

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПРОГНОЗУВАННЯ СПОРТИВНОЇ СТАТИСТИКИ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
FORECASTING SPORTS STATISTICS

Виконав: студент(ка) 2 курсу, групи КНТ-129м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Зінов'єв С.Д.

(прізвище та ініціали)

Керівник Гофман Є.О.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20__ року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Зінов'єва Станіслава Дмитровича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів прогнозування спортивної статистики. Research and Software Implementation of Methods for Forecasting Sports Statistics

керівник проєкту (роботи) Гофман Євген Олександрович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “3” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 5 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми. 5. Експлуатація, тестування та експериментальне дослідження програми. 6. Економіко-організаційна частина. 7. Охорона праці та безпека в надзвичайних ситуаціях

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Гофман Є. О, доцент		
Економіко-організаційна частина	Пожуєва Т.О., професор		
Охорона праці та безпека в надзвичайних ситуаціях	Коробко О.В., ст. викладач		
Нормоконтролер	Калініна М.В., асистент		

7. Дата видачі завдання “ 4 ” вересня 2020 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 та постановка завдань дослідження	2–3 тижні	Розділ 1
3	Розробка методу вирішення задачі	4–5 тижні	Розділ 2
4	Розробка архітектури програми	6 тиждень	Розділ 3
5	Розробка програми	7–8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення	9 тиждень	Розділ 5
7	Економіко-організаційна частина	10 тиждень	Розділ 6
8	Охорона праці та безпека в надзвичайних ситуаціях	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування	14–15 тижні	
11	Захист роботи	16 тиждень	

Студент(ка)

(підпис) Зінов'єв С.Д
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Гофман Є.О
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
137 с., 51 рис., 14 табл., 3 дод., 49 джерел.

ПРОГНОЗУВАННЯ СПОРТИВНИХ ЗМАГАНЬ, НЕЙРОННІ
МЕРЕЖІ, ДОСЛІДЖЕННЯ, TENSORFLOW, KERAS.

Об'єкт дослідження – процес прогнозування спортивної статистики та даних.

Предмет дослідження – методи прогнозування спортивної статистики.

Мета роботи – розробка програмного забезпечення прогнозування спортивної статистики для підвищення якості приймаємих рішень.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, бібліотека для машинного навчання Tensorflow, мова програмування Python, Typescript, персональний комп'ютер з процесором Intel Core i5-8300U під управлінням операційної Windows 10.

Актуальність роботи пояснюється тим, що програмно реалізовані методи будуть виконувати прогнозування, яке є основою для прийняття рішень під час участі футбольних команд у спортивних змаганнях, що в свою чергу впливає на результати, які досягає команда.

Результати: розроблено програмну систему, у вигляді вебзастосунку, з використанням запропонованого методу прогнозування спортивної статистики на основі багатошарового перцептрона.

Висновки: розроблено програмну систему, що реалізує класичні методи прогнозування, а також реалізацію запропонованого методу прогнозування. Система має графічний інтерфейс користувача для проведення прогнозувань, що зменшує витрати часу спеціаліста на проведення аналізу.

Галузь використання – спортивно-аналітичні сервіси, комп'ютерні ігри.

ABSTRACT

Explanatory note to the final qualifying work of the master:
137 pages, 51 figures, 14 tables, 3 appendixes, 49 sources.

PREDICTIONS SPORT COMPETITIONS, NEURAL NETWORK,
RESEARCH, TENSORFLOW, KERAS.

Object of study is the process of prediction sports statistics and data.

Subject of research: methods of prediction the results of sports competitions.

The purpose of the work is development of a software application for predictions the results of sports competitions based on statistical data to improve quality of decisions making.

Materials, methods, and tools: structured and object-oriented programming, machine-learning library Tensorflow, programming language: Python, Typescript, personal computer with an Intel Core i5-8300U processor under the control of the Microsoft Windows 10 operating system.

Results. The WEB-based application is created which based on improvement machine-learning algorithm of MLP for predictions sports statistics.

Conclusions A software system is created that implements the classical methods of forecasting by simple neural networks, as well as the implementation of the proposed method of forecasting. The system has a graphical user interface for forecasting, which reduces the time spent by the specialist to conduct the analysis.

The field of use is a sphere that are connected to sports, like: sports-analytics services, gaming industry.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Аналіз предметної області та постановка завдань дослідження.....	12
1.1 Огляд області «аналіз та прогнозування спортивних даних»	12
1.2 Використання НМ для прогнозування спортивних даних	13
1.3 Основи використання перцептрона.....	15
1.4 Особливості використання багат шарового перцептрона	22
1.5 Постановка завдання.....	26
1.6 Висновки за розділом 1	27
2 Матеріали і методи.....	28
2.1 Задача прогнозування позиції футболіста на полі.....	28
2.2 Аналіз даних	29
2.3 Використання статистичних методів	30
2.4 Запропонований метод прогнозування позиції футболіста на полі	34
2.4.1 Етап 1. Підготовка даних.....	35
2.4.2 Етап 2. Відбір гіперпараметрів	37
2.4.3 Етап 3. Відображення результатів.....	40
2.5 Висновки за розділом 2	41
3 Проєктування програмного забезпечення	42
3.1 Вибір засобів реалізації програмного забезпечення	42
3.2 Вибір середовища розробки.....	47
3.3 Опис основних вимог до програми	49
3.4 Проєктування бази даних	50
3.5 Структура програмного забезпечення	51
3.6 Вимоги до технічних засобів	52
3.7 Висновки за розділом 3	53
4 Основні рішення щодо реалізації компонентів системи.....	54
4.1 Архітектура інтерфейсу.....	54

4.2 Архітектура серверної частини	58
4.3 Модулі обробки даних та нейронних мереж.....	61
4.4 Висновки за розділом 4	62
5 Експлуатація та тестування системи.....	63
5.1 Опис застосування програми.....	63
5.2 Інструкція по запуску системи	63
5.3 Методика роботи з вебплатформою.....	64
5.4 Методика тестування системи.....	66
5.4.1 Тестування моделі.....	66
5.4.2 Тестування серверної частини.....	68
5.5 Висновки за розділом 5	68
6 Економіко-організаційна частина.....	69
6.1 Концепція економічного обґрунтування	69
6.2 Планування розробки програмного продукту	70
6.3 Розрахунок основної заробітної плати	72
6.4 Розрахунок додаткової заробітної плати	72
6.5 Єдиний соціальний внесок.....	73
6.6 Визначення витрат на матеріали	73
6.7 Витрати на спеціальне обладнання	74
6.8 Інші прямі витрати	78
6.9 Розрахунок накладних витрат.....	79
6.10 Розрахунок економічної ефективності програмного продукту.....	80
7 Охорона праці та безпека в надзвичайних ситуаціях.....	83
7.1 Заходи забезпечення безпеки.....	84
7.2 Заходи з забезпечення виробничої санітарії та гігієни праці	87
7.3 Заходи з пожежної безпеки	92
7.4 Сили і засоби, які залучаються для проведення рятувальних та інших невідкладних робіт.....	95
Висновки	99
Перелік джерел посилення	101

Додаток А Технічне завдання	107
Додаток Б Текст програми	112
Додаток В Слайди презентації.....	130

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- API – Application Programming Interface;
- HTML – HyperText Markup Language;
- LESS – Leaner Style Sheets,
- ORM – Object-Relational Mapping;
- ReLU – Функція «Випрямляч»;
- REST – Representational State Transfer;
- SASS – Syntactically Awesome Stylesheets;
- БШП – багатошаровий перцептрон;
- МН – машинне навчання;
- НМ – нейрона мережа;
- НМПП – нейроні мережі прямого поширення;
- ПК – персональний комп’ютер;
- ШН – штучний нейрон.

ВСТУП

У сучасному світі використання нейронних мереж (НМ) широко застосовується у багатьох сферах. Спортивні змагання не є винятком і в наш час розробка різних методів аналізу спортивних даних є актуальним завданням.

Сучасний спортивний світ досить активно впроваджує наукові підходи у багатьох напрямках, спортивні клуби мають власні відділи аналітиків та фахівців, що працюють над вдосконаленням та покращенням результатів.

Окрім самих учасників спортивного світу, численним є попит з боку розробників спортивних симуляторів. Сучасні комп'ютерні ігри, присвячені спорту намагаються максимально відтворити правдоподібність реальної гри. За допомогою сучасних методів штучного інтелекту, розробникам вдається з кожним роком наближатися до відтворення реальних дій гравців.

Також, важливим сегментом впровадження методів прогнозування на основі НМ є ігрові платформи. В сучасному світі фінансові можливості та прибутки всесвітньо відомих компаній в сфері бетінгу вимірюються мільйонами доларів.

Зважаючи на попит, засоби прогнозування на основі НМ знайшли відгук в наведених вище сферах. Побудова якісних моделей прогнозування, обробки даних не є простою задачею, оскільки базується на великих об'ємах реальних даних. З одного боку, в сучасному світі немає проблеми з отриманням даних, проте існує декілька гострих проблем при роботі з ними.

Основними викликами при використанні машинного навчання визначають наступні фактори:

- об'єм даних – впливає на такі проблеми, як необхідність залучення потужних фізичних засобів для обробки даних та навчання моделей, незбалансованість записів за поділом класів за ознаками;

- варіативність – ця проблема впливає на наступні виклики, як різні за типом та фізичними характеристиками джерела даних, кількість помилок, шумів та невпевненість у якості даних, нерепрезентативної вибірки тощо;
- проблематика вибору ознак;
- перенавчання моделі [1].

Однак, незважаючи на вище зазначені особливості роботи з НМ, вони в свою чергу мають можливість нелінійного моделювання і це робить їх затребуваними в сучасному світі. Побудова нейронних моделей при вирішенні тієї чи іншої задачі виконується за наступними кроками:

- оптимізація набору вхідних даних;
- вибір архітектури нейронної мережі;
- налаштування методів активації та оптимізації роботи мережі [2].

В магістерській роботі пропонуються метод, що використовує модифікацію багатошарового перцептрона.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

Цей розділ спрямовано на дослідження проблематики області аналізу та прогнозування спортивної статистики, доведення необхідності використання сучасних архітектур штучних нейронних мереж для розробки рішень для цієї області.

1.1 Огляд області «аналіз та прогнозування спортивних даних»

Сучасний спорт пропонує багато викликів у сфері аналізу даних та машинного навчання. Виклики пов'язані з бажанням досягати максимальних результатів, за допомогою обґрунтування тих чи інших подій, на основі математичних розрахунків. Зважаючи на незлічену кількість видів спорту та різних метрик, НМ широко використовуються для різних задач. Основою для них виступає передбачення тієї чи іншої події. В магістерській роботі запропонований метод буде стосуватися світу футболу, тому приклади використання НМ будуть розглянуті саме на цьому виді спорту.

Для початку, потрібно окреслити сфери застосування НМ у світі футболу, а саме:

- командна сфера (використання тренерами, командами, аналітичними центрами) – аналіз та робота у реальному часі з метою покращення результатів та виявлення помилок, дана сфера найбільш зрозуміла, команді потрібно покращити результати, знайти проблему в великій кількості травмованих, проаналізувати суперника, рух м'яча і т. п;
- спортивний бетінг – аналіз даних та побудова моделей прогнозування на різній кількості характеристик з метою отримання прибутку;
- комп'ютерні ігри – одними із найпопулярніших симуляторів спорту є футбольні. Аналіз даних (не тільки спортивних), а також фізичних

характеристик гравці, руху м'яча спрямоване на досягнення максимальної ідентичності з реальною грою [3]-[5].

Кожна сфера звертає увагу численної кількості спеціалістів з аналізу даних. Дуже часто, ці сфери об'єднуються, запозичуючи одна в одній щось спільне. Наприклад, відома комп'ютерна гра FIFA залучає до розробки кожної версії симулятора реальних гравців, для ретельного дослідження їх характеристик та відтворення їх у віртуальності. Або в зворотному тандемі працює FIFA Manager [6]. Якість симуляції та ігрових показників гравців спонукає реальних скаутів та аналітиків використовувати гру, як джерело знань про тих чи інших молодих талантів.

Принципи прогнозування подій або показників залежать виключно від типу задачі. Для прогнозування спортивних змагань, потрібно робити акцент на якість даних, щоб джерело було підтвердженим та постановку задачі для вибору методів прогнозування.

