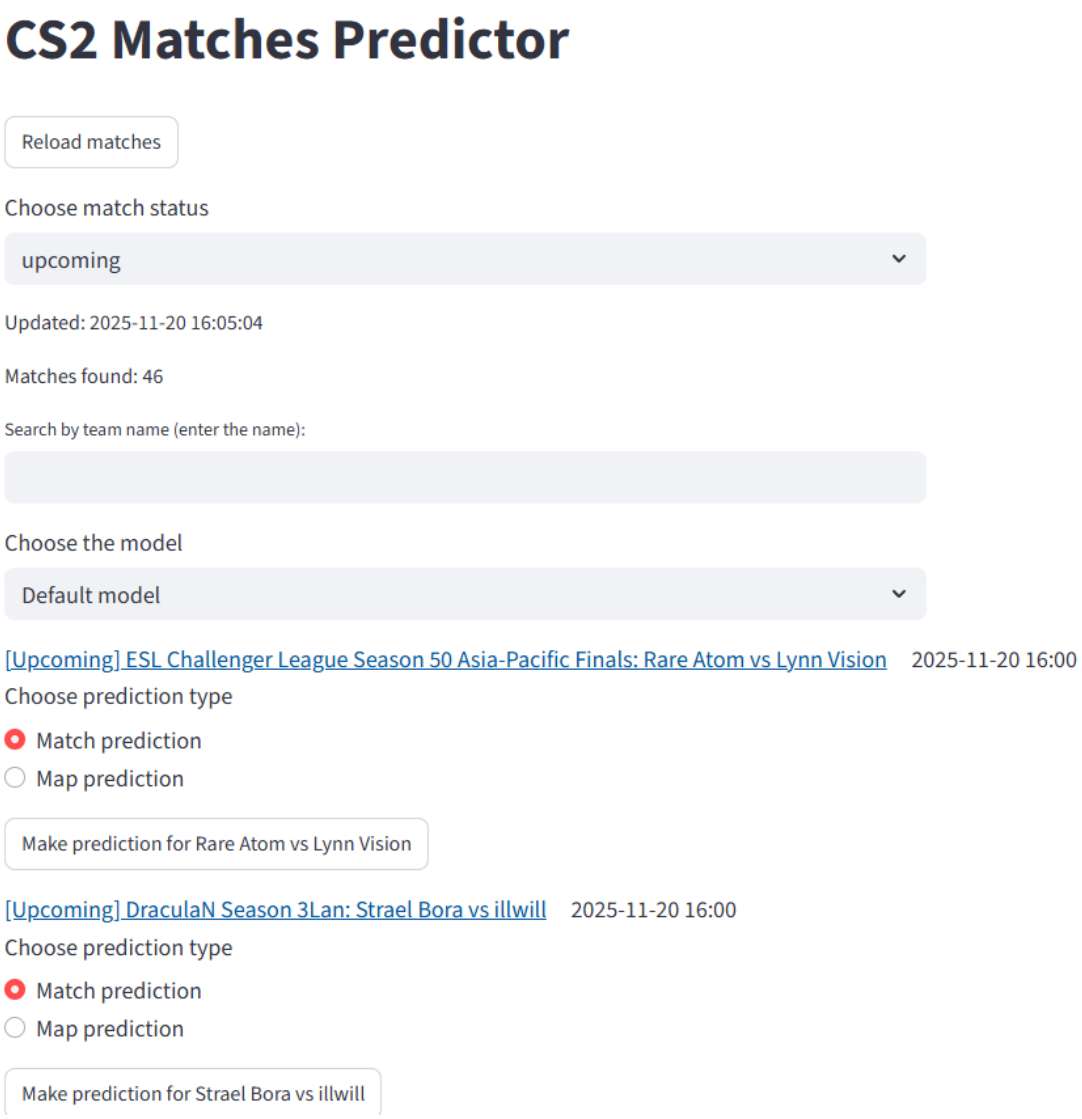


Рисунок 4.2 – Вкладка з майбутніми матчами

Далі він може обирати, який тип прогнозу він хоче отримати на матч, або на конкретну карту, за замовченням для всіх матчів стоїть прогноз на матч, для зміни типу прогнозу, потрібно в radio-селекторі обрати “Map prediction”, і якщо на цей матч вже відомі карти – в селекторі який з’явиться після вибору прогнозу на матч, обрати карту на який користувач хоче

зробити прогноз. Також потрібно звернути увагу, що після вибору прогнозу на карту, зміна моделі для прогнозування – неможлива, через те що вже готові моделі навчалися на історичних даних, які не враховували конкретні карти. Також, часто для майбутніх матчів карти заздалегідь невідомі, вони стають відомі приблизно за 5-10 хвилин до початку матчу, тому частіше вибір карти актуальніше використовувати в секції лайв, приклад вибору прогнозу на карту можна побачити на рис. 4.3.

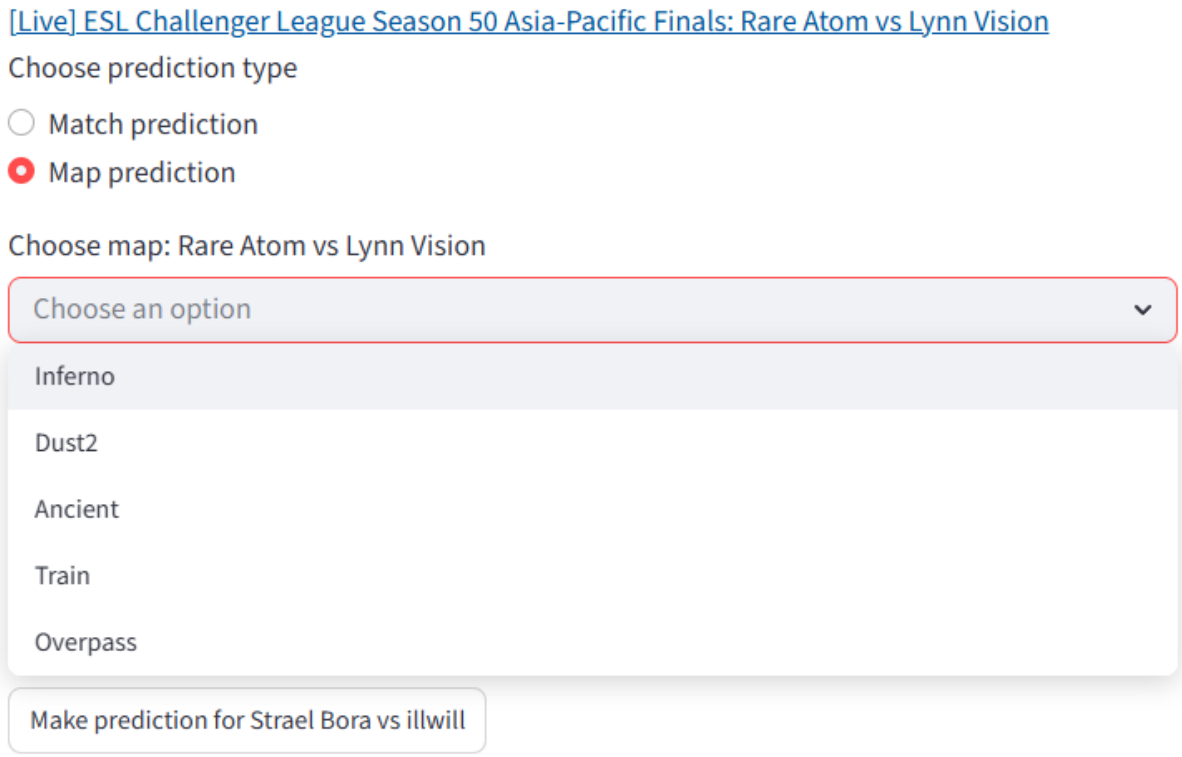


Рисунок 4.3 – Вибір карти для прогнозу на карту

Також, звичайно продумано вивід попереджень про те що карти на матч ще не визначені і попередження при спробі зробити прогноз на карту, якщо карти ще невідомі, або не обрані у селекторі карт, приклад таких повідомлень можна побачити на рис. 4.4 та 4.5 відповідно.

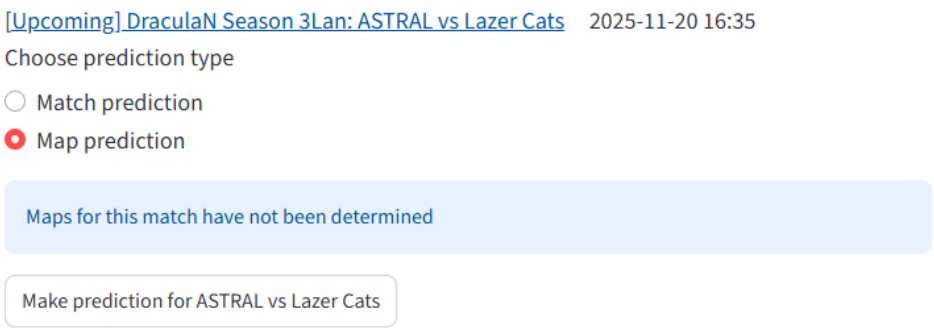


Рисунок 4.4 – Повідомлення про невідомі карти для майбутнього матчу

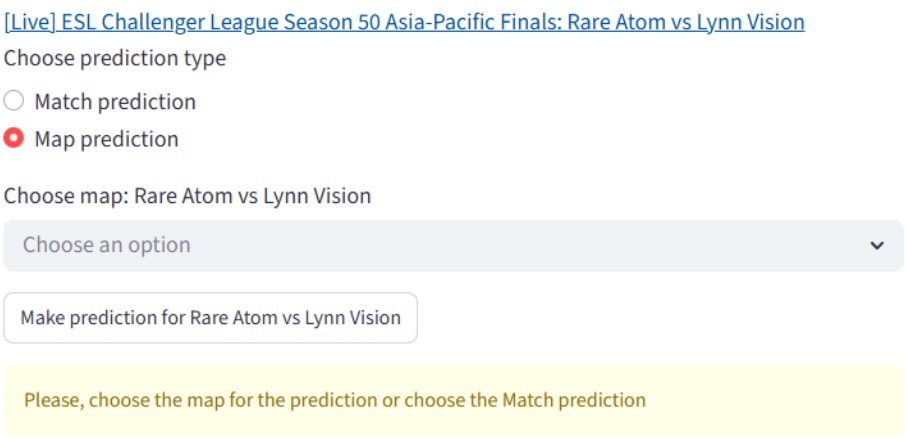


Рисунок 4.5 – Повідомлення про спробу прогнозу на карту без обраної карти

Також, для прогнозу на матч, користувач може вибрати за допомогою якої моделі він хоче отримати прогноз, для цього в селекторі “Choose the model” потрібно обрати певну модель, за замовченням стоїть власна модель, приклад вибору моделі вказано на рис. 4.6.

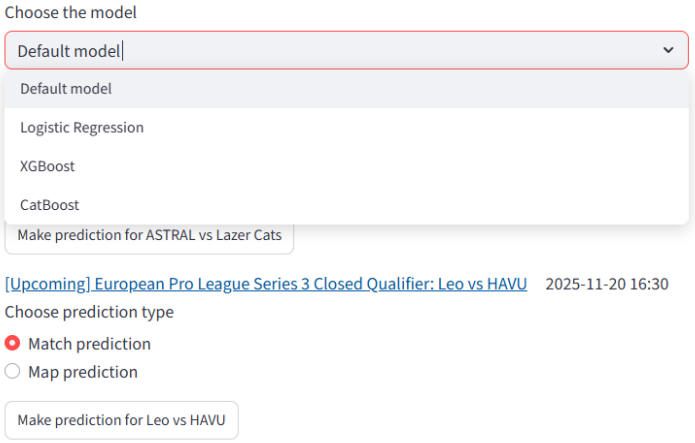


Рисунок 4.6 – Вибір моделі для прогнозування на матч

Для отримання прогнозу на матч, або карту по певному матчу необхідно натискати кнопку “Make prediction” після цього з’явиться прогноз на матч, якщо обрано прогноз на матч, який можна побачити на рис. 4.7, або на карту, якщо обрано конкретну карту для прогнозу який можна побачити на рис. 4.8.

Рисунок 4.7 – Прогноз на матч зі статистикою

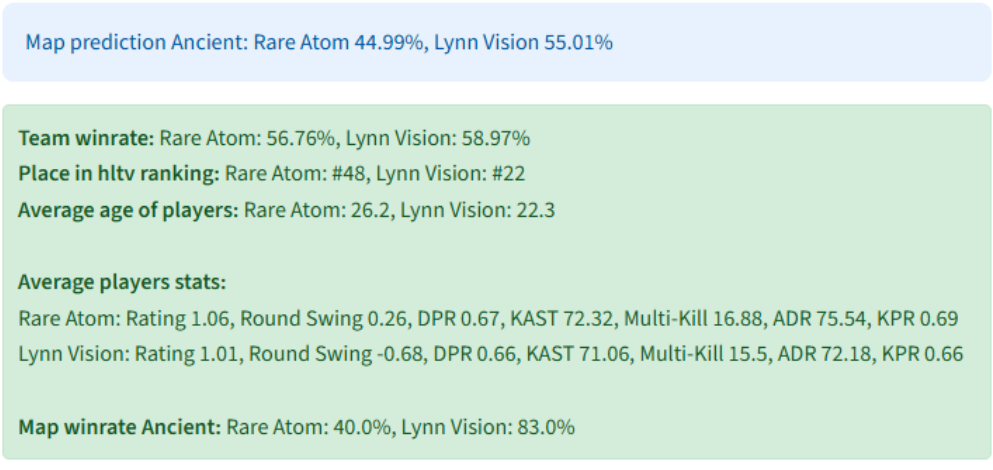


Рисунок 4.8 – Прогноз на карту зі статистикою

Також, можна детально переглянути статистичні дані гравців, які занесено в expandable таблиці, які можна розгорнути та таблицю % перемог по певним картам, їх можна побачити на рис. 4.9 та 4.10 відповідно.

Players stats Rare Atom

	nickname	rating	round_swing	dpr	kast	multi_kill	adr	kpr
1	Marek	0.91	-1.44	0.66	70	12.8	60.1	0.56
2	ChildKing	1.14	0.79	0.7	73.3	20.5	81.7	0.78
3	L1haNg	1.13	0.26	0.68	73	18.8	85.6	0.73
4	Summer	1.01	-0.54	0.67	73.5	14.5	73.8	0.66
5	Tiger	1.09	0.87	0.64	71.8	17.8	76.5	0.72

Players stats Lynn Vision

	nickname	rating	round_swing	dpr	kast	multi_kill	adr	kpr
1	Starry	1.09	-0.2	0.64	73.5	16.7	77.4	0.72
2	EmiliaQAQ	0.97	-0.68	0.68	68.8	15	70	0.61
3	C4LLM3SU3	0.94	-1.07	0.67	70	13.4	67.4	0.6
4	z4kr	1.1	1.14	0.6	73.1	16.3	73.7	0.71
5	Westmelon	0.97	-0.9	0.69	69.9	16.1	72.4	0.64

Рисунок 4.9 – Таблиця статистичних даних гравців

Map winrate(%)

Map	Rare Atom	Lynn Vision
ancient	40.0%	83.0%
dust2	67.0%	33.0%
inferno	64.0%	50.0%
mirage	69.0%	-
nuke	-	33.0%
overpass	-	25.0%
train	33.0%	75.0%

Рисунок 4.10 – Таблиця % перемог команд по картам

При виборі іншої моделі для прогнозування на матч результат прогнозування цієї моделі буде показано додатково з результатами стандартної моделі, приклад для моделі логістичної регресії можна побачити на рис. 4.11.

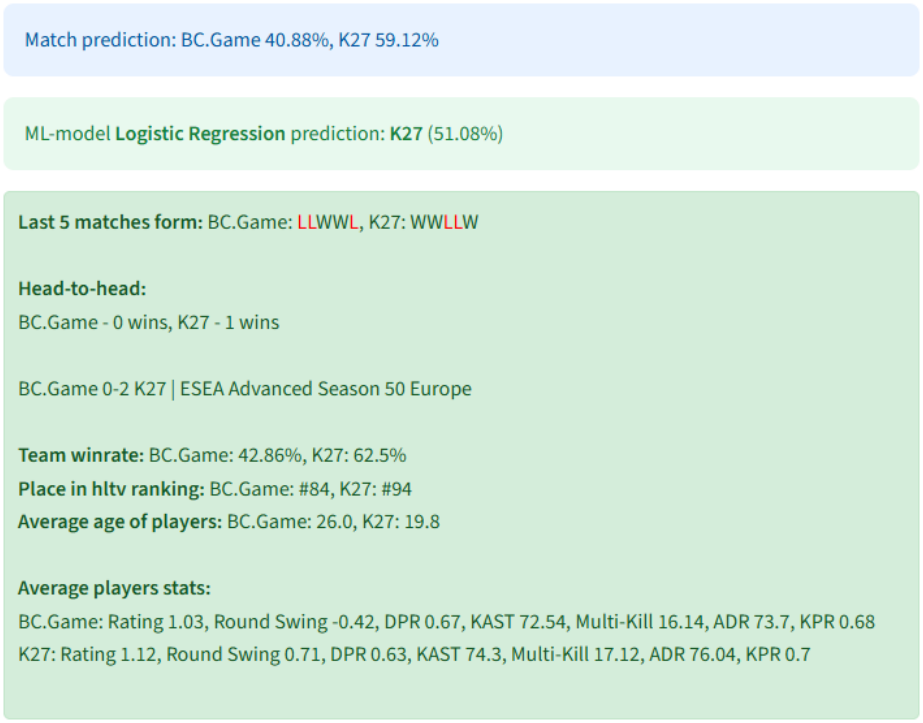


Рисунок 4.11 – Приклад прогнозу з іншою обраною моделлю

Також, користувач може оновити список доступних матчів для прогнозу, для цього потрібно натиснути кнопку “Reload matches” і якщо інформація в БД змінилася та в неї додалися нові матчі, тоді вони будуть відображені для користувача. Також, користувач може натискати на посилання будь-якого матчу який його цікавить та потрапляти до сайту hltv.org зі сторінкою цього матчу. І, користувачу доступен пошук серед матчів за назвою команди, для цього потрібно натиснути по полю з написом “Search by team name”, написати назву комнади в будь-якому регістрі на натиснути клавішу Enter, якщо для цієї команди знайдено матчі, лише вони залишаться на сторінці, для скасування пошукового запиту потрібно стерти

цю назву і натиснути клавішу Enter, приклад вдалого та невдалого пошукових запитів можна побачити на рис. 4.12 та 4.13 відповідно.

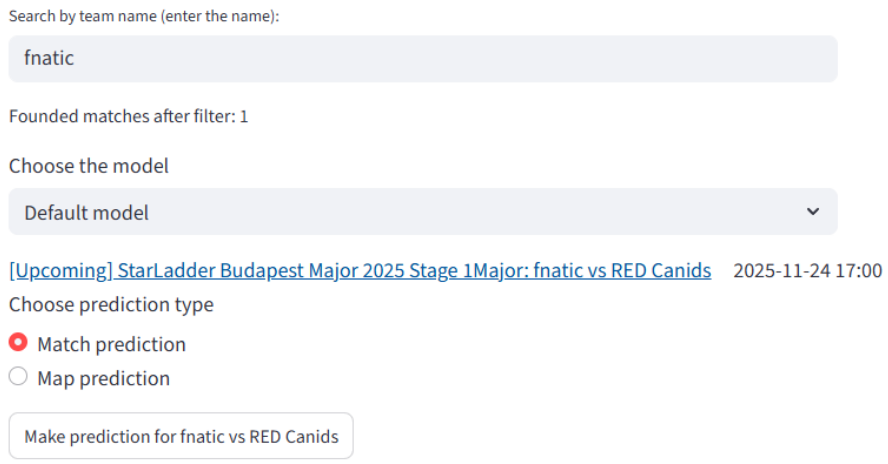


Рисунок 4.12 – Вдалий пошуковий запит

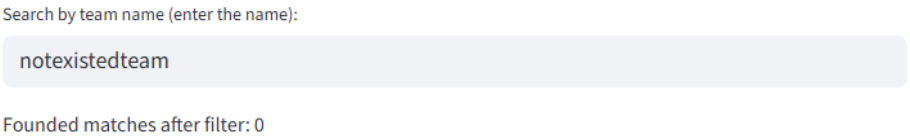


Рисунок 4.13 – Невдалий пошуковий запит

4.2 Тестування програми

Відповідно до вимог до ПЗ, які вказані в технічному завданні та підпункті 2.2 розроблено перевірки, які можна побачити у чеклисті в табл. 4.1, усі перевірки пройдено успішно, що свідчить про коректність реалізації та роботи програми.

Таблиця 4.1 – Чеклист тестування програми

Назва тесту	Результат
Перегляд вкладки з лайв матчами, якщо вони наявні	+
Перегляд вкладки з майбутніми матчами	+
Оновлення матчів	+
Пошук матчів за назвою команди з валідними даними	+

Продовження таблиці 4.1

Назва тесту	Результат
Пошук матчів за назвою команди з невалідними даними	Виводиться відповідне повідомлення про те що за такою назвою не знайдено жодного матчу
Скасування фільтру пошуку	+
Спроба прогнозування на матч, якщо карти невідомі	+
Спроба прогнозування на матч, якщо карти відомі	+
Спроба прогнозування на матч з іншими моделями	+
Спроба прогнозування на карту з іншими моделями	Вибір інших моделей блокується
Спроба прогнозування на карту, якщо карти невідомі	Виводиться відповідне повідомлення
Спроба прогнозування на карту, якщо карти відомі, але жодної карти не обрано	Виводиться відповідне повідомлення
Спроба прогнозування на карту з обраною картою	+
Коректність відображення прогнозування для усіх типів прогнозування та моделей	+
Багатокористувацьке використання	+
Збереження статистичної інформації по матчам в базу даних	+
Збереження результатів прогнозів моделі в базу даних	+
Коректність відображення інтерфейсу при зміні розмірів вікна	+

4.3 Експериментальне дослідження

Після тестування програми на конкретних матчах та отримання прогнозів можна подивитися наскільки точно моделі роблять прогнози та зрозуміти чим власностворена модель краще. Почати потрібно з того, що через те що власностворена модель може давати прогнози на карту, а інші моделі навчалися на вибірці без конкретних карт, і також очевидно, що власна модель використовує статистики гравців і актуальні склади команд, коли інші моделі враховують лише форму, історію зустрічей двох команд та

різницю по виграних та програних картах в історичній базі. Також плюсом власної моделі є те що за допомогою неї можна отримати прогноз навіть якщо команди ще не було в історичній базі, в той час як для готових моделей обов'язково потрібні мітки команд які беруться при навчанні моделей.

В якості експерименту протягом декількох днів усі моделі давали прогнози на певні матчі, тобто моделі перевірялися в режимі реального часу на реальних матчах.

Додатково потрібно відзначити що частина з матчів на які моделі робили прогнози – це матчі команд не дуже високого рівня, більшість цих матчів проходила в онлайн-форматі на турнірах з невисоким призовим фондом. У теорії, команди нижчого рівня можуть здолати будь-яку іншу, особливо в турнірах низького рівня: у таких матчах рівень конкуренції не надто суворий, а мотивація для "договірних" результатів може бути значною.