1.2 Використання НМ для прогнозування спортивних даних

Використання НМ для задач в сфері спортивного аналізу набуло широкого попиту в багатьох напрямках. До основних переваг НМ відносять гнучкість до різних видів даних, здатності до модифікацій та налаштування. Самі собою НМ – це обчислювальні системи, які взяли за основу процеси, подібні до нейронних систем живих організмів. НМ представляють собою велику кількість однотипних елементів, що називаються нейронами, кожен з яких виконує свої функції. Ідея про те, що нейрони можуть виконувати складні розрахунки з використанням логіки висловлювань, було запропоновано в 1943 році [7].

Зважаючи на ці знання при роботі з НМ, з'явилося поняття навчання нейронної мережі або моделі. Головна мета цього процесу подібна біологічному процесу навчання, наприклад людини. Навчання полягає в тому, щоб представити нейронній мережі набір даних та повідомити, що саме ми

хочемо досягти в результаті. Системи навчаються на основі прикладів та послідовно покращуючи свою точність в прийнятті рішень. Основним аспектом є те, що НМ роблять не володіючи апіорними знаннями, а подібно людському мозку, набувають його та формують власний набір необхідних характеристик з навчальних прикладів, які вони оброблюють.

Будь-яка НМ ґрунтується на з'єднаннях нейронів, ці з'єднання виконують роль подібну синапсам. Біологічні нейрони викликають короткий електричний імпульс (потенціал дії), який подорожує по аксонам, провокуючи створення хімічних реакцій, які в свою чергу при достатній кількості збуджують нейрон до власного електричного імпульсу. Подібний процес відносно простий, проте зважаючи на незлічену кількість нейронів придатний для використання в складних обчисленнях [8]. Подібний принцип лежить в основі НМ, штучний нейрон отримує сигнал, обробляє та сигналізує іншим.

Класична НМ представлена на рис. 1.1.

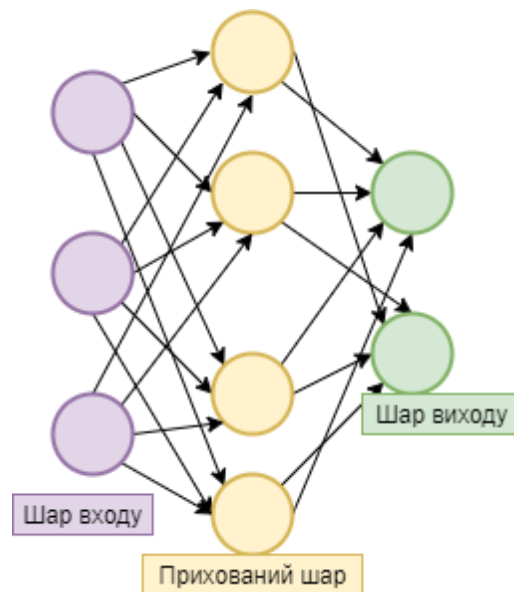


Рисунок 1.1 – Штучна нейронна мережа

В штучних НМ, нейрон це математична функція, яка отримую декілька даних на вхід («шар входу»), кожне вхідне значення позначається , як x_k , де k – це індекс елемента. З'єднання між нейронами (синапси) в свою чергу має параметр вага – w_k . Завдяки цьому параметру, початкові дані змінюються при

передачі від одного нейрону до іншого. Зазвичай, нейрони об'єднуються в шари, різні шари можуть мати свої властивості там впливати на результат сигналу. Після проходження всіх шарів, формується шар виходу.

Розглянемо основні можливі напрямки застосування НМ в світі спорту:

- застосування НМ для прогнозування кількісних характеристик спортсменів та команд при аналізі подій (матчу, гри тощо). Знаходження математичних моделей для обґрунтування тих чи інших результатів спортсмена. Наприклад з використанням НМ була створена нова метрика «очікуваних голів» (expected goal) [9]. На основі статистичних даних ударів, голів та їх характеристик було побудовано декілька моделей для аналізу матчів, що набули популярності у спортивному світі;

- застосування НМ для прогнозування економічних показників. Прикладом є побудування моделей регресії для прогнозування вартості гравців;

- застосування НМ для симулювання спортивних подій, прикладом виступають комп'ютерні ігри;

- застосування НМ при підготовці спортсменів та спостереженнями за станом їх здоров'я. Інтенсивність навантажень впливає на ризик отримання ушкоджень спортсменом, тому велика кількість видів спорту звернули увагу на використання моделей штучних мереж, для передбачення травм.

1.3 Основи використання перцептрона

Перцептрон – один з найпоширеніших представників НМ, який базується на використанні спрощеної моделі природнього нейрона, яка називається «штучний нейрон». Математично ШН зазвичай представляють як деяку нелінійну функцію від єдиного аргументу – лінійної комбінації всіх вхідних сигналів [10].

Вхідними та вихідними даними для перцептрона є числові значення, а кожний вхідний зв'язок має вагу. Вихід ШТ визначається через формули (1.1), (1.2):

$$y = f(u), \quad (1.1)$$

$$u = \sum_{i=1}^n w_i x_i + w_0 x_0, \quad (1.2)$$

де x_i – вхідні мережі;

x_0 – значення ініціалізації;

w_0 – значення ініціалізації;

w_i – ваги входу мережі;

$f(u)$ – передавальна функція (або функція активації).

Значення сигналів вхідних сигналів вважають заданими на інтервал $[0, 1]$. Вони можуть бути або дискретними, або аналоговими. Схематично алгоритм представлено на рис. 1.2.

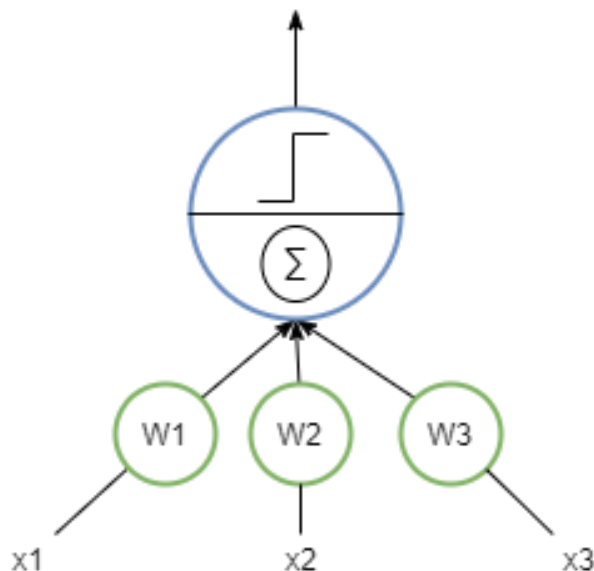


Рисунок 1.2 – Математичний алгоритм перцептрона

1.3.1 Опис функцій активації

Функція активації – це функція, що використовується у НМ, яка виводить невелике значення для малих вхідних значень, якщо її результат перевищує порогове значення. Якщо, вхідні значення досить великі, функція активації «повідомляє», в іншому випадку вона нічого не робить. Функція активації в простому розумінні схожа на шлюз або фільтр, який перевіряє, що вхідне значення більше критичного числа.

Розглянемо можливі типи передавальної функції.

Функція Гевісайда (або «крокова» функція) – це розривна функція дійсної змінної значення якої рівне 0 для від’ємних значень аргументу і рівне 1 для додатних. Широко використовується в теорії керування і обробці сигналів, формула (1.3):

$$H[n] = \begin{cases} 0, & n < 0 \\ 1, & n \geq 0 \end{cases}, \quad (1.3)$$

де n – ціле число.

Графічно функція представлена на рис. 1.3.

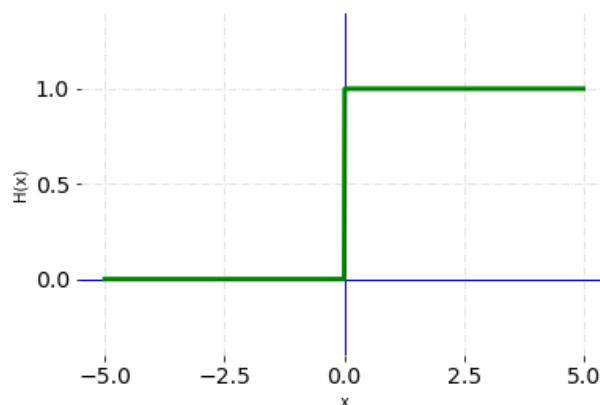


Рисунок 1.3 – Графік функції Гевісайда

Лінійна передавальна функція – є сигналом результуючого нейрона лінійно пов’язаного з ваговою сумою, формула (1.4), сигналів на його вході.

$$f(x) = tx, \quad (1.4)$$

де t — параметр функції.

В НМ із багатошаровою структурою, нейрони з передавальними функціями лінійного типу складають вхідний шар. Єдиний ШН може використовуватися для простої лінійно-бінарної класифікації.

— сигмоїда — це неперервно диференційована монотонна нелінійна S-подібна функція, яка часто застосовується для «згладжування» значень деякої величини. Головна особливість сигмоїди це робота на діапазоні значень $[0,1]$. Це допомагає будувати моделі для прогнозування імовірностей для вихідних значень. Представлено на формулі (1.5):

$$S(x) = \frac{1}{1 + e^{-x}}, \quad (1.5)$$

де x — дискретне значення.

Графічно функція зображена на рис. 1.4.

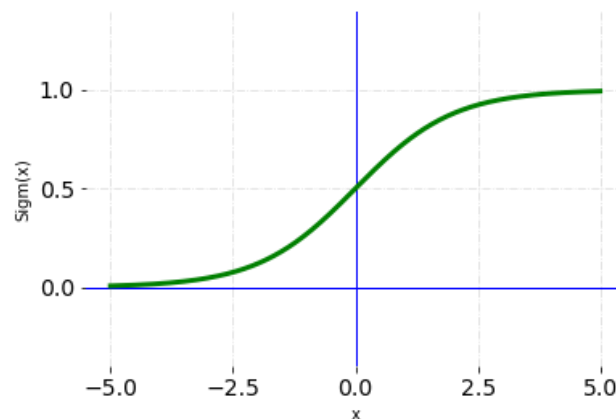


Рисунок 1.4 – «S» – подібна функція

Радіально-базисна функція — це окремий клас функцій, характерною особливістю яких, є монотонна зміна значень (зменшення або збільшення) з віддаленням від центральної точки. Значення цієї функції тим вище, чим

ближче до центру вхідний вектор. Як радіальну базисну функцію можна, наприклад, використовувати функцію Гауса, формула (1.6):

$$y = \exp\left(-\frac{(S - R)^2}{2\sigma^2}\right), \quad (1.6)$$

де $S = \|X - C\|$ – відстань між центром C і вектором вхідних сигналів;

R – визначає зсув активаційної функції по осі абсцис;

σ – скалярний параметр, який визначає швидкість спадання функції при віддаленні вектору від центру.

У реальних мережах активаційна функція може відображати розподіл імовірності будь-якої випадкової величин, або позначати будь-які евристичні залежності між величинами. Графік функції представлено на рис. 1.5.

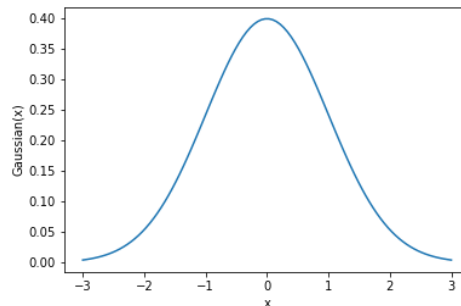


Рисунок 1.5 – Графік функції Гауса

Випрямляч (ReLU) – це нелінійна функція активації, яка використовується в багатошарових НМ, описується формулою (1.7):

$$f(x) = \max(0, x), \quad (1.7)$$

де x – вхідне дискретне значення.

Результатом функції – це максимальне значення між нулем та вхідним значенням, за формулою (1.8):

$$f(x) = \begin{cases} 0, & n < 0 \\ x, & n \geq 0 \end{cases}, \tag{1.8}$$

де x – вхідне дискретне значення.

Графічно функція представлена на рис. 1.6.