Далі наведено реальні приклади таких матчів:

– за останній рік ESIC (Esports Integrity Commission) провела розслідування проти ATOX Esports щодо підозр у маніпуляціях результатами ("match-fixing"). У травні 2025 року ESIC заборонила кількох гравців та співробітників ATOX через докази договірних матчів, включно з понад 70 "підозрілими ставками" на матчі своєї ж команди [26];

– Еркхан "gokushima" Багинанов отримав дворічний бан від ESIC за підставні матчі у квітні 2024 року. У жовтні 2025-го ESIC скоротила його бан, бо він співпрацював зі слідством [27];

– також зафіксовані випадки з українськими гравцями: наприклад, ESIC заблокувала гравця "Ganginho" – за даними розслідування, він зробив сотні ставок, в тому числі на матчі власної команди, і виграв понад 20 000 $ [28].

Аналітик OverDrive заявив, що "нова ера підставних матчів" починається в CS2 через те, що Valve (компанія-власник гри), дозволяє підозрілим командам брати участь у відкритому рейтинговому циклі – і це створює умови для потенційних маніпуляцій [29].

Потрібно відзначити, якщо турнір або матчі проводяться в онлайн-форматі неможливо повністю контролювати “чесність” гравців, на більшості турнірів вони зобов’язані грати з увімкненими вебкамерами, але це не може повністю нівелювати іншу проблему матчів низького рівня, а саме – використання читів (використання сторонньої апаратури, програм та помилок гри для забезпечення нечесної переваги в комп’ютерних іграх).

Далі наведено конкретні нещодавні приклади цієї проблеми:

– гравець Joel “joel” Holmlund отримав довічний бан за читерство: ESIC заявила, що використовував складні “апаратні” чити [30];

– розслідування підтвердило, що Райимбек “dune” Дусенов з KHAN грав на обліковому записі 1Drezz, що підтверджено даними обладнання з системи Akros Anti-Cheat. CCT кваліфікувала це як явне видавання себе за іншу особу та спільне використання облікового запису, що є серйозними порушеннями правил турніру [31];

– Akros Anti-Cheat виявила програмне забезпечення для шахрайства, що призвело до кількох постійних дискваліфікацій. В результаті кілька гравців з команд THE, KHAN та Y5 тепер були дискваліфіковані з усіх змагань CCT [31].

З цього можна зробити певні висновки:

– прогнозування точніше працює там, де команди стабільні, статистика гравців повна, а ймовірність зовнішніх маніпуляцій низька. У таких умовах алгоритми можуть адекватно оцінювати силу команд і враховувати історичні та актуальні дані;

– у матчах нижчого рівня є ймовірність “договірних матчів” та використання читів, а також непередбачувані результати через випадкові фактори. Моделі в таких випадках частіше дають невірні прогнози, оскільки частина подій знаходиться поза межами доступних статистичних даних;

– прогнозування в умовах потенційних шахрайських дій потребує обережності: навіть найсучасніші алгоритми не можуть врахувати приховані

ставки, змову команд чи використання сторонніх програм, що робить точність прогнозів суттєво нижчою.

Тому моделі для прогнозування результатів кіберспортивних матчів доцільно застосовувати для прогнозів на турніри професійного рівня з перевіреною статистикою гравців і команд. Для нижчих онлайн-турнірів та матчів із сумнівною чесністю результати прогнозів слід інтерпретувати з великою обережністю, розглядаючи їх як орієнтовні, а не абсолютні.

Перейдемо до експериментального дослідження – на рис. 4.14 зображено графік зіставлення фактичних переможців і прогнозів різних моделей.

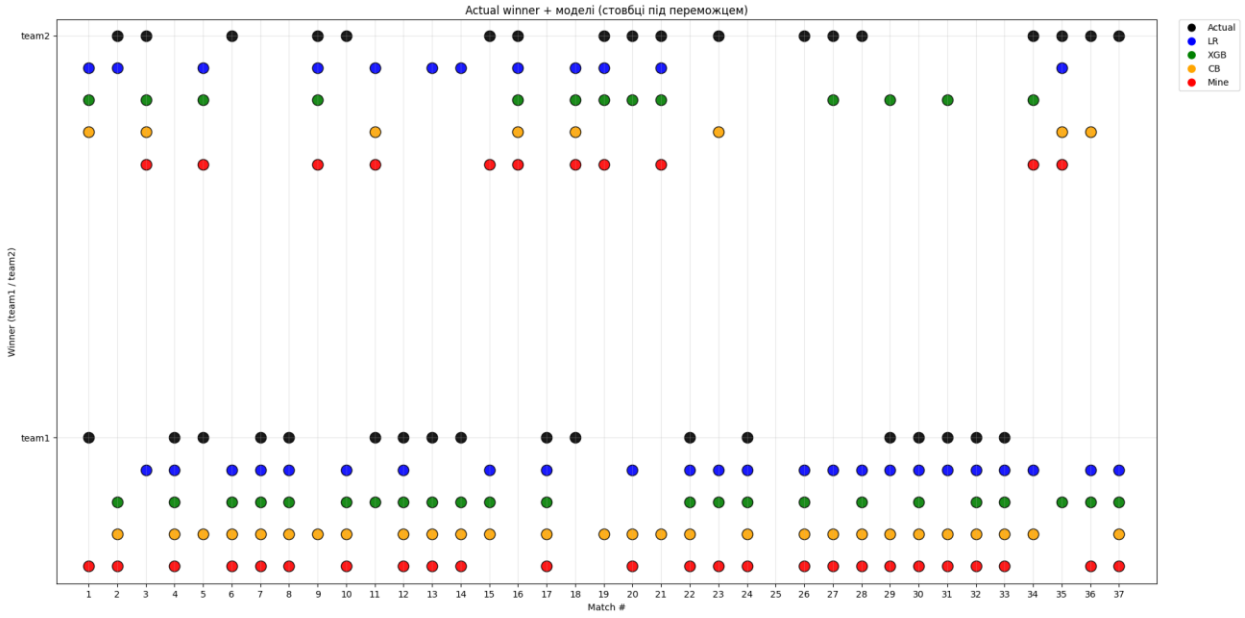


Рисунок 4.14 – Зіставлення прогнозів з фактичним переможцем

На рис. 4.15 зображено графік точності моделей. З цих графіків видно, що логістична регресія найгірше робить прогнози, ця модель продемонструвала найнижчу точність прогнозування – близько 50 %, що помітно гірше за інші застосовані моделі. На практиці цей алгоритм ефективний переважно для задач з лінійно відокремлюваними класами, де взаємозв’язки між ознаками є простими, стабільними та не містять складних нелінійних взаємодій.

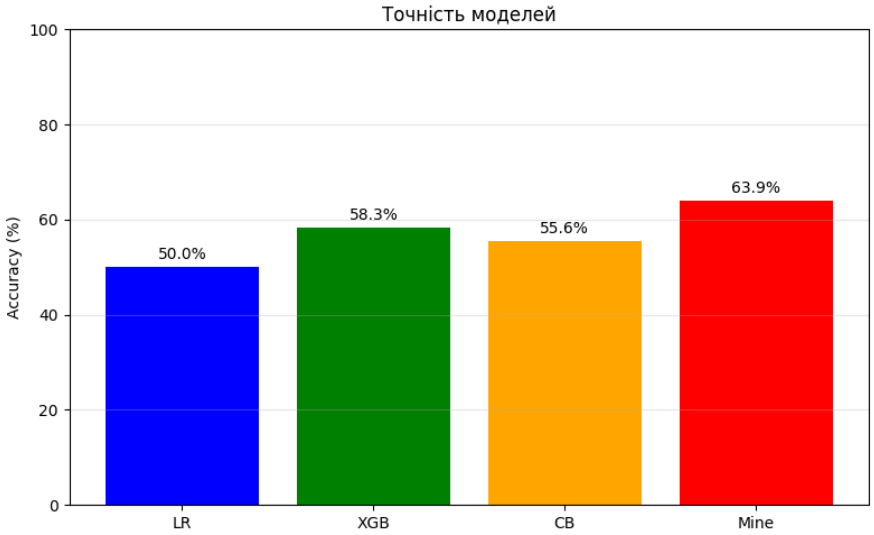


Рисунок 4.15 – Графік точності моделей

Логістична регресія найкраще працює у випадках, коли структура даних є відносно “пласкою”, а внесок кожної ознаки в результат можна описати лінійною функцією [32].

В задачі прогнозування результатів кіберспортивних матчів модель виявилась недостатньо гнучкою: взаємодії між статистиками гравців, формою команд, набором карт та іншими чинниками є явно нелінійними й складно структурованими. Саме тому логістична регресія часто давала хибні прогнози навіть у тих ситуаціях, де три інші моделі впевнено визначали переможця. Висока варіативність реальних матчів та залежність від комбінацій багатьох факторів робить задачу значно складнішою, ніж те, для чого цей метод був створений, що й зумовило його низьку ефективність у цьому контексті. Можна подивитись на те що в матчах №2, 13, 14 ця модель була окремо від інших моделей, і в двох випадках з трьох помилково обрала переможця.

Розглянемо моделі CatBoost та XGBoost, їх точність склала 55.6 % та 58.3 % відповідно. Обидві моделі – це градієнтний бустинг над деревами рішень, який дозволяє уловлювати складні та нелінійні взаємозв’язки між ознаками (наприклад, статистика гравців, форма команд, H2H). Бустинг послідовно “виправляє” помилки попередніх дерев, що надає моделі високу

виразну силу. Проте, навіть найкращі бустингові моделі залежать від якості ознак, якщо суттєві фактори не представлені у даних, моделі не можуть їх врахувати. Хоча XGBoost і CatBoost контролюють перенавчання, вони все одно можуть демонструвати “варіабельність” на обмежених чи мало різноманітних даних. Існують такі взаємодії між ознаками, які важко повністю описати деревами – можливо, нейронні мережі або ансамблеві моделі могли б досягти кращих результатів, але вони потребують значно більше даних, доступ до яких обмежено через відсутність API у сайту hltv.org, і ресурсів. Якщо гіперпараметри (глибина дерев, швидкість навчання, регуляризація) не підібрані оптимально, модель може не використовувати повністю свій потенціал. Також, у кіберспорті багато залежить від настрою гравців, тактики, синергії; ці динамічні фактори дуже важко формалізувати як табличні ознаки – і моделі не завжди здатні передбачити їх.

Цікавими є матчі № 6, 26 та 37 – усі моделі одночасно обрали одну команду, а перемогла інша. Так, у матчі №6 власна модель прогнозувала перемогу з ймовірністю 50,33 % проти 49,67 % для суперника, що свідчить про високу невизначеність та близькість потенційних результатів. У матчах №26 та №37 усі моделі обрали одну команду як фаворита, однак перемогу здобула інша, що свідчить про вплив факторів, які не враховуються моделями.

З наукової точки зору, прогнозування спортивних результатів ніколи не може досягти 100 % точності через існування недооцінки суперника, випадкових обставин, психологічного стану гравців, тактичних рішень у матчі та інших непередбачуваних подій. Моделі працюють лише з доступною статистичною інформацією, тому певна частина результатів завжди залишається невидимою для алгоритмів, що пояснює окремі невірні прогнози навіть для високоточних моделей.

Власна модель дала точність 63.9 %, що є найкращим показником серед протестованих моделей. Такий результат став можливим завдяки

комплексному підходу до побудови ознак та використанню актуальних даних. Модель враховує не лише історичні результати команд, а й статистику гравців, включаючи показники ефективності. Це дозволяє оцінювати не лише командну взаємодію, але й індивідуальні сильні та слабкі сторони учасників матчу.

Додатковою перевагою стало використання нормалізації ознак і правильно підібраних ваг для різних параметрів, що дозволяє моделі коректно порівнювати різнорідні дані (наприклад, рейтинги гравців та відсоток перемог команди). Такий підхід знижує вплив надмірно сильних або слабких показників і підвищує стабільність прогнозів.