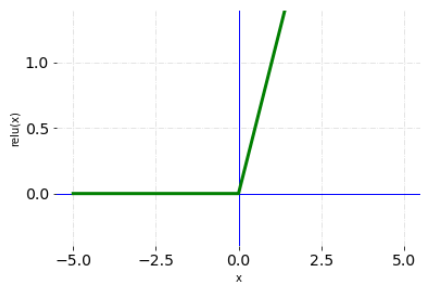


Рисунок 1.6 – Графік функції ReLu

Функція здатна прискорити швидкість навчання глибоких НМ порівняно з традиційними функціями активації, оскільки похідна ReLu дорівнює одиниці для позитивних вхідних значень. Через константне значення, НМ не потрібно витрачати додатковий час на обчислення помилок під час навчального етапу.

Проведемо систематизоване порівняння передавальних функцій, результати представлені у табл. 1.1 [11]-[15].

Таблиця 1.1 – Порівняльна таблиця передавальних функцій

Назва функції	Область значень	Переваги	Недоліки
Формула Гевісайда	{0,1}	Використовується для бінарної класифікації	Гradient дорівнює нулю
Лінійна функція	{ -∞, +∞ }	Можливо точно з'єднати кілька нейронів разом, якщо «спрацює» більше ніж один нейрон, тоді взяти максимальний.	Якщо є помилка в прогнозуванні, зміни, внесені зворотнім поширенням, є постійними і не залежать від вхідного x

Продовження таблиці 1.1

Назва функції	Область значень	Переваги	Недоліки
Сигмоїда	$(0, 1)$	Має нелінійний характер. Використовується для задач класифікації	Має проблему «зникаючих градієнтів» Мережа може «відмовлятися» вчити ся, або буде процес буде довгим
Радіально-базисна функція	$(0,1]$	Завдяки властивості стійкості до даних з «шумами» використовується для динамічних систем.	Повільність роботи в задачах класифікацій порівняно з багат шаровим перцептроном.
Випрямляч (ReLU)	$[0, +\infty)$	Дозволяє уникнути проблеми зникнення градієнта. Передбачає простіші математичні операції, що впливають на високу швидкість використання	Використовується лише прихованих шарах. Деякі градієнти можуть бути крихкими під час тренування та призвести до мертвих нейронів.

Розглянувши функції активації, перейдемо до процесу навчання перцептрона.

1.3.2 Алгоритм навчання перцептрона

Алгоритм навчання перцептрона був Ф. Розенблатом, пізніше розвинений М. Мінським та С. Пейпером в 1969 році [16]-[17]. Винайдений алгоритм базувався на правилі теорій Гебба. Згодом ідея була сформовано в наступному виразі «Між клітинами, які спрацьовують разом, встановлюється зв'язок». Це спрощення не слід сприймати буквально, воно існує для узагальнення та в розгляді НМ це значить, що вага з'єднання між двома нейронами має тенденцію до збільшення, коли вони «спрацьовують» одночасно. Правило алгоритму навчання має на меті підсилення з'єднань, які в свою чергу зменшують помилку, формула (1.9):

$$w_{i,j}^{(next\ step)} = w_{i,j} = \eta(y_j - \hat{y}_j)x_i, \quad (1.9)$$

де $w_{i,j}$ – вага між i^n вхідним нейроном та j^n вихідним нейроном;

x_i – i^n вхідне значення поточного екземпляру навчання;

$\hat{y}_j$ – вихідний сигнал j^n вихідного нейрона для поточного екземпляру навчання;

y_j – результуюче значення j^n вихідного нейрона для поточного екземпляру навчання;

η – рівень навчання.

Перцептрон подається по одному навчальному екземпляру за раз і для кожного випадку він робить свої розрахунки. Для вихідного нейрона, який в результаті дав невірний прогноз, він підсилює ваги зв'язку з вхідними, які могли би вплинути на розрахунки та призвести до правильного прогнозу.

Однією з проблем перцептрону є нездатність до вирішення часом простих задач. Наприклад, як класифікація на основі виключної диз'юнкція (XOR). Це проблема використання НМ для прогнозування логічних виходів з урахування бінарних значень. Функція повинна повертати істинне значення, якщо два вхідних значення не рівні, і хибне значення якщо рівні [18]. Дану обмеженість було вирішено за допомогою складання перцептронів. В подальшому в сфері НМ це отримало назву багатошарового перцептрона.

1.4 Особливості використання багатошарового перцептрона

Багатошаровий перцептрон (БШП) – це тип нейронних мереж прямого поширення (НМПП), який базується на алгоритмі зворотного поширення помилки. Вважається одним із основних типів НМ. Використовується переважно для побудови прототипів та перевірки гіпотез в зв'язку з простою структурою.

БП складається з одного вхідного шару, наступний або більше називаються прихованими шарами та базуються на ШН, останнім є вихідний шар. Схематично БП представлено на рис. 1.7.

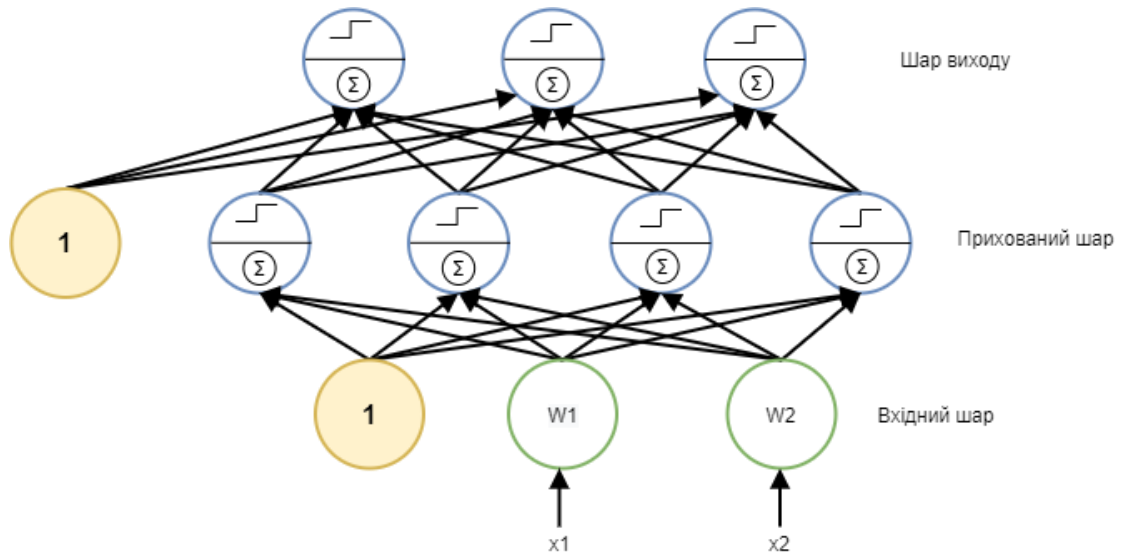


Рисунок 1.7 – Архітектура БП з явний відображенням «нейрону зміщення»

Нейрон зміщення – це нейрон вхідне та вихідне значення якого завжди дорівнюють 1 та ніколи не мають вхідних з'єднань. Це дає можливість отримати вихідний результат, шляхом зсуву графіку функції активації вправо або вліво [19].

1.4.1 Опис алгоритму та методу зворотного поширення

При навчанні БШП використовується метод зворотного поширення помилки. Метод зворотного поширення відноситься до методів навчання з вчителем, тому потребує від початкових прикладів інформацію про результуючі значення. Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилки роботи багатошарового перцептронну та отримання бажаного виходу. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в

напрямку, зворотному прямому поширенню сигналів, формула (1.10) у звичайному режимі роботи [20].

$$E = \frac{1}{2} \sum_{i=1}^k (y - y')^2, \quad (1.10)$$

де k – число вихідних нейронів мережі;

y – результуюче значення;

y' – фактичне результуюче значення.

Алгоритм складається з наступних кроків.

Крок 1. Проходить повний навчальний набір кілька разів беручи невеликий зріз даних за раз (batch), такий прохід називається «епохою».

Крок 2. Кожний такий зріз передається на вхідний рівень мережі, який передає дані до першого прихованого. Алгоритм вираховує вихід всіх нейронів цього шару (для кожного зрізу). Результат передається далі і так доки не буде отримано результат останнього шару. Це складова прямого проходу НМ. Єдиним винятком є збереження результатів для реалізації зворотного проходу.

Крок 3. Алгоритм вимірює вихідну помилку мережі, використовує функцію втрат, яка порівнює бажаний вихід і фактичний вихід мережі.

Крок 4. Потім алгоритм вимірює, скільки цих помилок надходило від кожного з'єднання в шарі на рівні нижче, використовуючи правило диференціювання складної функції.

Крок 5. Потім алгоритм вимірює, скільки цих помилок надходило від кожного з'єднання в шарі нижче, працюючи назад, доки алгоритм не досягне вхідного рівня. Корекція ваг нейронів виконуються з відповідністю до формули (1.11):

$$\Delta w_{j,i}(n) = -\eta \frac{\partial E}{\partial w_{i,j}}, \quad (1.11)$$

де $w_{j,i}$ – ваги і-того з'єднання та j-го нейрона;

η – параметр швидкості навчання, який дозволяє додатково керувати величиною кроку корекції $\Delta w_{j,i}$. Змінюється на інтервалі від 0 до 1.

Враховуючи, що вихідна сума j-го нейрону дорівнює $S_j = \sum_{i=1}^n w_{i,j} x_i$, з цього випливає наступне твердження (формула 1.9), що диференціал ∂S_j активаційної функції нейронів мережі $f(s)$ повинен існувати та не дорівнювати 0 в будь-якій точці, тобто функція активації повинна бути диференційованою на всій числовій осі $\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial S_j} \frac{\partial S_j}{\partial w_{i,j}} = x_i \frac{\partial E}{\partial S_j}$.

Крок 6. Алгоритм використовує так званий стохастичний градієнтний спуск, «просуваючись» в багатовимірному просторі ваг в напрямку антиградієнта з метою досягти мінімуму функції помилки [20].

Для того щоб, цей алгоритм працював належним чином, його автори внесли ключову зміну в архітектуру БШП. Замінили крокову функцію на сигмоподібну. Це було важливо, оскільки крокова функція містить лише плоскі відрізки, а оскільки градієнтний спуск не може рухатися по рівній поверхні, тому немає градієнта для роботи. Тоді як логістична функція має чітко визначену ненульову похідну скрізь, дозволяючи градієнтному спуску досягти певного прогресу на кожному кроці. До популярних сигмоїдальних функцій відносять логістичну, гіперболічний тангенс та «випрямляч»

До переваг алгоритму зворотного поширення помилки відносяться простота реалізації і стійкість до аномалій і викидів в даних. До недоліків можна віднести:

- невизначено довгий процес навчання;
- можливість «паралічу мережі», коли при великих значеннях робоча точка передавальної функції виявляється в області насичення сигмоїд і похідна в виразі (1) стає близькою до 0, а корекції ваг практично не відбувається і процес навчання «завмирає»;
- вразливість алгоритму до потрапляння в локальні мінімуми функції помилки.

1.4.2 Використання багат шарового перцептрона

Багат шаровий перцептрон використовується переважно для задач:

- класифікації. У класифікації метою є передбачення мітки класу, яка є вибором із заздалегідь визначеного. Класифікація іноді поділяється на бінарну класифікацію, що є особливим випадком розмежування рівно двох класів від багатокласової класифікації, тобто класифікація між більш ніж двома класами. Бінарна класифікація – це як спроба відповісти на запитання так / ні. Класифікація електронних листів як або спам, або не спам є прикладом проблеми бінарної класифікації. До багатокласової класифікації можна віднести передбачення мови тексту на веб сайті. Навчальні класи тут будуть заздалегідь визначеним списком можливих мов.

- регресії. Для завдань регресії метою є прогнозування безперервного чи дійсного числа. Прогнозування річного доходу людини від її освіти, її віку та місця проживання є прикладом завдання регресії. При прогнозуванні доходу передбачене значення є сумою і може бути будь-яким числом у заданому діапазоні. Ще один приклад регресії Завдання передбачає прогнозування врожайності зернового господарства з урахуванням таких характеристик, як попередні врожаї, погода та кількість працівників, що працюють на фермі. Прибутковість знову може бути довільним числом.

1.5 Постановка завдання

Головна мета випускної роботи полягає у розробці програмного забезпечення прогнозування спортивної статистики для підвищення якості приймаємих рішень. Сформовано такі завдання до роботи:

- дослідження методів прогнозування спортивної статистики;
- вибір засобів розробки;
- розробка технічного завдання;
- проєктування структури програмного забезпечення;

- розробка програмного забезпечення;
- тестування та відлагодження програми;
- економічне обґрунтування доцільності розробки.

1.6 Висновки за розділом 1

Розглянуто проблемну область. В результаті проведеного аналізу запропоновано використовувати штучні НМ у якості моделей прогнозування спортивних показників та характеристик.

Також у першому розділі було проведено дослідження відомих підходів та методів для навчання НМ. Під час дослідження було виявлено, що на сьогоднішній день існує велика кількість засобів для покращення точності роботи НМ. Проте кожен з них вимагає детального дослідження та накладає притаманні йому обмеження стосовно вхідних та вихідних параметрів, що в свою чергу впливає на складність реалізації алгоритмів та їх виконання. Деякі з підходів вже вважаються застарілими та вимагають додаткових перевірок або ж просто не можуть застосовуватися для роботи із сучасними архітектурами НМ.

Для вирішення існуючих проблем було запропоновано нову стратегію підходу до реалізації прогнозування характеристик футболістів. Сформовано завдання до роботи у вигляді дослідження методів прогнозування спортивної статистики.

2 МАТЕРІАЛИ І МЕТОДИ

В першому розділі було початково окреслено практичну задачу прогнозування позиції гравця на футбольному полі відносно його характеристик.

Другий розділ буде присвячено виробленню власного методу прогнозування, який має враховувати особливості роботи методів машинного навчання на даному етапі та визначати особливості їх застосування для задачі, що розглядається.

2.1 Задача прогнозування позиції футболіста на полі

Як було зазначено в розділі 1, світ спорту має незлічену кількість даних, що можливо аналізувати, показників, які можна прогнозувати та використовувати в різних сферах під час прийняття рішень менеджером. Прогнозування позицій гравця є актуальною задачею з достатніми об'ємами даних, для підтвердження та візуалізації гіпотез, зокрема того, чи впливають різні фізіологічні характеристики, де саме буде грати гравець, або знаходження зв'язку між його параметрами та позицією. Будь-який футбольний аналітик стверджуватиме, що для нападника важливий удар, вибір позиції. Відповідно під час розв'язання задачі прогнозування позиції гравця за його навичками та характеристиками проте це і є ключовим завданнями, створити модель, яка за ознаками буде прогнозувати позицію.

Подібна модель може бути використана в різних сферах, наприклад, скаутами, аналітиками для прогнозування складу на матч, в відеоіграх для симуляції складу та в подальшому результаті матчу, тощо.

Оскільки ми оперуємо позиціями, їх можливо представити як класи і завдання можливо сформулювати наступним чином: «передбачити клас гравця на футбольному полі за його параметрами».

2.2 Аналіз даних

Аналіз даних важлива складова при навчанні НМ. Іноді саме вибір та вивчення даних займає більше часу, ніж навчання моделі. Головними вимогами до даних в МН є:

- достатня кількість – різні алгоритми МН, включаючи досить прості, майже однаково здатні вирішувати складні проблеми, коли їм надано достатньо даних. Результати досліджень, говорять про те, що потрібно завжди переглядати компроміс між витрачання часу на розробку алгоритму та дослідженню природи даних [21];
- нерепрезентативні дані – використовуючи навчальний набір, ми призовимо до моделі, яка навряд чи може роботи точні прогнози. Дуже важливо використовувати навчальний набір, який є репрезентативним для випадків, які ви досліджуєте. На вибірка з недостатньою кількістю даних стосовно одних чи інших параметрів, ви можете отримати шум вибірки, водночас, для великих даних в разі перепадів між кількістю одних даних та інших, ви можете отримати зміщення вибірки (упередженість);
- нерелевантні характеристики – пошук релевантних характеристик виокремлений в окремий процес, що називається конструювання ознак. Завдання процесу, виокремити найбільш релевантні характеристик для дослідження, а також комбінування їх для створення нових;
- низька якість даних – в будь-якому наборі даних можливі помилки та пропуски. Це ускладнює навчання системи та пошук для з'єднань та відстеження шаблонів при навчанні. Часто, процес очищення даних є одним із найбільш затратних стосовно часу.

Незважаючи на велику кількість даних в світі спорту, знайти реальні показники футболістів в відкритому доступі дуже важко. Тому було прийняте рішення, взяти дані з футбольної комп'ютерної гри FIFA 2019/2020. Зважаючи на репутацію гри, її вклад у розвиток спортивних симуляторів, дані наявні в наборі відповідають майже всім зазначеним вимогам вище.

Дані представляють собою в табличному форматі, в якому колонки відображають характеристики, а один рядок є відповідним до футболіста.

– навчальна вибірка містить 18278 футболістів та 102 характеристики їх;

– тестова вибірка містить 17770 запис та 102 параметри.

Об'єм даних 8.7 та 8.6 МБ відповідно.

Приклади характеристик футболістів представлені в табл. 2.1.

Таблиця 2.1 – Характеристики футболістів

Номер	Назва параметру	Пояснення
1	power_long_shots	сила дальнього удару
2	skill_dribbling	дриблінг
3	gk_diving	стрибки (для голкіпера)
4	power_stamina	фізична витривалість
5	mentality_aggression	агресивність
6	attacking_short_passing	короткий пас
7	skill_long_passing	довгий пас
8	movement_speed	швидкість бігу
9	power_jumping	сила стрибка

Незважаючи на достатню якість даних, враховуючи кількість характеристик, проблемами даної вибірки можна назвати наступне:

- наявність надлишкових показників;
- наявність відсутніх значень;
- наявність категоріальних значень.

2.3 Використання статистичних методів

Для порівняння з запропонованим методом, використаємо статистичні методи:

- логістичну регресію;
- метод k-найближчих сусідів;
- метод опорних векторів.

2.3.1 Логістична регресія

Логістична регресія моделює ймовірності проблем класифікації з двома можливими результатами. Однак існують різні стратегії, за допомогою яких можна виконати багатокласову класифікацію з кількома бінарними класифікаторами. Модель логістичної регресії використовує логістичну функцію для стиснення результату лінійного рівняння від 0 до 1. Модель логістичної регресії для класифікації вираховує ймовірність за наступною формулою (2.1):

$$\hat{p} = \sigma(x^T \theta), \quad (2.1)$$

де σ – сигмоїдна функція;

x^T – сума ваг;

θ – вектор вхідних параметрів включаючи значення зміщення.

Ідея навчання моделі визначається розрахунком вектору вхідних параметрів, таким чином, щоб модель оцінювала високі ймовірності для позитивних випадків та низькі для негативних. Функція втрат для логістичної регресії описується наступним чином, формула (2.2) [22]:

$$c(\theta) = \begin{cases} -\log(\hat{p}), y = 1 \\ -\log(1 - \hat{p}), y = 0 \end{cases} \quad (2.2)$$

Функція втрат, формула (2.3), для всієї навчальної вибірки – це середня вартість за всі навчальні екземпляри.

$$J(\theta) = -\frac{1}{m} \sum [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})], \quad (2.3)$$

де $y^{(i)}$ – передбачуване значення;

$\hat{p}^{(i)}$ – імовірність для i -того значення (гіпотетична функція).

Для мінімізації функції втрат використовується градієнтний спуск. Теоретичний опис якого наведено в розділі 1.

2.3.2 Метод К-найближчих сусідів

Алгоритм К-найближчих сусідів передбачає, що подібні речі існують у безпосередній близькості. Алгоритм використовується як для бінарної так і для багатокласової класифікації. Алгоритм на відміну від інших не потребує параметрів, тому навчання не вимагає проходження класичного ітераційного проходу з метою оптимізації набору параметрів. Алгоритм можливо описати наступними кроками:

Крок 1. Завантаження даних.

Крок 2. Ініціалізація К до вибраної кількості сусідів.

Крок 3. Вирахувати дистанцію для кожного екземпляра вибірки.

Крок 4. Вибрати К екземплярів, відстань яких мінімальна.

Крок 5. Клас шуканого екземпляру – це клас, який найчастіше зустрічається серед К найближчих сусідів.

Дистанцію між двома точками в n -мірному просторі розраховують з використанням Евклідової метрики, формула (2.4):

$$\begin{aligned} d &= \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} = \\ &= \sqrt{\sum_{i=1}^n (p_i - q_i)^2}, \end{aligned} \quad (2.4)$$

де p – перша точка $(p_1, p_2, \dots, p_n)$,

q – друга точка $(q_1, q_2, \dots, q_n)$.

Метод не має функції втрат, яку можна мінімізувати під час тренування. Цей алгоритм є засобом запам'ятовування даних, тому поняття «навчання» до нього не є доречним. Єдиним "тренуванням", яке відбувається для нього, є запам'ятовування даних (створення локальної копії), щоб під час прогнозування ви могли здійснити пошук і проголосувати більшістю голосів. Технічно жодна функція не підключена до даних, а отже, оптимізація не проводиться (її неможливо навчити, використовуючи градієнтний спуск).

2.3.3 Метод опорних векторів

Метод опорних векторів – це лінійна модель для задач класифікації та регресії. Він здатний вирішувати лінійні та нелінійні задачі та підходить для багатьох практичних задач. Алгоритм будує лінію або площину, яка розділяє дані на класи, приклад зображено на рис. 2.1.

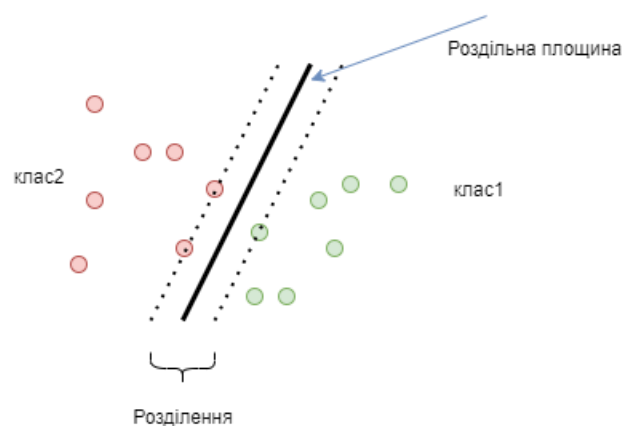


Рисунок 2.1 – Метод опорних векторів

Для розділення двох класів точок даних існує безліч можливих площин, які можна вибрати. Мета знайти площину, яка має максимальний запас, тобто максимальну відстань між точками даних обох класів. Максимізація відстані розділення забезпечує певне підсилення, щоб майбутні точки даних можна було класифікувати з більшою впевненістю. Опорні вектори – це точки, які знаходяться ближче до роздільної площини і впливають на положення та

орієнтацію гіперплощини. За допомогою них максимізується розділення класифікації.

Модель вираховує імовірність класу нового екземпляра, обчислюючи функцію прийняття рішення $w^T x + b = w_1 x_1 + \dots + w_n x_n + b$. Якщо результат позитивний, прогнозований клас дорівнюватиме 1, в іншому випадку 0, представлено у вигляді формули (2.5):

$$\hat{y} = \begin{cases} 0, & w^T x + b < 0, \\ 1, & w^T x + b \geq 0 \end{cases} \quad (2.5)$$

де $\hat{y}$ – прогнозоване значення.

Функція втрат для всієї навчальної вибірки виглядає наступним чином, формула (2.6):

$$J(w, b) = -\frac{1}{2} w^T w + C \sum_{i=1}^m \max(0, 1 - t^{(i)}(w^T x^{(i)} + b)), \quad (2.6)$$

де w – масив ваг;

x – навчальний екземпляр;

C – функція втрат;

$\max(0, 1 - t)$ – звісна функція (функція для обчислення втрат) дорівнює 0 при $t \geq 1$. Похідна дорівнює -1, при $t < 1$ та 0, при 1 [23]-[24].

2.4 Запропонований метод прогнозування позиції футболіста на полі

Будь-яка дослідницька задача з використанням нейронних мереж включає в собі наступні етапи:

- етап 1. підготовка даних;
- етап 2. відбір гіперпараметрів;
- етап 3. відображення результатів.

В наступних пунктах розглянемо детальніше кожен з описаних кроків вище.