Порівняно з CatBoost та XGBoost, власна модель краще інтегрує часову компоненту – актуальні дані про форму гравців та команд, що дає перевагу при прогнозуванні матчів у реальному часі. Наприклад, якщо команда має зміну складу або проходить серію поразок, модель швидко адаптує прогноз, на відміну від моделей, які використовують лише історичні дані без динамічного коригування.

Також, за допомогою цієї моделі було зроблено прогноз на конкретні карти, знову ж таки, через обмеження доступу до статистичної інформації для інших моделей зробити це було проблематично. В прогнозі по картах ця модель показала кращі результати, а саме близько 70% точності (23 з 33 правильних переможців по картах). На це є дві головні причини: команди мають стабільніші показники на певних мапах, ніж у загальному матчі. Наприклад, деякі команди спеціалізуються на конкретних картах, і їх перевага проявляється більш явно, що полегшує моделі розпізнавання закономірностей. У прогнозі на матч загалом враховується більше факторів – форма команд, історія зустрічей двох команд, на рівні однієї карти ці фактори проявляються менш хаотично, що підвищує передбачуваність результату.

Очевидно, що ця модель не є ідеальною, для її покращення можна додати більше тактичних метрик, наприклад % виграних раундів при

перевазі 5v4, % виграних раундів при ситуації 4v5, % виграних пістолетних раундів, тощо. Також, можна поєднати різні моделі, наприклад додати використання ансамблевої мережі, це може підвищити точність за рахунок інтеграції різних підходів до прогнозування. Можна спробувати більш точніше підібрати гіперпараметри, наприклад змінювати їх до тих пір поки точність не сягне 65%. Також, можна враховувати зовнішні фактори такі як: рівень мотивації, психологічний стан гравців, призовий фонд турніру. Впровадження таких покращень дозволить підвищити адаптивність моделі до змін у формі команд та до непередбачуваних подій у матчах, що особливо важливо для прогнозування на реальних турнірах.

4.4 Висновки за розділом 4

В цьому розділі описано процес експлуатації розробленого ПЗ, написано перелік дій які необхідно зробити для коректної роботи з ним, описано усі можливі дії під час експлуатації.

Створено чеклист перевірок розробленого ПЗ який показав що вебзастосунок коректно відображає усі розроблені елементі інтерфейсу, правильно реагує на дії користувача, видає відповідні повідомлення при виключних ситуаціях, дозволяє працювати з ним багатьом користувачам.

Проведено експериментальне дослідження програми, в якому порівняно результати моделей, розібрано чому власна модель дала кращі результати, розглянуто фактори які могли вплинути на невисоку точність моделей.

У підсумку розроблена програма є зручною для використання, не містить помилок та дозволяє користувачам отримувати прогнози на актуальні кіберспортивні матчі. Власну модель можна покращити шляхом розширення набору даних, використання більш детальної статистики гравців та команд, а також інтеграції нових алгоритмів машинного навчання для підвищення точності прогнозів. Це дозволить зробленому ПЗ бути ще більш

надійним та ефективним у прогнозуванні результатів кіберспортивних матчів.

ВИСНОВКИ

У ході виконання дипломної роботи було проведено дослідження предметної області та розроблено програмне забезпечення для прогнозування результатів кіберспортивних матчів у дисципліні Counter Strike 2. Було визначено більшу складність прогнозування результатів матчів порівняння з традиційними видами спорту, розглянуто які показники можуть впливати на прогноз. Проаналізовано існуючі аналоги та виділено прогалини у доступних рішеннях, зокрема відсутність безкоштовних зручних додатків із прозорими прогнозами та можливістю вибору конкретних матчів. На основі цього сформовано вимоги до ПЗ та поставлено завдання на розробку.

Вибір інструментарію був обґрунтований потребами проєкту: мова Python та фреймворк Streamlit забезпечили зручну інтеграцію ML-моделей, база даних PostgreSQL дозволила ефективно зберігати та обробляти статистику гравців і команд. Було спроєктовано структуру програми, створено алгоритми функціонування, детально описано компоненти інтерфейсу та взаємодію користувача з ПЗ.

Особливу увагу приділено розробці власної моделі прогнозування, яка інтегрує як історичні, так і актуальні дані, що дозволяє швидко адаптувати прогнози до змін у складі команд або поточної форми гравців. Проведене експериментальне тестування показало, що власна модель перевищує точність стандартних алгоритмів CatBoost та XGBoost на 8.3 % та 5.6 % відповідно, а також дозволяє прогнозувати результати на рівні окремих карт із вищою передбачуваністю.

В процесі експлуатації ПЗ було підтверджено його стабільність та коректну роботу, зручність інтерфейсу та надійність обробки даних. Разом з тим, власну модель можна покращити шляхом розширення набору статистичних даних, впровадження більш детальних тактичних метрик, інтеграції ансамблевих методів та оптимізації гіперпараметрів. Також

перспективним є врахування зовнішніх факторів, таких як психологічний стан гравців та рівень мотивації.

У підсумку, розроблене програмне забезпечення забезпечує користувачам можливість отримувати точні та актуальні прогнози результатів кіберспортивних матчів, поєднує зручний інтерфейс із потужними алгоритмами прогнозування та має потенціал для подальшого розвитку та вдосконалення точності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Esports. Wikipedia [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Esports.

2. Greig Connor. The Ongoing Rise of the Esport Industry [Electronic resource] / Connor Greig. – Access mode: https://sites.lsa.umich.edu/mje/2024/01/15/the-ongoing-rise-of-the-esport-industry.

3. Most viewed esports events of 2024 [Electronic resource]. – Access mode: https://escharts.com/news/best-esports-events-2024.

4. Global Engagement & Audience Report (Executive summary) [Electronic resource]. – Access mode: https://inside.fifa.com/tournament-organisation/audience-reports/qatar-2022.

5. List of most-watched television broadcasts. Wikipedia [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/List_of_most-watched_television_broadcasts.

6. Sattar Maya. Esports vs. Sports: How do the two compare? [Electronic resource] / Maya Sattar. – Access mode: https://esportsinsider.com/esports-vs-sports.

7. Wagner Patrick. How eSport Prize Purses compare to Traditional Sports [Electronic resource] / Patrick Wagner. – Access mode: https://www.statista.com/chart/14121/esport-prize-pools-in-comparison-to-traditional-sporting-events.

8. Kane Daniel. Recognizing Esports as a Sport [Electronic resource] / Daniel Kane, Brandon D. Spradley. – Access mode: https://thesportjournal.org/article/recognizing-esports-as-a-sport.

9. Ziya Kocak Umut. Are eSports more than just sitting? A study comparing energy expenditure [Electronic resource] / Umut Ziya Kocak. – Access mode: https://becarispublishing.com/doi/pdf/10.2217/cer-2021-0223.

10. Ketelhut S. Heartbeats and high scores: esports triggers cardiovascular and autonomic stress response [Electronic resource] / S. Ketelhut, C. R Nigg. –

Access mode: https://pmc.ncbi.nlm.nih.gov/articles/PMC11024241.

11. Olympic Esports Games. Wikipedia [Electronic resource]. – https://en.wikipedia.org/wiki/Olympic_Esports_Games.

12. Top Esports Games Ranked by Live Viewership Records [Electronic resource]. – https://escharts.com/news/top-esports-games-livestreaming-records-seo.

13. Шутер від першої особи. Вікіпедія [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Шутер_від_першої_особи.

14. Björklund A. Predicting the outcome of CS:GO games using machine learning [Electronic resource] / A. Björklund, W. J. Visuri, F. Lindevall, P. Svensson. – Chalmers University of Technology. – 2018. – Access mode: https://publications.lib.chalmers.se/records/fulltext/256129/256129.pdf.

15. Faceit [Electronic resource]. – Access mode: faceit.com.

16. Rubin A. Predicting Round and Game Winners in CSGO [Electronic resource] / A. Rubin. – Access mode: https://doi.org/10.31219/osf.io/u9j5g.

17. Hltv. Analytics [Electronic resource]. – Access mode: https://www.hltv.org/betting/analytics.

18. Bo3. Matches [Electronic resource]. – Access mode: https://bo3.gg/matches/current.

19. Point516. Github [Electronic resource]. – Access mode: https://github.com/point516/cs_bot.

20. Python. Вікіпедія [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Python.

21. Streamlit documentation [Electronic resource]. – Access mode: https://docs.streamlit.io.

22. PostgreSQL documentation. JSON Types [Electronic resource]. – Access mode: https://www.postgresql.org/docs/12/datatype-json.html.

23. Whats The Difference Between PostgreSQL vs MySQL [Electronic Resource]. – Access mode: https://www.netdata.cloud/academy/postgres-mysql-differences.

24. Railway. Documentation [Electronic resource]. – Access mode: https://docs.railway.com/reference/databases.

25. Musaab Abdalla. Railway Case Study [Electronic resource] / Abdalla Musaab. – Access mode: https://library.noroff.dev/railway/railway-case-study.

26. Taifalos Nicholas. Multiple CS2 players handed lifetime bans by ESIC for match-fixing, betting [Electronic resource] / Nicholas Taifalos. – Access mode: https://esports.gg/news/counter-strike-2/atox-esic-lifetime-bans.

27. Biazzi Leonardo. Esports watchdog hands 2-year ban to CS2 pro for match-fixing [Electronic resource] / Leonardo Biazzi. – Access mode: https://esports.gg/news/counter-strike-2/esports-watchdog-hands-2-year-ban-to-cs2-pro-for-match-fixing.

28. ESIC наклала трирічний бан на Ganginho за договірні матчі [Електрон. Ресурс]. – Режим доступу: https://sport.ua/uk/news/743550-ofitsiyno-esic-naklala-tririchniy-ban-na-ganginho-za-dogovirni-matchi.

29. OverDrive predicts "new match-fixing era" will start in CS2 because of VRS [Electronic resource]. – Access mode: https://escorenews.com/en/csgo/news/65518-overdrive-predicts-new-match-fixing-era-will-start-in-cs2-because-of-vrs.

30. CS2 Pro Joel Permanently Banned for Cheating and Blackmailing Tournament Officials [Electronic resource]. – Access mode: https://www.aesa.org.au/popular/cs2-pro-joel-permanently-banned-for-cheating-and-blackmailing-tournament-officials.html

31. CCT issues bans to three teams for cheating [Electronic resource]. – Access mode: https://www.hltv.org/news/42833/cct-issues-bans-to-three-teams-for-cheating.

32. Hosmer Jr. D. W. Applied logistic regression [Text] / D. W. Hosmer Jr., S. Lemeshow, R. X. Sturdivant. – Hoboken: Wiley, 2013. – 528 p.

ДОДАТОК А

Технічне завдання

А.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Дослідження та програмна реалізація засобів прогнозування результатів кіберспортивних матчів» відповідно до наказу № 447 від 30 вересня 2025 року Національного університету «Запорізька політехніка».

А.2 Призначення розробки

Програма має дозволяти користувачам робити прогнози кіберспортивних матчів у дисципліні CS2, отримувати детальну статистику гравців і команд, обирати конкретні матчі для аналізу та переглядати ймовірності перемоги команд як у загальному результаті, так і на окремих картах. Користувачі можуть порівнювати прогнози різних моделей, а також отримувати показники, які було враховано під час прогнозування моделей, що робить роботу з програмою інформативною та зручною для прийняття рішень.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціоналу

Функціональні вимоги яким має відповідати програма:

– користувач повинен мати можливість відкрити вебзастосунок і бачити його вміст;

– користувач повинен мати можливість обирати між відображенням матчів які йдуть у реальному часі та майбутніх матчів;

– користувач повинен мати можливість бачити матчі, що відбуваються у режимі реального часу;

– користувач повинен мати можливість бачити майбутні матчі протягом найближчого тижня;

– користувач повинен мати можливість переходити за гіперпосиланнями на сторінку матчу з hltv.org;

– користувач повинен мати можливість вибирати тип прогнозу: на весь матч або на конкретну карту (якщо вона відома);

– користувач повинен мати можливість робити прогноз на матч, коли карти ще не визначені;

– користувач повинен мати можливість робити прогноз на матч, коли карти визначені;

– користувач повинен мати можливість вибирати карту для прогнозу зі списку доступних;

– користувач повинен отримувати повідомлення про помилку, якщо намагається зробити прогноз на карту, коли вони ще невідомі;

– користувач повинен мати можливість обирати додаткові ML-моделі для прогнозу результату матчу;

– користувач повинен мати можливість бачити прогноз результату обраної ML-моделі у відсотках для обох команд;

– користувач повинен мати можливість бачити форму команд (останні 5 матчів) при прогнозі на матч;

– користувач повинен мати можливість бачити історію зустрічей команд (head-to-head) при прогнозі на матч;

– користувач повинен мати можливість бачити середній вік гравців команди, якщо він доступний на hltv.org;

– користувач повинен мати можливість бачити позицію команд у топі hltv.org;

– користувач повинен мати можливість бачити загальний відсоток перемог команд (для актуального складу команди);

– користувач повинен мати можливість бачити середні статистики гравців за різними параметрами;

– користувач повинен мати можливість бачити окремі таблиці зі статистикою гравців для обох команд;

– користувач повинен мати можливість бачити окремі таблиці з відсотком перемог команд по різних картах (для актуального складу команди);

– користувач повинен мати можливість шукати матчі за назвою команди;

– користувач повинен мати можливість скасувати фільтр пошуку, якщо текст у полі пошуку видалено;

– система повинна парсити та зберігати історичні матчі у БД, які будуть використані для основної та додаткових моделей, і для формування вхідних ознак команд (форма та head-to-head);

– система повинна парсити інформацію зі сторінки матчу, а саме показники гравців та статистичні данні команд;

– система повинна брати інформацію про матчі на які можливо зробити прогноз з бази, якщо дані актуальні, або якщо парсинг неможливий;

– система повинна завантажувати статистичні дані гравців та команд у відповідні таблиці в БД та оновлювати їх, якщо минуло більше 24 годин з моменту останнього запису або нової спроби прогнозу;

– система повинна завантажувати збережені моделі при запуску;

– система повинна формувати вхідні ознаки для моделей на основі історії команд, їх форми та статистики гравців;

– система повинна робити прогноз на весь матч та на конкретну карту за наявності даних;

– система повинна відображати відсоток команд по вибраній карті при прогнозі на карту;

– система не повинна враховувати форму команд та h2h при прогнозі на карту;

– система повинна зберігати прогнози у БД окремо для матчів та карт;

– система повинна відображати дату та час матчу поруч із назвою матчу;

– система повинна зберігати та оновлювати моделі після навчання на історичних даних;

– система повинна мати можливість донавчати моделі після поповнення вибірки історичних матчів;

– система повинна видавати повідомлення про помилки у разі некоректного введення або відсутності даних;

– система повинна відображати результати прогнозів додаткових моделей для порівняння з основною моделлю.

А.3.2 Умови експлуатації

Під час роботи з вебзастосунком потрібно мати стабільне інтернет-з’єднання та мати на комп’ютері будь-який сучасний браузер. Експлуатація програми здійснюється відповідно до пояснення експлуатації відносно розробленої програми, яке міститься в розділі 4 пояснювальної записки, що містить інформацію необхідну для використання.

А.3.3 Вимоги до складу та параметрів технічних засобів

Для нормального функціонування програми, повинні виконуватися наступні умови до апаратної частини сервера:

– процесор AMD Ryzen 7 7700 і новіше;

– оперативна пам’ять 32 Гб і більше;

– постійна пам’ять 500 Гб і більше.

Вимоги для програмної частини клієнта:

– операційна система Windows 11;

– сучасний браузер (Chrome, Edge, Opera, Firefox та інші).

ДОДАТОК Б

Текст програми

Б.1 Текст файлу matches_history.py

```
def fetch_matches_from_page(offset):

    url = f"https://www.hltv.org/results?offset={offset}"
    print(f"Fetching: {url}")

    max_retries = 3
    for attempt in range(max_retries):
        response = scraper.get(url, timeout=15)
        soup = BeautifulSoup(response.text, "html.parser")
        blocks = soup.select(".results-sublist")
        if blocks:
            return blocks
        print(f"Try {attempt + 1} failed, wait 5 seconds...")
        time.sleep(5)
    return []

def parse_matches(blocks, stop_on_last=True):
    new_matches = []
    stop_parsing = False

    for block in blocks:
        date_elem = block.select_one(".standard-headline")
        if not date_elem:
            continue
        date_obj = parse_date(date_elem.text)

        matches = block.select(".result")
        for match in matches:
            teams = match.select(".team")
            if len(teams) < 2:
                continue

            team1 = clean_text(teams[0].text)
```

```
        team2 = clean_text(teams[1].text)

        tournament_elem = match.select_one(".event-name")
        tournament = clean_text(tournament_elem.text) if tournament_elem else
"Unknown"

        map_text_elem = match.select_one(".star-cell .map-text")
        map_text = map_text_elem.text.strip() if map_text_elem else "Unknown"
        match_format = map_text.lower() if map_text.lower() in ["bo1", "bo3", "bo5"]
else "bo1"

        score_elem = match.select_one(".result-score")
        if score_elem:
            scores = score_elem.text.strip().split("-")
            try:
                s1_raw = int(scores[0])
                s2_raw = int(scores[1])
            except:
                s1_raw, s2_raw = 0, 0
        else:
            s1_raw, s2_raw = 0, 0

        if match_format == "bo1":
            s1, s2 = (1, 0) if s1_raw > s2_raw else (0, 1)
        else:
            s1, s2 = s1_raw, s2_raw

        if stop_on_last and last_date:
            if (
                date_obj == last_date and
                team1 == last_team1 and
                team2 == last_team2 and
                tournament == last_tournament
            ):
                print("The latest match was found in the database on the page. We are
```

```
adding more recent matches....")
                stop_parsing = True
                break

            new_matches.append((date_obj, team1, team2, match_format, s1, s2,
tournament))

        if stop_parsing:
            break

    return new_matches, stop_parsing


new_matches = []

if not last_match:

    for offset in range(20000, -100, -100):
        blocks = fetch_matches_from_page(offset)
        if not blocks:
            continue
        page_matches, _ = parse_matches(blocks, stop_on_last=False)

        new_matches.extend(page_matches)
        time.sleep(1)

else:

    for offset in range(0, 2000, 100):
        blocks = fetch_matches_from_page(offset)
        if not blocks:
            continue
        page_matches, stop = parse_matches(blocks, stop_on_last=True)
        new_matches.extend(page_matches)
        if stop:
```

```
            break
        time.sleep(1)

if new_matches:
    new_matches = list(reversed(new_matches))

if last_date:
    before_filter = len(new_matches)
    new_matches = [
        m for m in new_matches
        if m[0] > last_date
        or (m[0] == last_date and (m[1], m[2], m[6]) != (last_team1, last_team2,
last_tournament))
    ]
    print(f"Filtering old matches: {before_filter} → {len(new_matches)}")

if not new_matches:
    print("There are no new matches, the database is up to date.")
else:
    print(f"New matches found: {len(new_matches)}")
    for m in new_matches:
        try:
            cur.execute("""
                INSERT INTO matches (date, team1, team2, match_format, score1, score2,
tournament)
                VALUES (%s, %s, %s, %s, %s, %s, %s)
                ON CONFLICT (team1, team2, date, tournament) DO NOTHING
            """, m)
            print(f"Added: {m[0]} | {m[1]} vs {m[2]} | {m[6]}")
        except Exception as e:
            print(f"Insert error {m[1]} vs {m[2]}: {e}")
conn.commit()
cur.close()
conn.close()
print("DB check completed.")
```

Б.2 Текст файлу models_train.py

```
df['date'] = pd.to_datetime(df['date'])
df = df.sort_values('date').reset_index(drop=True)

def get_winner(score1, score2):
    if score1 > score2:
        return 1
    elif score2 > score1:
        return 0
    else:
        return -1

df['winner'] = df.apply(lambda row: get_winner(row['score1'], row['score2']), axis=1)
df['team1_maps_won'] = df['score1']
df['team2_maps_won'] = df['score2']

le = LabelEncoder()
all_teams = pd.concat([df['team1'], df['team2']]).unique()
le.fit(all_teams)
df['team1_enc'] = le.transform(df['team1'])
df['team2_enc'] = le.transform(df['team2'])

last_n = 5
team_history = {}
h2h_counts = {}
team_maps_diff = {}
team1_form, team2_form, h2h_feat = [], [], []
team1_maps_diff, team2_maps_diff = [], []

for idx, row in df.iterrows():
    t1, t2 = row['team1'], row['team2']
    key = frozenset([t1, t2])

    hist1 = team_history.get(t1, [])
```

```
hist2 = team_history.get(t2, [])
form1 = np.mean(hist1[-last_n:]) if hist1 else 0.0
form2 = np.mean(hist2[-last_n:]) if hist2 else 0.0

pair = h2h_counts.get(key, {t1: 0, t2: 0})
h2h_val = pair.get(t1, 0) - pair.get(t2, 0)

maps_diff1 = team_maps_diff.get(t1, 0)
maps_diff2 = team_maps_diff.get(t2, 0)

team1_form.append(form1)
team2_form.append(form2)
h2h_feat.append(h2h_val)
team1_maps_diff.append(maps_diff1)
team2_maps_diff.append(maps_diff2)

if row['winner'] == 1:
    result_t1 = 1
    result_t2 = -1
elif row['winner'] == 0:
    result_t1 = -1
    result_t2 = 1
else:
    result_t1 = result_t2 = 0

team_history.setdefault(t1, []).append(result_t1)
team_history.setdefault(t2, []).append(result_t2)

team_maps_diff[t1] = team_maps_diff.get(t1, 0) + (row['score1'] - row['score2'])
team_maps_diff[t2] = team_maps_diff.get(t2, 0) + (row['score2'] - row['score1'])

if key not in h2h_counts:
    h2h_counts[key] = {t1: 0, t2: 0}
if row['winner'] == 1:
    h2h_counts[key][t1] += 1
```

```
    elif row['winner'] == 0:
        h2h_counts[key][t2] += 1

df['team1_form'] = team1_form
df['team2_form'] = team2_form
df['h2h'] = h2h_feat
df['team1_maps_diff'] = team1_maps_diff
df['team2_maps_diff'] = team2_maps_diff

features = ['team1_enc', 'team2_enc', 'team1_form', 'team2_form', 'h2h',
'team1_maps_diff', 'team2_maps_diff']
X = df[features]
y = df['winner']

split_idx = int(len(df) * 0.8)
X_train, y_train = X.iloc[:split_idx], y.iloc[:split_idx]
X_test, y_test = X.iloc[split_idx:], y.iloc[split_idx:]

lr = LogisticRegression(max_iter=2000)
lr.fit(X_train, y_train)

xgb = XGBClassifier(eval_metric='logloss')
xgb.fit(X_train, y_train)

cb = CatBoostClassifier(iterations=300, verbose=0, cat_features=['team1_enc',
'team2_enc'])
cb.fit(X_train, y_train)

for name, model in [('Logistic Regression', lr), ('XGBoost', xgb), ('CatBoost', cb)]:
    y_pred = model.predict(X_test)
    probs = model.predict_proba(X_test)[:, 1]
    print(f"{name} Accuracy:", accuracy_score(y_test, y_pred))
    print("Logloss:", log_loss(y_test, model.predict_proba(X_test)))
    print("Brier score:", brier_score_loss(y_test, probs))
    print("-" * 40)
```