2.4.1 Етап 1. Підготовка даних

Незважаючи на відносно цілісний формат даних, перед тим, як навчати модель, необхідно провести попередню обробку даних. Для навчання та тестування моделі було обрано дві вибірки даних за різні роки. Алгоритм підготовки даних наведено нижче.

Крок 1. Очистка даних. Першим кроком є видалення непотрібних характеристик для прогнозування позицій, наприклад, імені гравця, його вартості, зарплатня, національності, тощо. Всі характеристики, які не впливають на позицію гравця, були обрані на основі експертної оцінки.

Наступним кроком є робота з даними, які є відсутніми. Для цього існує три наступних варіанти:

- видалення цих даних за рядками;
- видалення атрибуту (за колонками);
- встановлення деякого значення (нуль, середнє, медіану).

Для третього варіанту бажано обрати медіану з тренувальної вибірки та заповнити пропущені дані. Важливо подібним чином розраховувати медіану для тестової вибірки, або зберегти медіану тренувальної і застосувати для тестової.

При роботі з наявним набором даних було знайдено близько 40 унікальних стовпці, які налічували відсутні дані.

Останнім кроком очищення даних, була робота з категоріальними значеннями. Більшість алгоритмів НМ передбачають роботу з числовими даними, тому потрібно конвертувати категоріальне значення, яке переважно текстове до цифрового. В даному наборі були виокремлені такі поля, як «лівша-правша», «працьовитість», «структура тіла», тощо.

Після того, як усі дані були оброблені та приведені до єдиних числових типів, можна перейти до етапу нормалізації даних.

Крок 2. Нормалізація даних – це процес приведення даних до однакового діапазону або масштабу, щоб жодна змінна не переважала іншу. Цей процес дуже важливий, оскільки дуже часто вибірки даних мають дуже різний діапазон між параметрами [25]. Оскільки, більшість алгоритмів використовують відстань між двома точками, розрахунки на різних діапазонах можуть стати проблемою. Суть нормалізації полягає у зміні діапазонів вхідних значень, щоб їх можливо було описати як нормальний розподіл.

Для виконання роботи, було обрано один з найпоширеніших алгоритмів Min-Max. Алгоритм дуже простий, його суть полягає у зміні вхідного діапазону до діапазону між 0 та 1, представлено на формулі (2.7):

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}, \quad (2.7)$$

де x – вхідне значення;

x' – нормалізоване.

Алгоритм підготовки даних схематично зображений на рис. 2.2.

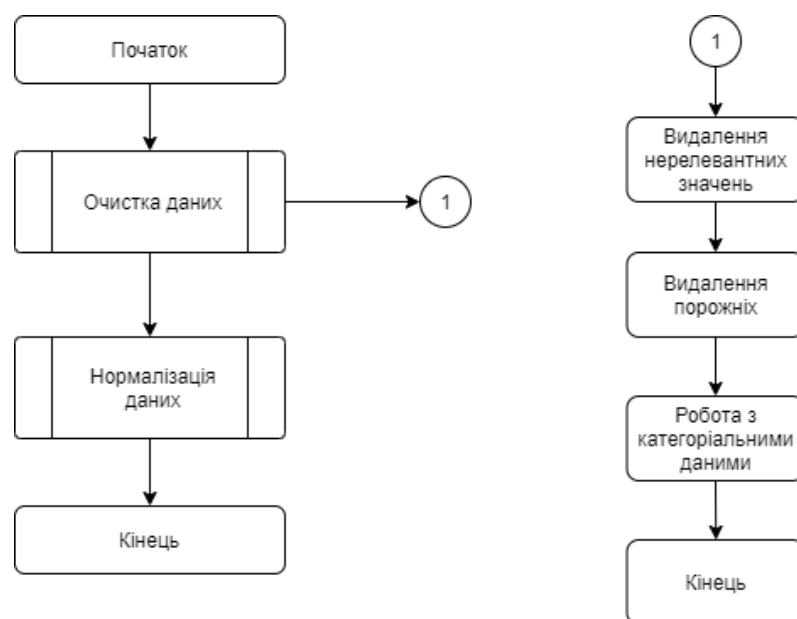


Рисунок 2.2 – Схема обробки даних

В запропонованому методі функція підготовки поєднує одночасно і очистку даних і нормалізацію. На вхід до функції подаються 4 параметри: навчальна вибірка характеристик, навчальна вибірка значень, тренувальна вибірка характеристик, тренувальна вибірка значень. Вибірки характеристик – це показники гравця: швидкість, точність удару, тощо. Вибірка значень описує індекс гравця та його реальну позицію на полі.

Вихідними значеннями функції є нормалізовані набори даних, для яких було проведено вище зазначені операції.

2.4.2 Етап 2. Відбір гіперпараметрів

Побудова архітектури нейронної мережі включає в собі вибір необхідних гіперпараметрів та налаштування їх з метою покращення результатів. До гіперпараметрів нейронної мережі відносять:

- кількість прихованих шарів;
- кількість нейронів на прихованих шарах;
- вибір функцій активації;
- вибір параметрів оптимізації;
- вибір результуючих показників.

Зазвичай вибір усіх параметрів залежить від типу задачі та досвіду науковця. Тип задач формує основні уявлення про НМ, що впливає на деякі показники та представлення наявних архітектур для подібних задач. Прикладом залежності між параметрами та типом задач можна навести різницю вихідного слою у кількості нейронів. Як було вище зазначено, задача регресії передбачає отримання певної числової відповіді, тому кількість нейронів відповідно 1. В задачах класифікації кількість нейронів вихідного слою залежить від кількості прогнозованих класів.

Рекомендована архітектура НМ для задач класифікації представлена у табл. 2.2 наступним чином [26].

Таблиця 2.2 – Базові гіперпараметри НМ для задачі класифікації

Гіперпараметри	Значення
Вхідна кількість нейронів	1 для кожної характеристики
Кількість прихованих шарів	В залежності від задачі, 1 до 5
Нейрони для прихованих шарів	В залежності від задачі, 10 до 100
Вихідна кількість нейронів	Для кожного класу
Вихідна функція активації	Softmax/Sigmoid
Функція втрат	Перехресна ентропія

Перехресна ентропія між двома розподілами ймовірності p та q над спільним простором подій вимірює середню кількість біт, необхідних для впізнання події з простору подій, якщо схема кодування, що використовується, базується на розподілі ймовірностей q , замість «істинного» розподілу p [27].

Для МН – це показник ступеня відмінності між двома розподілами ймовірностей, відповідно до навчання з вчителем, один із розподілів ймовірностей показує позначку "істинно" для навчальних зразків, а правильні відповіді вказуються зі значенням сто відсотків. Математично виражається наступною формулою (2.8) [28]:

$$I(p, q) = - \sum p(x) \log q(x), \quad (2.8)$$

де x – прогнозовані значення;

$p(x)$ – імовірність розподілу «позитивних» категорій;

$q(x)$ – відображає оцінку алгоритму.

Перехресна ентропія досліджує прогнози моделей із справжнім розподілом ймовірностей. Він знижується, коли прогнози стають більш точними, і стає нульовим, коли прогнози мають тенденцію до вдосконалення.

Проте, в більшості випадків базова архітектура потребує змін в залежності від задачі, а також емпіричного підходу до підбору параметрів.

Тому, було вирішено згенерувати максимально можливу кількість НМ, з урахуванням вище зазначених особливостей. Для пошуку найкращих гіперпараметрів обрано Random Search, зважаючи на великий обсяг показників.

Random Search – група методів числової оптимізації, які не вимагають обчислення градієнту для розв'язання задач оптимізації; отже, випадковий пошук можна використовувати для функцій, що не є неперервними або диференційованими. Подібні методи оптимізації також називаються методами прямого пошуку, методами «чорної скриньки» або методами без використання похідної [29].

Даний метод підходить для пошуку неявних з'єднань між характеристиками, які важко виокремити зважаючи на кількість, а іноді неоднозначність параметрів [30].

Метод було застосовано з поділом вибірки на 5 частин та проведені 10 ітерацій навчання, до наступних заздалегідь сформованих вхідних гіперпараметрів, представлених в табл. 2.3.

Таблиця 2.3 – Початкові значення гіперпараметрів

Гіперпараметри	Значення
Вхідна кількість нейронів	[512, 256]
Кількість прихованих шарів	1-3
Нейрони для шар1	[256, 128, 64]
Нейрони для шар2	[128, 64]
Функція активації вхідний шар	relu
Функція активації шар1	[relu, softmax, tanh, sigmoid, hard_sigmoid]
Функція активації шар2	[relu, softmax, tanh, sigmoid, hard_sigmoid]
Вихідна функція	[sigmoid, softmax]
Функція оптимізації	[adam, adamax]

За результатами виконаного методу було отримано наступні гіперпараметри, які представлені в табл. 2.4.

Таблиця 2.4 – Розроблена архітектура НМ

Гіперпараметри	Значення
Вхідна кількість нейронів	512
Кількість прихованих шарів	2
Нейрони для шар1	128
Нейрони для шар2	128
Функція активації вхідний шар	relu
Функція активації шар1	tanh
Функція активації шар2	relu
Вихідна функція	sigmoid
Функція оптимізації	adam

Після отримання найоптимальніших гіперпараметрів, було проведено повторне тренування та тестування архітектури НМ на повномірних вибірках даних, для збільшення об'єму та уникнення упередженості моделі. Кількість ітерацій була збережена.

2.4.3 Етап 3. Відображення результатів

Результатом прогнозування є значення класу, що відображає позицію гравця на полі.

Для наглядного відображення роботи методу було зображено графіки порівняння змін значення точності та втрат для тренувальної та тестової вибірок. Окрім цього, було проведено прогнозування та порівняння для футболістів з їх реальними позиціями на полі.

Для наочності результатів прогнозування було створено порівняльну таблицю запропонованого методу та вже існуючих статистичних методів.

2.5 Висновки за розділом 2

У другому розділі проаналізовано алгоритм вирішення завдань машинного навчання. Було досліджено найпоширеніші алгоритми, методи та принцип роботи з даними для підготовки використання у НМ для покращення точності прогнозування. Було запропоновано порівняти сучасні методи на основі статистичних функцій для прогнозування позицій на полі. В процесі дослідження було практично засвідчено деякі недоліки статистичних моделей при виконанні поставленої задачі.

Дослідження базувалися на емпіричному пошуку найкращих гіперпараметрів з використанням методу Random Search.

Наукова новизна роботи полягає в тому, що в ній запропоновано метод прогнозування позиції футболіста на полі за його навичками та характеристиками, що характеризується використанням багатошарової нейронної мережі прямого поширення сигналу для розв'язання задачі класифікації, що дозволяє приймати рішення щодо визначення відповідальності футболіста на полі.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір засобів реалізації програмного забезпечення

В даному параграфі описано основні програмні засоби, які використовувались для побудови програмної системи.

3.1.1 Вибір мови програмування

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною типізацією [31].

Основними аспектами та перевагами Python розробники зазвичай виділяють: простоту, вільність та портативність, підтримка об'єктно-орієнтованого програмування, великі стандартні бібліотеки та можливість розширення програмного коду, інтеграційні механізми.

Завдяки вільному поширенню мови, з плином розвитку науки аналізу даних та машинного навчання з'явилися численні математичні бібліотеки. На сьогоднішній день мова є однією з найпопулярніших при використанні нейронних мереж та обробки даних, оскільки налічую в собі такі потужні бібліотеки як: NumPy, Pandas, Scikit-Learn, TensorFlow, Matplotlib.

В табл. 3.1 наведено порівняльну характеристику зазначених мов програмування.

Таблиця 3.1 - Порівняльна таблиця мов програмування

Критерії порівняння	Мови програмування			
	C++	Java	C#	Python
1	2	3	4	5
Складність мови	++	±	±	-
Математичний апарат	++	±	±	++

Продовження таблиці 3.1

1	2	3	4	5
Простота розроблення графічного інтерфейсу	±	±	++	±
Продуктивність роботи	+ +	+	++	±

Також Python використовується для побудови вебзастосунків на базі фреймворків Django так Flask.

3.1.2 Вибір серверного фреймворку

У якості фреймворку було обрано Django.

Django – високоякісний вебфреймворк, який дозволяє швидко стартувати проєкти завдяки наявності готових рішень. Під готовими рішеннями розуміють наявність усіх необхідних вебкомпонентів для початку проєкту, наприклад адмін-панелі, налаштування авторизації та реєстрації користувачів [32].