Б.3 Текст файлу model.py

```
def predict_map(team1, team2, map_name,
                players_stats_team1=None, players_stats_team2=None,
                team1_maps_wr=None, team2_maps_wr=None,
                team1_wr=None, team2_wr=None,
                team1_rank=None, team2_rank=None,
                team1_avg_age=None, team2_avg_age=None, match_id=None, conn=None,
cur=None):
    print(f"[DEBUG] predict_map: {team1} vs {team2}, map: {map_name}")
    print(f"[DEBUG] team_wr: {team1_wr} vs {team2_wr}")
    print(f"[DEBUG] team_rank: {team1_rank} vs {team2_rank}, avg_age:
{team1_avg_age} vs {team2_avg_age}")

    def weighted_metrics(players_stats):
        if not players_stats:
            return {
                "rating": 1.0, "round_swing": 0.0, "dpr": 1.0,
                "kast": 0.0, "multi_kill": 0.0, "adr": 50.0, "kpr": 0.1,
                "best_rating": 1.0
            }

        sorted_players = sorted(players_stats, key=lambda p: p.get("rating", 1.0),
reverse=True)

        player_weights = [0.5, 0.3, 0.1, 0.07, 0.03]

        def weighted_avg(key, default=0.0):
            return sum(p.get(key, default) * w for p, w in zip(sorted_players,
player_weights))

        metrics = {
            "rating": weighted_avg("rating") * 60,
            "round_swing": weighted_avg("round_swing") * 50,
            "dpr": weighted_avg("dpr") * 50,
```

```
            "kast": weighted_avg("kast"),
            "multi_kill": weighted_avg("multi_kill") * 5,
            "adr": weighted_avg("adr"),
            "kpr": weighted_avg("kpr") * 50,
            "best_rating": sorted_players[0].get("rating", 1.0) * 60
        }
        return metrics

    metrics1 = weighted_metrics(players_stats_team1)
    metrics2 = weighted_metrics(players_stats_team2)

    map_wr1 = team1_maps_wr.get(map_name.lower(), 50) if team1_maps_wr and
map_name else 50
    map_wr2 = team2_maps_wr.get(map_name.lower(), 50) if team2_maps_wr and
map_name else 50

    team_wr1 = team1_wr if team1_wr is not None else 50
    team_wr2 = team2_wr if team2_wr is not None else 50

    weights = {
        "rating": 0.3,
        "round_swing": 0.2,
        "dpr": 0.03,
        "kast": 0.06,
        "multi_kill": 0.03,
        "adr": 0.03,
        "kpr": 0.03,
        "map_wr": 0.12,
        "team_wr": 0.1,
        "rank": 0.2,
        "avg_age": 0.01
    }

    def rank_factor(rank1, rank2):
        if rank1 is None or rank2 is None:
```

```
        return 0, 0

    gap = rank2 - rank1

    base1 = 120 / math.log(rank1 + 1)
    base2 = 120 / math.log(rank2 + 1)



    bonus1, bonus2 = base1, base2
    if gap > 0:

        factor = 1 + (gap / 10) ** 1.5
        if gap > 0:
            bonus1 *= factor
            bonus2 /= factor
        else:
            factor = 1 + (abs(gap) / 10) ** 1.5
            bonus2 *= factor
            bonus1 /= factor

    return bonus1, bonus2

def age_factor(age):
    if age is None:
        return 80
    return 100 - abs(age - 24.25) * 2

f_rank1, f_rank2 = rank_factor(team1_rank, team2_rank)

score1 = sum(metrics1[k] * weights.get(k, 0) for k in metrics1) + \
         map_wr1 * weights["map_wr"] + team_wr1 * weights["team_wr"] + \
         f_rank1 * weights["rank"] + age_factor(team1_avg_age) * weights["avg_age"]

score2 = sum(metrics2[k] * weights.get(k, 0) for k in metrics2) + \
         map_wr2 * weights["map_wr"] + team_wr2 * weights["team_wr"] + \
```

```
            f_rank2 * weights["rank"] + age_factor(team2_avg_age) * weights["avg_age"]

    score1 += map_wr1 * 0.1
    score2 += map_wr2 * 0.1

    score1 += metrics1["rating"] * 0.05
    score2 += metrics2["rating"] * 0.05

    print(f"[DEBUG] scores: {score1} vs {score2}")

    total = score1 + score2
    if total == 0:
        return 50.0, 50.0

    prob1 = round(score1 / total * 100, 2)
    prob2 = round(score2 / total * 100, 2)
    predicted_winner = team1 if prob1 >= prob2 else team2
    if match_id and conn and cur and map_name:
        cur.execute("""
            INSERT INTO map_predictions
            (match_id, map_name, team1, team2, predicted_prob1, predicted_prob2,
predicted_winner, prediction_date)
            VALUES (%s, %s, %s, %s, %s, %s, %s, NOW())
            ON CONFLICT (match_id, map_name)
            DO UPDATE SET
                team1 = EXCLUDED.team1,
                team2 = EXCLUDED.team2,
                predicted_prob1 = EXCLUDED.predicted_prob1,
                predicted_prob2 = EXCLUDED.predicted_prob2,
                predicted_winner = EXCLUDED.predicted_winner,
                prediction_date = NOW();
        """, (match_id, map_name, team1, team2, prob1, prob2, predicted_winner))
        conn.commit()

    print(f"[DEBUG] probs: {prob1}% vs {prob2}%")
```

```
    return prob1, prob2
```

Б.4 Текст файлу main.py

```
def get_matches():
    scraper = cloudscraper.create_scraper()
    live_matches = []
    upcoming_matches = []
    seen = set()
    today = datetime.now()
    urls = [("https://www.hltv.org/matches", True)] + [

(f"https://www.hltv.org/matches?selectedDate={(today+timedelta(days=d)).strftime('%
Y-%m-%d')}", False)
        for d in range(1, 7)
    ]

    map_list = ["dust2","inferno","nuke","mirage","ancient","train","overpass","vertigo"]

    for url, include_live in urls:
        resp = scraper.get(url, timeout=15)
        resp.raise_for_status()
        soup = BeautifulSoup(resp.text, "html.parser")

        if include_live:
            for mw in soup.select(".liveMatches .match-wrapper"):
                tournament_el = mw.select_one(".match-event")
                tournament = tournament_el["data-event-headline"] if tournament_el and
tournament_el.has_attr("data-event-headline") else "Unknown"
                teams = mw.select(".match-teamname")
                if len(teams) != 2:
                    continue
                team1, team2 = [t.get_text(strip=True) for t in teams]
                link_el = mw.select_one("a[href*='/matches/']")
                match_link = "https://www.hltv.org" + link_el['href'] if link_el else None
```

```
            key = f"{team1}_{team2}_{tournament}"
            if key in seen:
                continue
            seen.add(key)
            time_el = mw.select_one(".match-time")
            if time_el and time_el.has_attr("data-unix"):
                timestamp = int(time_el["data-unix"]) / 1000
                match_time = datetime.fromtimestamp(timestamp).strftime("%Y-%m-%d
%H:%M")
            else:
                match_time = None

            maps = []
            upcoming_maps = []

            try:
                if match_link:
                    match_resp = scraper.get(match_link, timeout=15)
                    match_resp.raise_for_status()
                    match_soup = BeautifulSoup(match_resp.text, "html.parser")

                    for mholder in match_soup.select(".mapholder"):
                        map_name_el = mholder.select_one(".mapname")
                        if not map_name_el:
                            continue
                        map_name = map_name_el.get_text(strip=True)
                        results = mholder.select(".results-left .results-team-score, .results-right
.results-team-score")
                        if results:
                            scores = [r.get_text(strip=True) for r in results]
                            if all(s == "-" for s in scores):
                                status = "upcoming"
                                upcoming_maps.append(map_name)
                            else:
                                status = "played"
```

```
                    else:
                        status = "upcoming"
                        upcoming_maps.append(map_name)

                    maps.append({"map": map_name, "status": status})
            except Exception:
                maps = []
                upcoming_maps = []

            live_matches.append({
                "tournament": tournament,
                "team1": team1,
                "team2": team2,
                "format": "Live",
                "status": "live",
                "match_link": match_link,
                "maps": maps,
                "upcoming_maps": upcoming_maps,
                "match_time": match_time
            })
    for ew in soup.select(".matches-event-wrapper"):
        tournament_el = ew.select_one(".event-headline-text")
        tournament = tournament_el.get_text(strip=True) if tournament_el else
"Unknown"
```

ДОДАТОК В

Слайди презентації

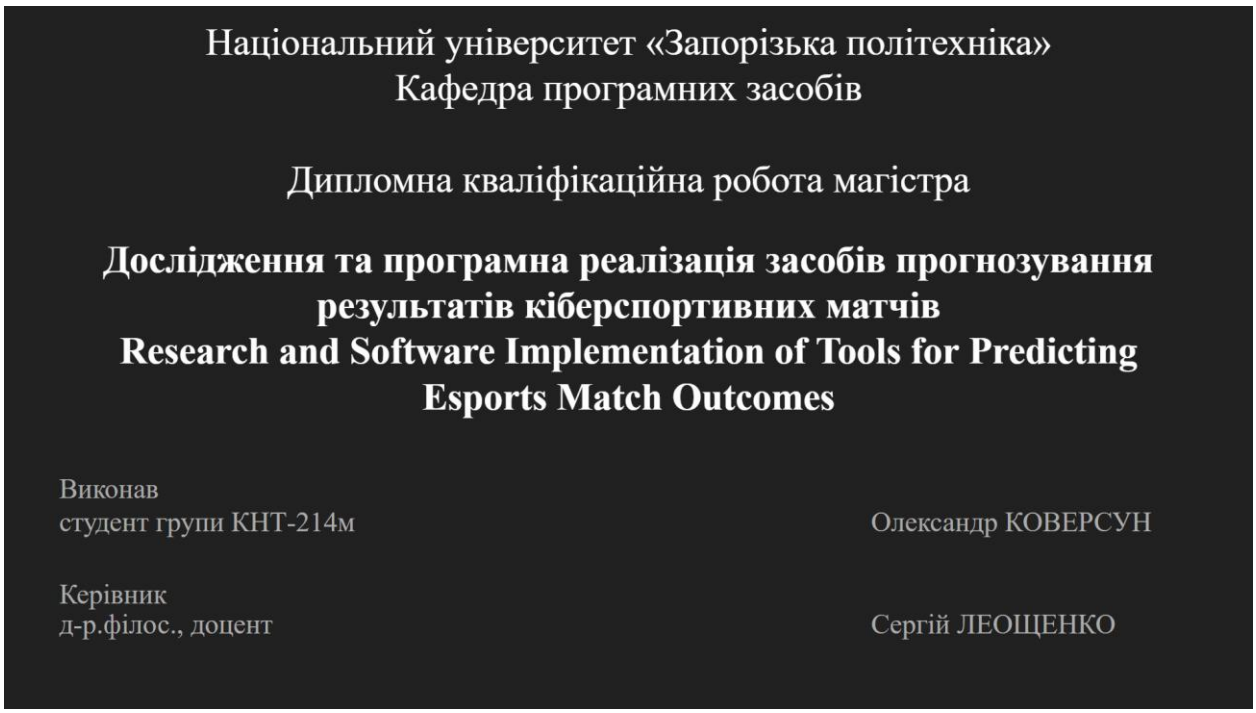


Рисунок В.1 – Вступний слайд

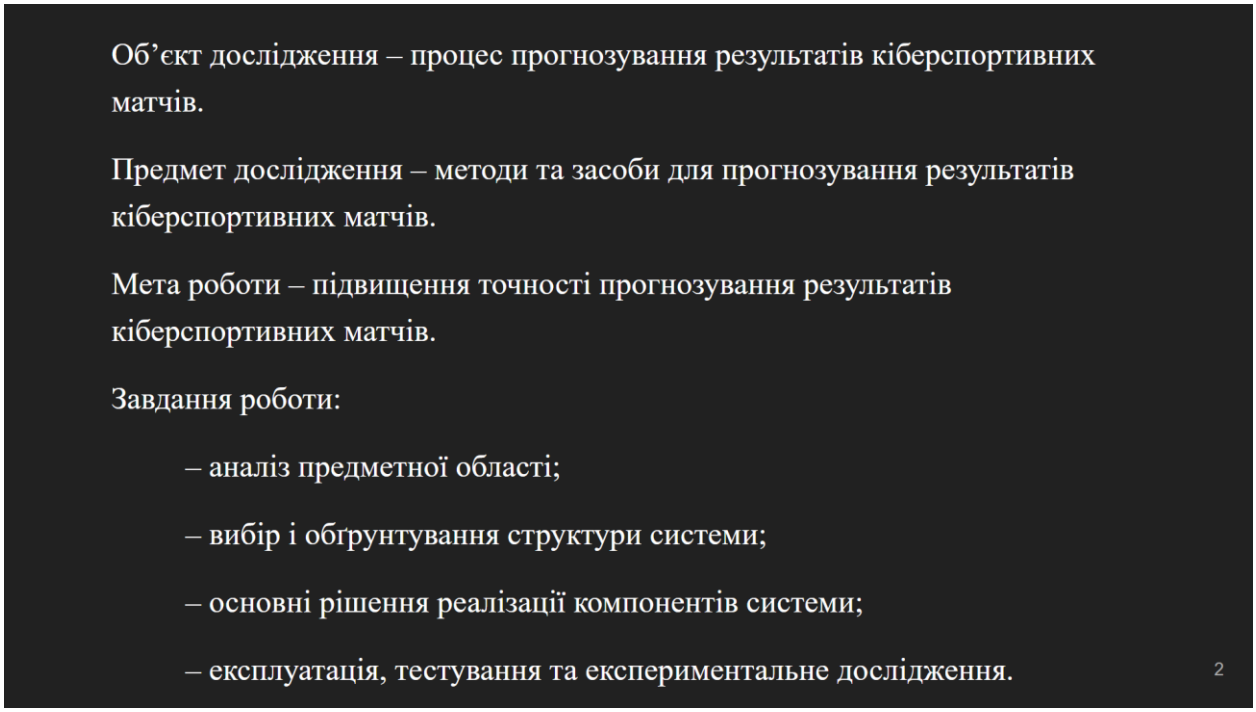


Рисунок В.2 – Слайд об’єкту, предмету дослідження, завдань та мети роботи

Порівняння програмних продуктів

Критерій порівняння	Сайти з аналітикою (hltv.org, bo3.gg)	FACEIT CS2 Predictor	Point516 Github проєкт
Тип	Вебсайти	Браузер-розширення	Відкритий код
Наявність інтерактивного інтерфейсу	Немає	Немає	Потрібен окремий деплой, більше під розробників
Вибір матчів вручну	Немає	Немає	+-, можливо вставляти посилання на матч
Застосовані технології	Статистична аналітика та моделі	Логістрична регресія	Алгоритми машинного навчання
Пояснення прогнозування	Лише в аналітичному вигляді	Немає	Є, прозоре
Автоматичний збір даних	Є, але дані недоступні користувачу	+	+
Вартість	Безкоштовно	Безкоштовно	Безкоштовно

3

Рисунок В.3 – Слайд порівняння програмних продуктів

Порівняння мов програмування

Критерій порівняння	Python	JavaScript	Java	C#
Швидкість виконання коду	Середня	Висока	Висока	Висока
Зручність розробки	Висока	Середня	Середня	Середня
Швидкість розробки веб	Дуже висока	Висока, але фронт окремо	Середня	Середня
Інтеграція з БД	Легка	Легка	Легка	Легка
Доступність бібліотек для ML	Велика різноманітність	Середня різноманітність	Середня різноманітність, небагато прикладів	Середня різноманітність, небагато прикладів
Кросплатформність	Повна	Повна	Повна	Часткова
Підтримка розробниками	Висока	Висока	Висока	Висока
Підтримка асинхронності	Висока	Висока	Середня	Висока

4

Рисунок В.4 – Слайд порівняння мов програмування

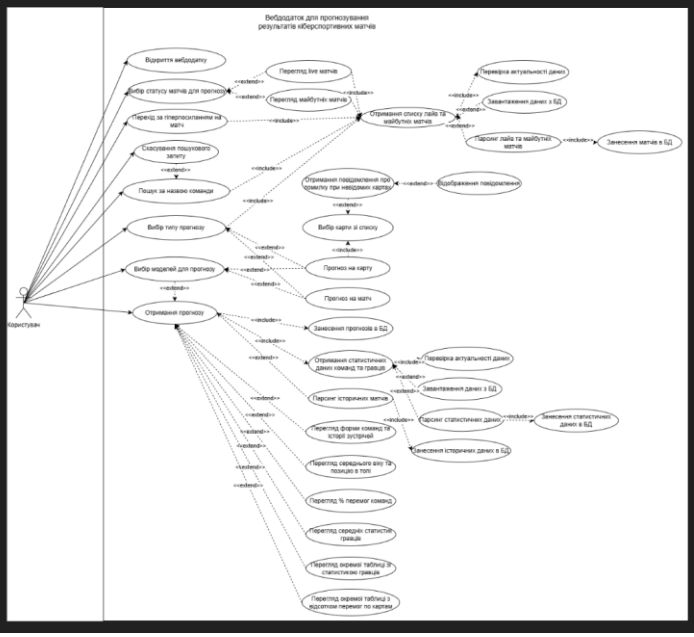
Порівняння фреймворків Python

Критерій порівняння	Streamlit	Flask	Django
Зручність використання	Висока	Середня	Висока
Зручність розробки	Висока	Середня	Середня
Актуальність	Висока	Висока	Висока
Потреба в бекенді	Не потрібен	Потрібно створювати сервер самостійно	Не потрібен, але UI та ML інтеграцію потрібно налаштовувати
Можливості UI	Середні	Середні	Високі
Розгортання	Дуже просте	Середнє	Просте, добре масштабується