Одними з головних принципів побудови проєкту на Django – є використання різних модулів. Розбиття програмного продукту на різні модулі впливає на швидкість написання коду та тестування. Незалежні один від одного компоненти, дозволяють проводити тестування локально в обраному модулі, не зачіпаючи іншу систему.

Розбиття на модулі, також дозволяє розділити задачі для різних розробників, наприклад:

- написання модулю роботи з базою даних (Django підтримує використання ORM та транзакцій);
- написання модулів бізнес логіки з використанням різних математичних бібліотек;
- модифікація інтерфейсу. Django містить в собі шаблон для написання статичних сторінок.

Завдяки використанню мови Python в розробці фреймворка, він надає можливості використовувати диспетчер обробки запитів на основі регулярних виразів, підключення усіх бібліотек на мові Python.

Також зручним додатком до технічних можливостей є вбудована автоматична документація по тегах шаблонів та сутностям даних.

Порівняльна характеристика найпоширеніших фреймворків мови Python наведена у табл. 3.2.

Таблиця 3.2 – Порівняння вебфреймворків

Показник	Flask	Django
Швидкість роботи	++	++
Простота використання	++	+
Простота вивчення	+	+/-
Підтримка версій мови	2.7+	2.7+
Підтримка БД	(MySQL, Postgress SQLite)	(MySQL, Postgress, SQLite)
Підтримка розширень	++	++
Модульність	+	++
Спектр використання	+	++

Враховуючи особливості проєкту, а саме математичний модуль для НМ та необхідність надійності в реалізації взаємодії між різними частинами програми, було обрано Django. Також, особливістю фреймворка є легке створення API-інтерфейсів REST. В Django включені майже всі необхідні компоненти для швидкого старту REST проєкту.

3.1.3 Вибір фреймворку для реалізації інтерфейсу

Для відображення результатів та взаємодії користувача з програмою, було вирішено використати одно-сторінковий фреймворк (SPA) – Angular2+.

Angular2+ – це фреймворк, розроблений компанією Google, на заміну старій версії, яка була написана на чистому JavaScript [33].

До головних ознак переписаного фреймворку відносяться наступні:

- покращення кросплатформеної розробки, тобто для екранів з різними розмірами та роздільною здатністю;
- покращення показників швидкості роботи, а саме компілювання та завантаження сторінки сайту у веббраузері;
- відмінні засоби розробки для використання фреймворку. Незлічена кількість плагінів для Visual Studio Code та WebStorm;
- покращення надійності кодування завдяки використанню типізованої мови TypeScript.

Оновлена версія фреймворку дійсно склала конкуренцію таким аналогам як React та Vue. Однією з найбільших переваг Angular над зазначеними аналогами, є те, що всі необхідні модулі та компоненти вже складають базу фреймворка. Зі створенням проєкту розробник отримує такі готові рішення, як:

- написання коду в ООП-стилі;
- структурування проєкту за модулями;
- використання сервісів та DI;
- вбудований навігаційний модуль;
- використання модулю HTTP-запитів.

Порівняльна характеристика SPA-фреймворків мови Python наведена у табл. 3.3.

Таблиця 3.3 – Порівняння SPA-фреймворків

	Angular2	React	Vue
Швидкість роботи	+	++	++
Простота використання	+	+	+
Модульність	++	+	+

Продовження таблиці 3.3

	Angular2	React	Vue
Наявність готових рішень	++	-	-
Мова програмування	TypeScript	JavaScript TypeScript	JavaScript
Спектр використання	Великі проєкт	Середні/Невеликі проєкт	Середні/Невеликі проєкт
Простота вивчення	-	+-	+-
Надійність	++	+	+

Зважаючи на те, що Angular2+ використовує готові рішення в своїй основі, тому старт проєкту та створення архітектури займає менше часу, зважаючи на відносно простий інтерфейс, було прийнято зосередити увагу на швидкості реалізації функціоналу.

3.1.4 Вибір бази даних

Оскільки програмний продукт пов'язаний з сферою машинного навчання, для якої прийнято використовувати базу даних SQLite.

SQLite – полегшена реляційна система керування базами даних. Дана реляційна база поширена у вигляді відкритого коду та підтримується спільнотою користувачів, серед яких є і відомі компанії як: Adobe, Oracle, Mozilla [34].

Особливістю використання SQLite є її портативність та малий розмір. База представлена у вигляді файлу, яких з легкістю можна переміщувати та зберігати в середині програмного застосунку, як локальне сховище. Зважаючи на це, SQL є деяким стандартом при роботі в сферах аналізу даних та машинному навчанні. На відміну від текстових або табличних файлів типу (.txt, csv), SQLite виступає, як централізоване сховище різних даних.

Окрім створення єдиного джерела даних, SQLite забезпечує повноцінне виконання SQL скриптів, що полегшує отримання та обробку даних.

Порівняльна характеристика найпоширеніших баз даних наведена у табл. 3.4.

Таблиця 3.4 – Порівняння баз даних

	SQLite	MySQL	Postgress
Необхідність пам'яті	--	++	-
Безпека	-	++	++
Ціна	-	+	-
Розширення	-	+	++
Швидкість	+	+	+
Сфери використання	вбудовані проекти, машинне навчання	системи безпеки, вебсайти	вебсайти

Зважаючи на те, що наші дані представлені у вигляді набору таблиць без реляційних сполучень, а також на більшість запитів до БД будуть на читання, було вирішено обрати БД відносно досліджуваної області.

3.2 Вибір середовища розробки

Програмний комплекс розроблявся за допомогою PyCharm для створення математичного модулю та серверної частини та Visual Studio Code для інтерфейсу.

3.2.1 Опис PyCharm

Для реалізації математичних методів та використання фреймворку Django, було використано інтегроване середовище розробки PyCharm для мови програмування Python.

PyCharm – це ще одна розробка від компанії JetBrains, яка використовує єдиний інтерфейс для всіх своїх програмних засобів, проте кожен з яких підтримує певну мову та напрям. Зважаючи на це, будь-якому розробнику, який користувався іншими засобами цієї компанії, буде простіше використовувати інші її продукти [35]. Окрім, того PyCharm являється повноцінним інтегрованим середовищем, на відміну від Visual Studio Code. В свою чергу це надає більше можливостей і переваг, наприклад:

- всі типові проєкти для мови Python доступні в базовій комплектації, відсутня необхідність плагінів.
- доступ до контролю версій, як базова можливість;
- кращий редактор та синтаксичний аналізатор;
- повноцінний зневаджувач, що сприяє швидкості розробки;
- підтримка важливого компоненту для машинного навчання Jupyter;
- до типових проєктів відносяться вебфреймворки: Django та Flask;
- синтаксична підтримка для вебінтерфейсів та статичних сторінок з використанням HTML/CSS/SASS/LESS;
- велика інтеграційна база для налаштування процесів розробки, інтеграція з Jira/GitLab/GitHub;
- можливості підключення до БД та виведення діаграм.

3.2.2 Опис Visual Studio Code

Visual Studio Code – потужний редактор від компанії Microsoft, зазнав популярності для використання у веброботі [36].

Редактор підтримує численну кількість плагінів, що спрощують ведення розробки вебінтерфейсів.

Однією із характерних особливостей редактору є кросплатформеність та відносно малий розмір необхідних ресурсів, на відміну від повноцінних інтегрованих систем.

Як зазначалось вище, VsCode має велику кількість плагінів і для Angular2+ також. Редактор повноцінно підтримує мову TypeScript та засоби зневадження проєктів типу Angular. Доступне підключення аналізатору коду TSLint, що впливає на якість написання та передбачення синтаксичних помилок. Також плагін Angular Scheme 9+ спрощує створення архітектурних компоненті для Angular2+ таких як: компоненти інтерфейсу, сервіси, модулі, пайпи, директиви і т.п.

До особливостей редактора, також відносять вбудовану інтеграцію з GitHub та Azure.

3.3 Опис основних вимог до програми

Головне завдання до програмної системи, що розробляється – це реалізувати метод прогнозування позицій гравці та представити приклад його роботи для звичайного користувача.

Серед функціональних вимог до системи можна виділити:

- аналіз та обробка даних: проведення очищення, нормалізації даних, формування тренувальної та тестової вибірок;
- пошук гіперпараметрів моделі;
- створення моделі, тренування та тестування;
- візуальне порівняння справжніх значень та прогнозованих;
- передбачення позиції гравця;
- реалізація відомих алгоритмів класифікації;
- виведення проміжних та фінальних результатів роботи програми;
- виведення інформації про запропоновану архітектуру НМ.

Після оновлення інформаційної бази або функціоналу, додаток треба перезапустити.

3.4 Проєктування бази даних

В даному проєкті БД використовується як джерело даних для НМ моделей та запису їх результатів. З цього випливає, що зв'язків між таблицями немає. Використання БД в протиставу текстовим форматам даних зумовлено створенням централізованого джерела даних та зручним використанням мови запитів SQL. Таблиці бази даних представлені на рис. 3.1.

season19	
index	integer
sofifa_id	integer
player_url	text
short_name	text
long_name	text
age	integer
dob	text
height_cm	integer
weight_kg	integer
nationality	text
club	text
overall	integer
potential	integer
value_eur	integer
wage_eur	integer
player_positions	text
preferred_foot	text
international_reputation	integer
weak_foot	integer
skill_moves	integer

results_mlp	
index	integer
train	real
test	real
train_loss	real
test_loss	real
input	integer
h1	integer
h2	integer
output	text
act1	text
act2	text
optimizer	text

results	
index	integer
models	text
train	real
test	real

season20	
index	integer
sofifa_id	integer
player_url	text
short_name	text
long_name	text
age	integer
dob	text
height_cm	integer
weight_kg	integer
nationality	text
club	text
overall	integer
potential	integer
value_eur	integer
wage_eur	integer
player_positions	text
preferred_foot	text
international_reputation	integer
weak_foot	integer
skill_moves	integer

Рисунок 3.1 – База даних системи

База даних представлена у вигляді трьох таблиць (огляд деяких даних було зроблено у другому розділі):

- season19 та season20 – дані представлені за 2019 рік та 2020;
- results – таблиця до якої записуються результати створених статистичних моделей. Представлені поля: index – номер запису, models – назва моделі, train – показник точності для тренувальної вибірки, test – показник точності для тестової вибірки;
- results_mlp – таблиця створеної НМ на основі БШП. Представлені поля: index – номер запису, train – показник точності для тренувальної вибірки, test – показник точності для тестової вибірки, train_loss – показник втрат тренувальної вибірки, test_loss – показник втрати тестової вибірки, input – кількість нейронів на входному шарі, h1 – кількість нейронів на першому прихованому шарі, h2 – кількість нейронів на другому прихованому шарі,

output – кількість нейронів на вихідному шарі, act1 – функція активація для першого шару, act2 – функція активації для другого шару, optimizer – функція оптимізації.

3.5 Структура програмного забезпечення

При проектуванні структури майбутньої програми було виділено 4 основних програмних модуля, які представлено на рис. 3.2.

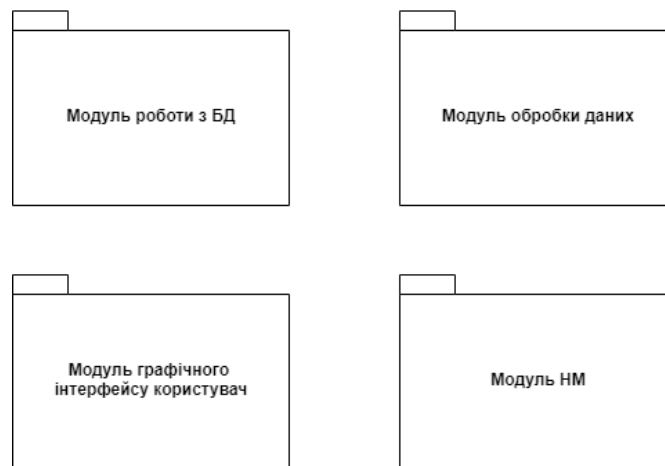


Рисунок 3.2 – Основні програмні модулі

Модульна архітектура програмного комплексу дозволяє розробляти різні компоненти незалежно один від одного, що сприяє гнучкості в розробці та тестуванні. До модулю роботи з БД відносяться такі функції:

- функції зчитування даних із БД;
- функції пошуку даних в БД;
- функції запису даних до БД.