5

Рисунок В.5 – Слайд порівняння фреймворків Python

Діаграма прецедентів



6

Рисунок В.6 – Слайд діаграми прецедентів

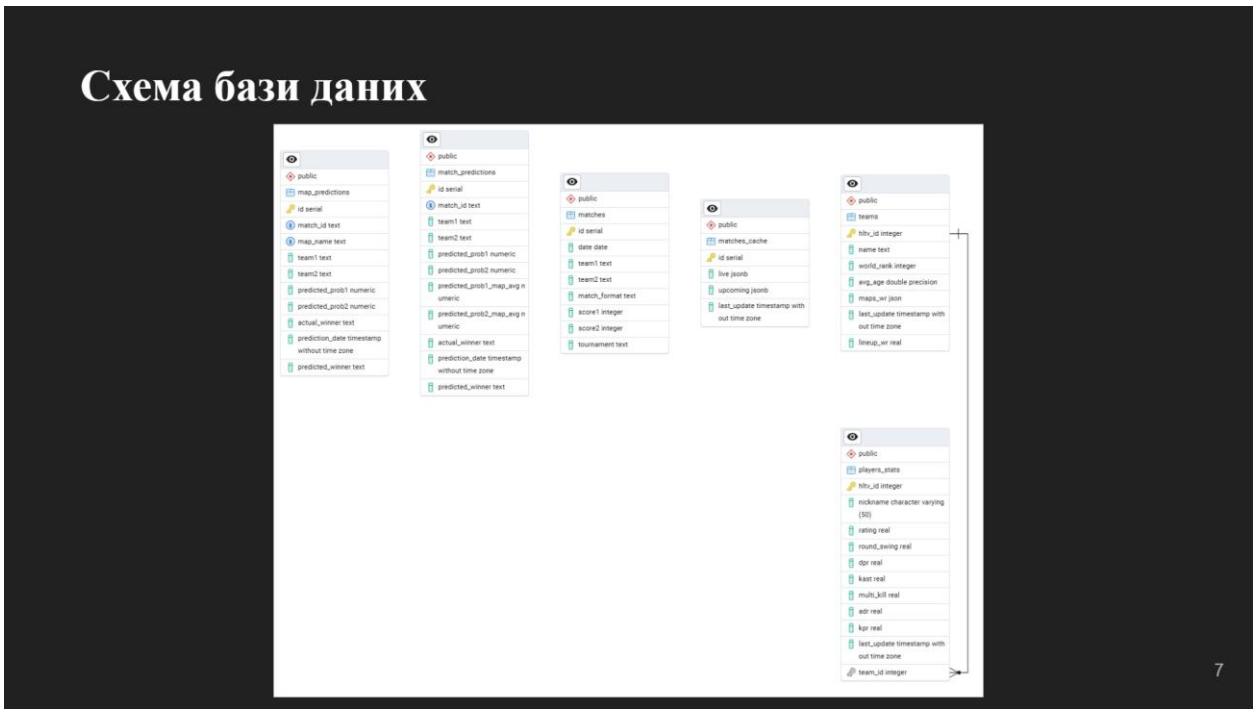


Рисунок В.7 – Слайд схеми бази даних

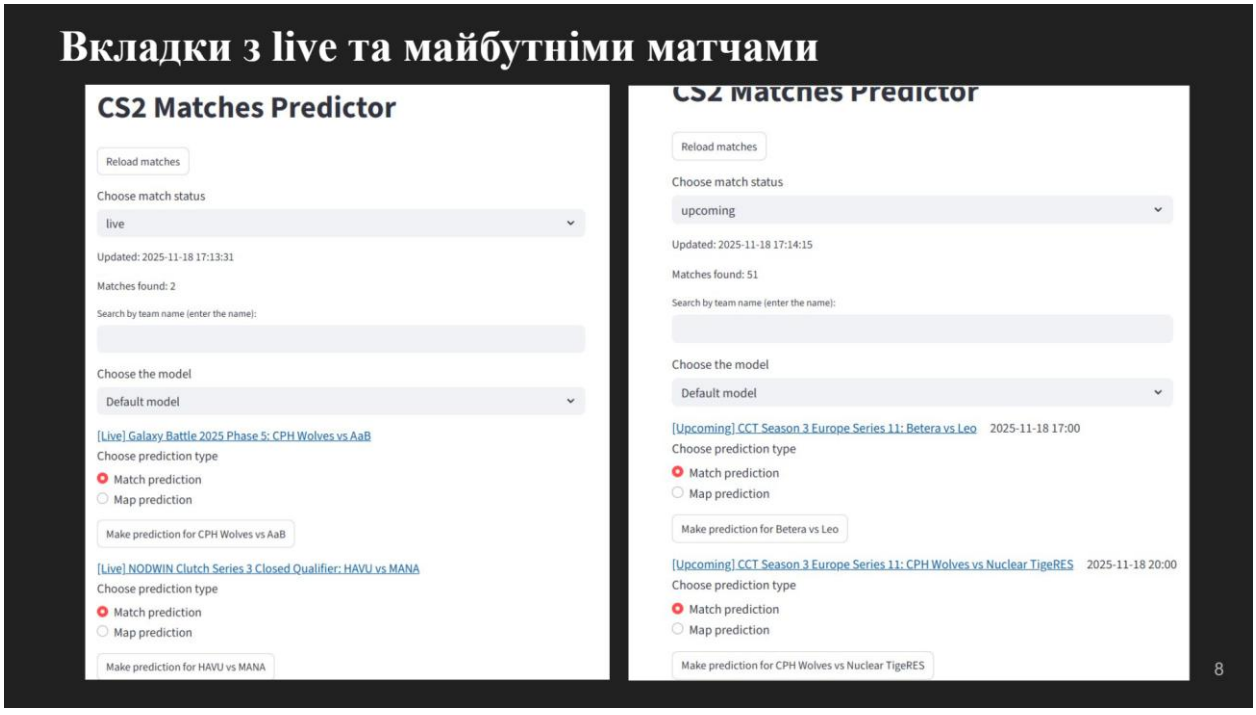


Рисунок В.8 – Слайд вкладок з live та майбутніми матчами

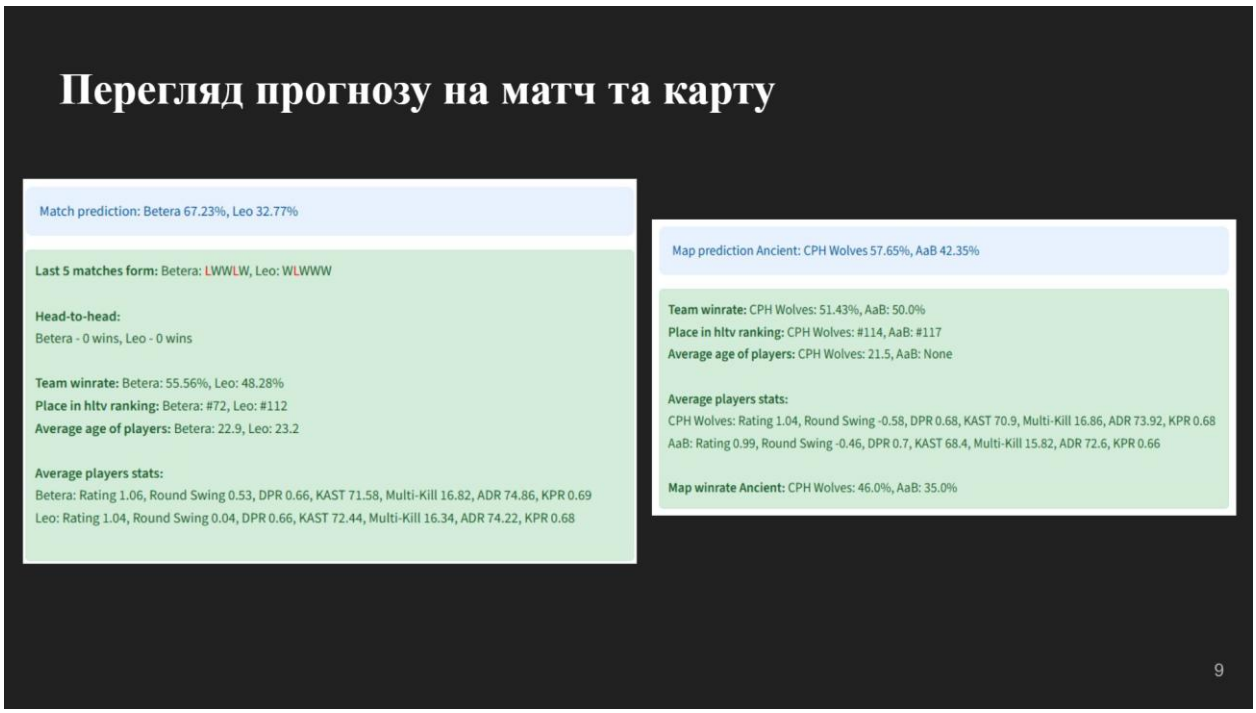


Рисунок В.9 – Слайд перегляду прогнозу на матч та карту

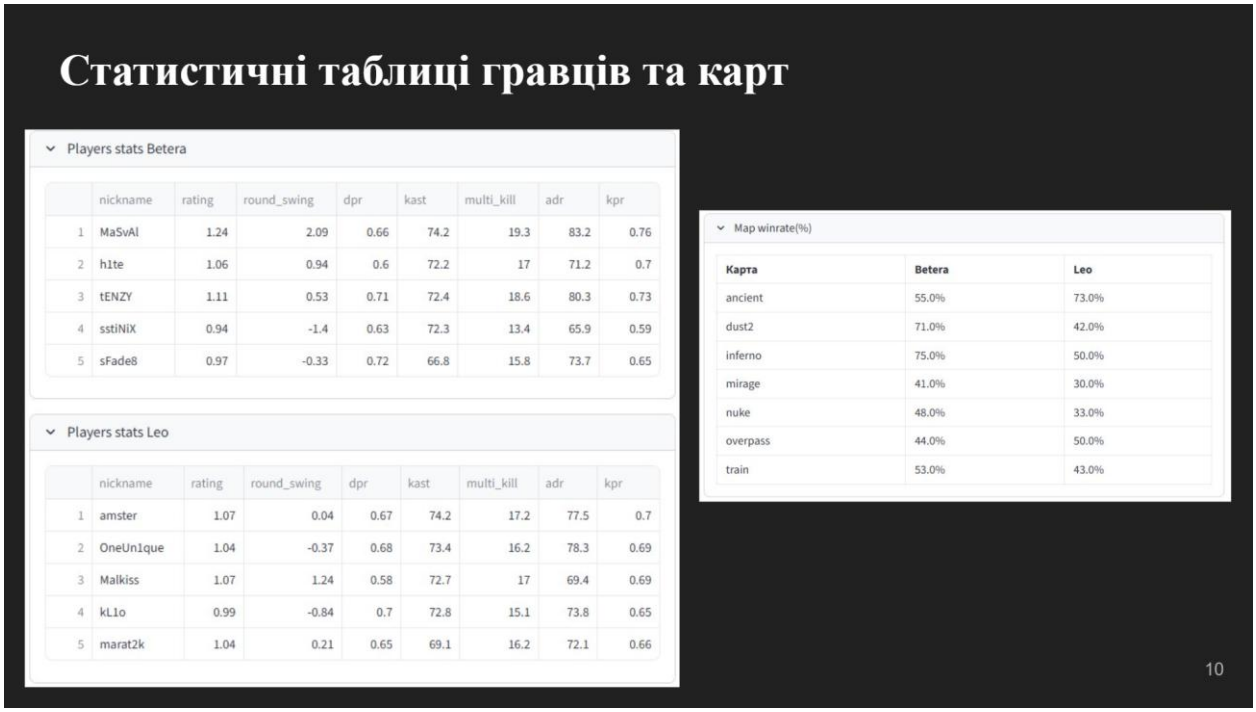


Рисунок В.10 – Слайд статистичних таблиць гравців та карт

Рисунок В.11 – Слайд зіставлення прогнозів з фактичним переможцем

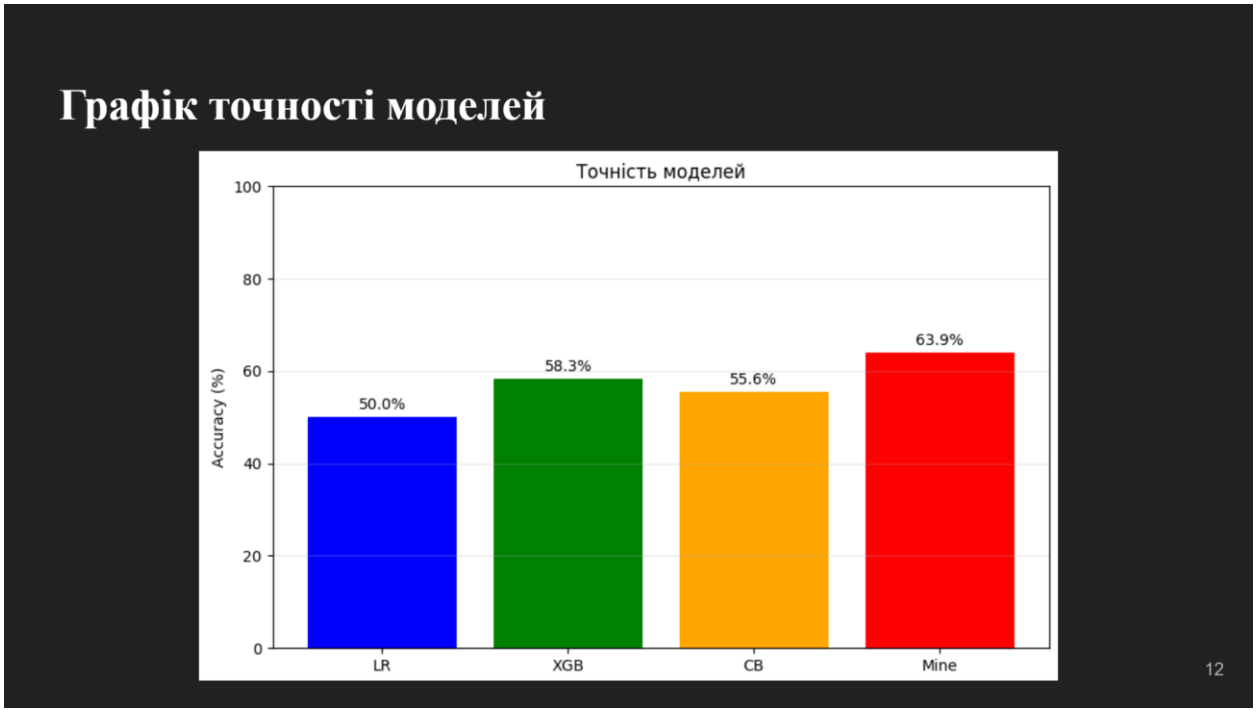


Рисунок В.12 – Слайд точності моделей

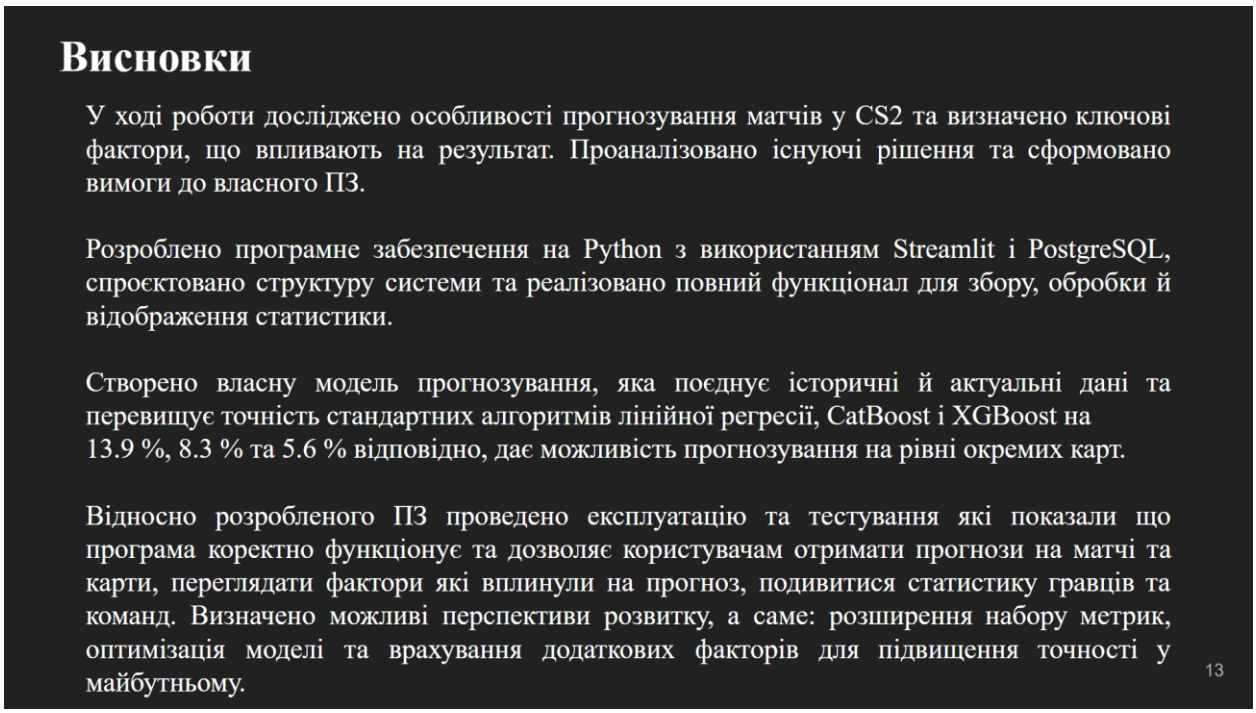


Рисунок В.13 – Слайд висновків