До модулю обробки даних відносяться:

- завантаження навчальної та тестової вибірки;
- обробка даних.

До модулю графічного інтерфейсу відноситься:

- функції виведення графічного інтерфейсу користувача;

- функції зчитування та обробки вводу та подій користувача;
- виведення результатів роботи методу.

До модулю НМ відносяться:

- функції збереження НМ;
- функції завантаження НМ;
- функції підрахунку результатів;
- функції виконання прогнозування.

Варто зазначити, що всередині модулів дозволяється додаткове ділення на класи та моделі.

Діаграма прецедентів надає перелік функцій, які доступні користувачу та функцій, які надає система (рис. 3.3).

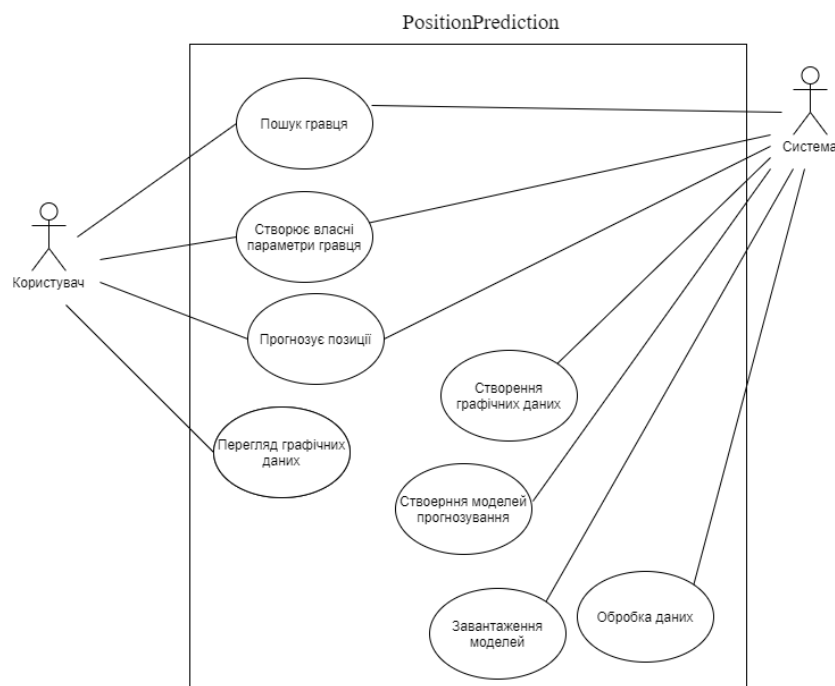


Рисунок 3.3 – Діаграма прецедентів

3.6 Вимоги до технічних засобів

Для нормальної розробки програми, повинні виконуватися наступні умови до апаратної частини:

- наявність IBM-сумісної персональної електронної обчислювальної машини (ПЕОМ), на якій буде працювати додаток;
- процесор із тактовою частотою 1,8 GHz та більше;
- оперативна пам'ять 8 ГБ та більше;
- вільний простір на жорсткому диску не менше, ніж 20 ГБ.

3.7 Висновки за розділом 3

Під час роботи над розділом було спроектовано архітектуру майбутнього ПП. Було розроблено модулі вебплатформи для прогнозування позиції гравця. Також проаналізовано найбільш популярні мови програмування та фреймворки, в якості мови програмування НМ та серверної частини обрано Python та в якості фреймворку Django, для побудови клієнтської частини було обрано Angular2. Після ретельного аналізу та порівняння переваг середовища розробки було обрано середовище JetBrains PyCharm та Visual Studio Code.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

Відповідно до зазначених вимог, система була розділена на два проєкти: інтерфейсну частину та серверну. В свою чергу серверна частина включає в собі такі важливі модулі як обробка даних та робота з НМ.

4.1 Архітектура інтерфейсу

Як було зазначено в розділі 3, для реалізації клієнта було обрано мову програмування TypeScript та фреймворк Angular2. Типові проєкти з використання Angular2 складаються з трьох важливих базисах:

- використання компонентів;
- застосування принципів DI;
- графічний інтерфейс реалізуються за допомогою HTML/SCSS;
- використання модулів програми.

4.1.1 Графічні компоненти

Інтерфейс користувача створений з використанням 4 компонентів зображених на рис. 4.1.

```
import {HttpClientModule} from '@angular/common/http';
import {AppRoutingModule} from './app-routing.module';
import { DashboardComponent } from './dashboard/dashboard.component';
import { HomeComponent } from './home/home.component';
import {NavbarComponent} from './navbar/navbar.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import {MatAutocompleteModule} from '@angular/material/autocomplete';
import {ReactiveFormsModule} from '@angular/forms';
import {MatProgressSpinnerModule} from '@angular/material/progress-spinner';
import {MatTableModule} from '@angular/material/table';

@NgModule({
  declarations: [
    AppComponent,
    DashboardComponent,
    HomeComponent,
    NavbarComponent,
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    AppRoutingModule,
    DashboardComponent,
    HomeComponent,
    NavbarComponent,
    BrowserAnimationsModule,
    MatAutocompleteModule,
    ReactiveFormsModule,
    MatProgressSpinnerModule,
    MatTableModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Рисунок 4.1 – Підключення компонентів в Angular2

Кожний компонент має своє призначення:

- AppComponent – стартове відображення, використовується для побудови навігації;
- DashboardComponent – компонент для відображення результатів методів (точність, помилка та т.п);
- HomeComponent – компонент для передбачення позиції гравця;
- NavbarComponent – компонент для хедеру сайту.

Типовий графічний компонент складається з трьох або чотирьох (якщо підключений файл для тестування): файл інтерфейсу типу html, файл стилів (CSS/SASS/LESS) та файл логіки мовою TypeScript. Приклад компоненту на рис. 4.2.

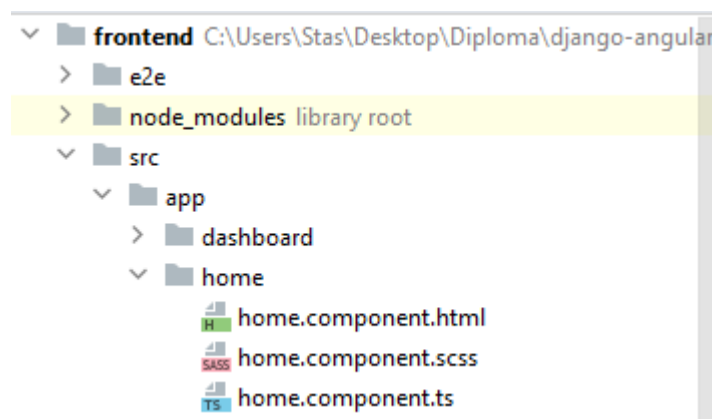


Рисунок 4.2 – Структура компоненту Angular2

Для реалізації логіки програми було виділено сутності та сервіси.

4.1.2 Сутності інтерфейсу

Відносно призначення сутності поділилися на дві групи, для обробки результатів НМ та роботи з гравцем. До сутностей НМ належать:

- IPredictionPosition – обробка результату прогнозування різними НМ. Містить поля: method – назва архітектури НМ, result – прогнозована позиція;

- `IAccuracyMethod` – обробка результату прогнозування усіх алгоритмів НМ та типів вибірок: Містить поля: `model` — назва архітектури НМ, `train` – результати для тренувальної вибірки, `test` – результати для тестової вибірки;

- `IParamsMLP` – обробка результату запиту про інформацію створеної НМ. Містить поля: `index` – індекс, `train` – показник точності для тренувальної вибірки, `test` – показник точності для тестової вибірки, `train_loss` – показник втрат тренувальної вибірки, `test_loss` – показник втрати тестової вибірки, `input` – кількість нейронів на вхідному шарі, `h1` – кількість нейронів на першому прихованому шарі, `h2` – кількість нейронів на другому прихованому шарі, `output` – кількість нейронів на вихідному шарі, `act1` – функція активація для першого шару, `act2` – функція активації для другого шару, `optimizer` – функція оптимізації;

- `IParamsGraph` – обробка запиту на виведення графіків точності та втрат створеної MLP. Містить поля: `loss` – втрати, `acc` – точність.

Сутності для роботи з гравцем:

- `IPlayerParams` – обробка результату запиту за індексом гравця та отримання його характеристик. Містить поля: `id` – індекс, `age` – вік, `short_name` – скорочене ім'я, `club` – назва клубу, `url` – посилання на діаграму навичок (зображення), `position` – реальна позиція, `attack` – навички атаки, `technique` – техніка, `speed` – швидкість, `physics` – фізична сила, `defence` – навички в захисті, `mental` – психологічні навички, `goalkeeping` – навички воротаря;

- `ISearchPlayer` – обробка результату пошуку гравця. Містить поля: `id` — індекс, `short_name` – скорочене ім'я;

- `SearchPlayer` – допоміжний клас для виведення обробленого значення у пошуку гравця. Містить такі поля: `id` — індекс, `short_name` – скорочене ім'я; Методи: `getIdFromViewFiled` – отримання індексу з пошуку, `searchFiled()` – форматування строки для пошуку, `ViewFiled()` – вивід обробленої строки до пошуку.

4.1.3 Взаємодія з сервером

Для взаємодії з сервером в Angular2 прийнято виділяти сервіси. В сервісах описується базова логіка роботи застосунку, виконання http-запитів на сервер, обробка значень, тощо. В системі використовуються два сервіси:

- PlayerService – сервіс для пошуку гравця та отримання інформації про нього за індексом. Містить два методи, які виконуються http-запит: `searchPlayer(word:string)` – повертає гравця за введеними даними (текст), `getPlayer(playerId: number)` – повертає гравця з його характеристиками (сутність `IPlayerParams`);
- MethodService – сервіс для отримання показників НМ та прогнозування значень. Містить два методи: `predict(playerId: number)` – прогнозування позицій гравця статистичних методів, `getAccuracy()` – отримання значень статистичних методів, `predictMLP(playerId: number)` – прогнозування позицій гравця за допомогою створеної НМ, `getAccuracyMLP()` – отримання значення метрик НМ, `getParamsMLP()` – отримання опису гіперпараметрів НМ, `getGraphMLP()` – отримання графічних матеріалів за MLP.

Графічно інтерфейсу частину для одного компоненту відобразити наступним чином рис. 4.3.

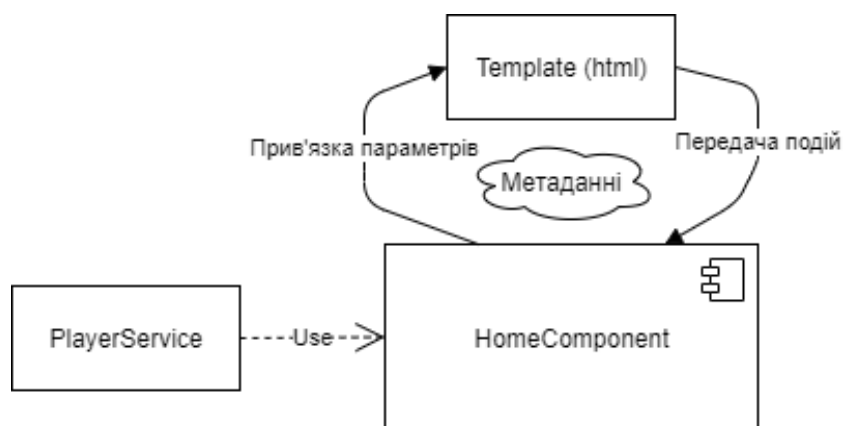


Рисунок 4.3 – Приклад архітектури для одного компоненту

4.2 Архітектура серверної частини

Для серверної частини, як було зазначено вище використовується фреймворк Django. Враховуючи застосування SPA-фреймворка Angular, серверна частина реалізує REST-full арі. Схематично принципи даної архітектури приведено на рис. 4.4.

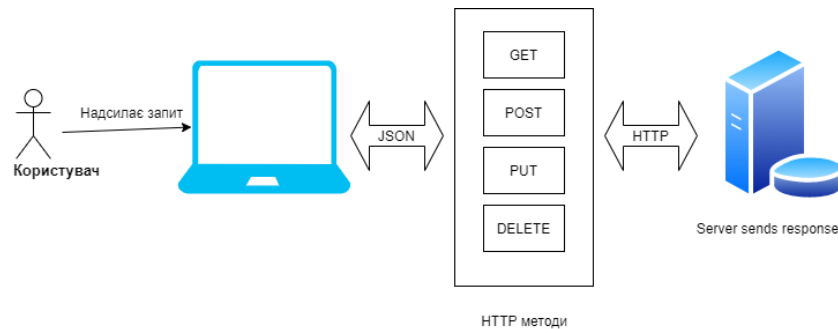


Рисунок 4.4 – Архітектура REST

Серверна частина представляє собою такі структурні компоненти:

- контролер (обробка запитів);
- сервіси (виклик доменних моделей);
- допоміжні файли (підключення до бази даних).

Основні компоненти вебплатформи демонструються через діаграму компонентів на рис. 4.5.

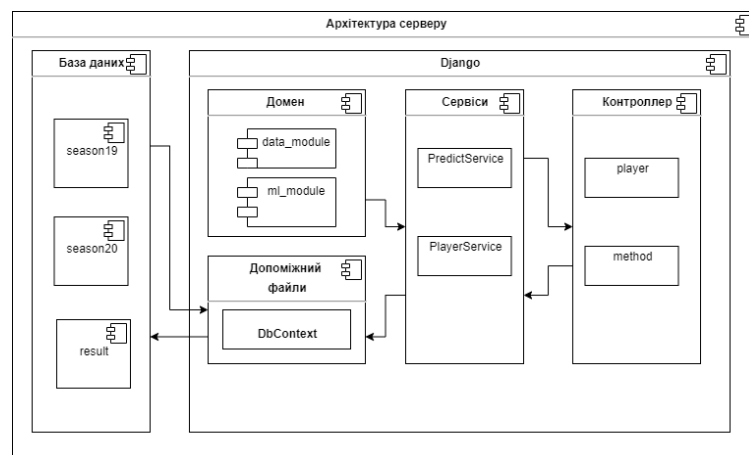


Рисунок 4.5 – Архітектура REST

4.2.1 Контролери серверу

Система містить два контролери для обробки запитів гравця та методи НМ. До основних функцій `player` можна віднести:

- `search(word)` – функція виконує пошук гравця, викликаючи необхідні сервіси;
- `getById(id)` – функція виконує GET-запит для отримання інформації про гравця.

До основних функцій `method` можна віднести:

- `prediction(id)` – функція виконує виклик сервісу пов'язаного з модулем НМ. Результуючим значенням є передбачення позиції гравця;
- `accuracy()` – функція виконує GET-запит для отримання метрик розроблених статистичних методів;
- `accuracy_mlp()` – функція виконує GET-запит для отримання метрик розробленої НМ;
- `accuracy_mlp_params()` – функція виконує GET-запит для отримання гіперпараметрів розробленої НМ;
- `predict_mlp(id)` – функція виконує виклик сервісу пов'язаного з модулем НМ. Результуючим значенням є передбачення позиції гравця;
- `get_mlp_graph()` – функція виконує GET-запит для отримання графічних показників MLP.

4.2.2 Сервіси системи та допоміжні файли

Сервісами системи є два класи: `PlayerService` – створений для виконання різних операцій для сутностей пов'язаних з гравцем (пошук, отримання даних і т.п) та `PredictService` – створений для виконання операцій з створеними нейронними моделями.

До методів `PlayerService` відносяться:

- `search(word)` – пошук гравця;

- `getPlayerById(id)` – отримання даних за індексом для гравця.

`PredictService` містить поле `self.id` – індекс-гравця, для подальшого використання в методах. До методів `PredictService` відносяться:

- `data_processing_by_user()` – обробка вхідних даних гравця за його індексом. Виклик модулю обробки даних;
- `statistic_prediction()` – виклик прогнозування за допомогою відомих статистичних архітектурних НМ. Виклик модулю НМ;
- `mlp_prediction()` – виклик прогнозування за допомогою створеної НМ;
- `get_stat_results()` – отримання метрик статистичних НМ. Виклик модулю НМ;
- `get_mlp_results()` – отримання метрик створеної НМ. Виклик модулю НМ;
- `get_mlp_params()` – отримання гіперпараметрів створеної НМ. Виклик модулю НМ.
- `get_mlp_graph()` – отримання графічного представлення створеної MLP.

До допоміжних файлів відноситься `DbContext`. Цей файл забезпечую комунікацію між сервісами та базою даних. Містить наступні методи:

- `execute_query_like(query, params='')` – метод для виконання запиту до БД з SQL атрибутом `LIKE`. Вхідними параметрами є тіло запиту – `query` та параметр для підстановки в `LIKE` = `params`;
- `execute_query(query, params='')` – метод для виконання простих запитів на вибірку або з конструкцією `where`. Вхідними параметрами є тіло запиту – `query` та параметр для підстановки в `WHERE` = `params`;
- `execute_query_2params(query, t='')` – метод для виконання зіставних запитів з двома параметрами, наприклад `UPDATE` з `WHERE`. Вхідними параметрами є тіло запиту – `query` та параметр для підстановки `t` (типу кортеж);
- `search(word)` – пошук гравця;

- `getPlayerById(id)` – отримання даних за індексом для гравця.

4.3 Модулі обробки даних та нейронних мереж

До домену системи виділяють такі частини, що відображають проблематику предметної області. В даному випадку домен містить модулі обробки даних та побудови НМ.

4.3.1 Модуль обробки даних

Модуль обробки даних містить два класи: `Preprocess` та `AttributeFactory`. `Preprocess` відповідає за обробку вхідних даних та містить функцію нормалізації `preprocessing (x_train, x_test, y_train, y_test)` – параметрами функції є навчальна вибірка, тестова вибірка, навчальні характеристик та тестові характеристики. `AttributeFactory` відповідає за побудову діаграми навичок гравця та містить наступні поля: `id` – гравця та методи:

- `getParams()` – метод завантажує та обробляє необхідні характеристики для побудови діаграми;
- `get_shape_attributes()` – метод формує характеристики за заданими категоріями;
- `create_shape(path)` – метод будує діаграму та зберігає її у відкритій директорії;
- `create_graph_mlp(train, test, path, title, ylabel)` – метод будує графіки втрат та точності для тренувальної та тестової вибірки.

4.3.2 Модуль НМ

Модуль НМ містить в два класи: `StatPrediction` та `MlpPrediction`. `StatPrediction` – клас який будує вже відомі архітектури НМ, виконує навчання,

завантаження. Містить статичний масив `files` – який містить назви файлів та моделей та наступні методи:

- `logistic_regression(x_train, y_train)` – метод створення НМ на основі логістичної регресії;
- `knn_model(x_train, y_train)` – метод створення НМ на основі К-ближніх сусідів;
- `svc_model(x_train, y_train)` – метод створення НМ на основі методу опорних векторів;
- `load_model(file)` – метод завантаження існуючої моделі за шляхом на диску;
- `loading_models()` – метод завантаження усіх моделей;
- `prediction(params)` – метод прогнозування за параметрами гравця – `params`;
- `get_accuracy()` – метод отримання метрик створених НМ.

`MlpPrediction` – клас який будує НМ за результатами дослідження, виконує навчання, завантаження, тренування, тестування. Містить статичний масив `files` – який містить назву файлу і моделі та наступні методи:

- `create_model(neurons_input, level1, level2, a1, a2, opt, a_output)` – метод створення моделі;
- `get_params()` – повертає значення гіперпараметрів моделі;
- `load_model()` – метод завантаження існуючої моделі за шляхом на диску.

4.4 Висновки за розділом 4

Розроблено систему для прогнозування позиції гравця з використанням БШП, а також реалізовано класичні алгоритми прогнозування для створення порівняння. Розроблена система повністю реалізує поставлену задачу. Розроблена система представлена для користувача в якості вебзастосунку.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

Цей розділ присвячено опису розробленої системи для прогнозування позицій гравця. Буде описано сферу застосування, та наведено алгоритм користування. Також у цьому розділі буде проведено тестування та оцінка точності алгоритму.

5.1 Опис застосування програми

Розроблена програмна система призначена для зменшення витрати часу людської роботи та покращення точності передбачення позицій гравців за їх характеристиками.

Для цього система представлена у вигляді вебдодатку з інтерактивною сторінкою пошуку гравця та прогнозування його позиції та сторінки з інформацією про точність застосованих алгоритмів та типу НМ. Вебплатформа представлена для зручного керування прогнозами та пояснень, стосовно типу архітектури НМ. Користувач знаходить гравця в рядку пошуку, отримує статистичні показники, після чого може запускати процес прогнозування створеними методами.

5.2 Інструкція по запуску системи

Для запуску усіх частин системи, як інтерфейсу так і вебсерверу необхідно виконати наступні кроки:

Крок 1. В папці інтерфейсу frontend виконати встановлення пакетів командою “npm install” після чого запустити команду “ng build”, рис. 5.1;

```
Microsoft Windows [Version 10.0.19041.630]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\masterD\frontend>npm install
C:\masterD\frontend>ng build
```

Рисунок 5.1 – Компіляція інтерфейсу програми

Крок 2. Для запуску серверної частини також потрібно виконати встановлення пакетів мови Python, командою `pip install -r requirements.txt`, після чого виконати наступні команди: `python manage.py migrate` та `python manage.py runserver`, рис. 5.2.

```
(masterD) C:\masterD>pip install -r requirements.txt
(masterD) C:\masterD>python manage.py migrate|
(masterD) C:\masterD>python manage.py runserver
```

Рисунок 5.2 – Запуск серверу

Для програми створено README файл та завантажено всі вихідні коди на платформу контролю версій GitHub, рис. 5.3.



Рисунок 5.3 – Readme файл

5.3 Методика роботи з вебплатформою

Програмна система запускається у веббраузері, тому користувачеві достатню перейти за адресним посиланням, що з'явиться після запуску серверу.

Головна сторінка сайту відображає основну мету створення подібної системи на створення нового алгоритму на основі НМ прогнозування позицій. У верхній частині сайту знаходиться навігаційне меню, що дозволяє користувачеві перемикатися між головною та інформаційною сторінкою. Під

навігаційним меню, знаходиться стрічка пошуку разом з кнопкою підтвердження вибору. Пошук відбувається за ім'ям гравця для якого буде прогнозуватися позиція. Під даним компонентом відображається базова інформація про гравця і з'являється кнопка прогнозування, після чого результат представлений в табличному вигляді. Роботу системи представлено на рис. 5.4.

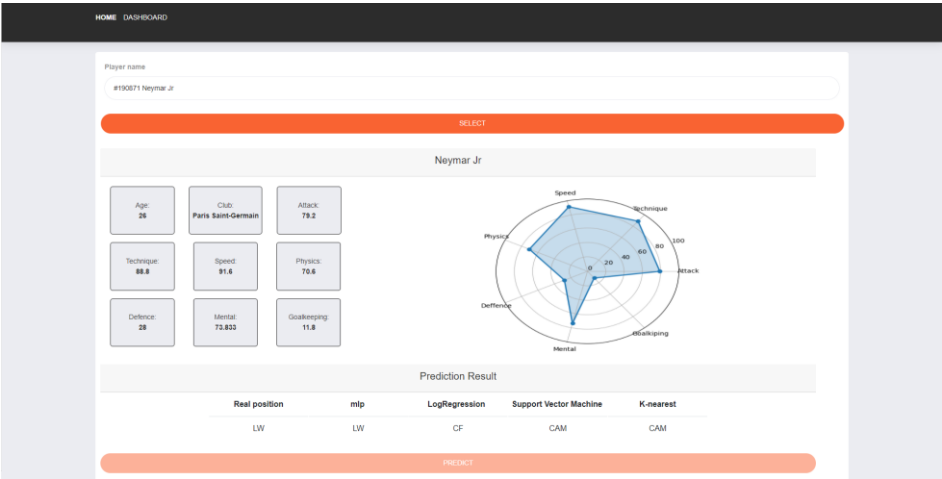


Рисунок 5.4 – Головна сторінка програми

На аналітичній сторінці системи, представлені показники точності для тренувальної та тестової вибірки в порівняльній таблиці, описано параметри архітектури НМ та виведено графіки втрат та точності тестової та тренувальної вибірок. Представлено на рис. 5.5.

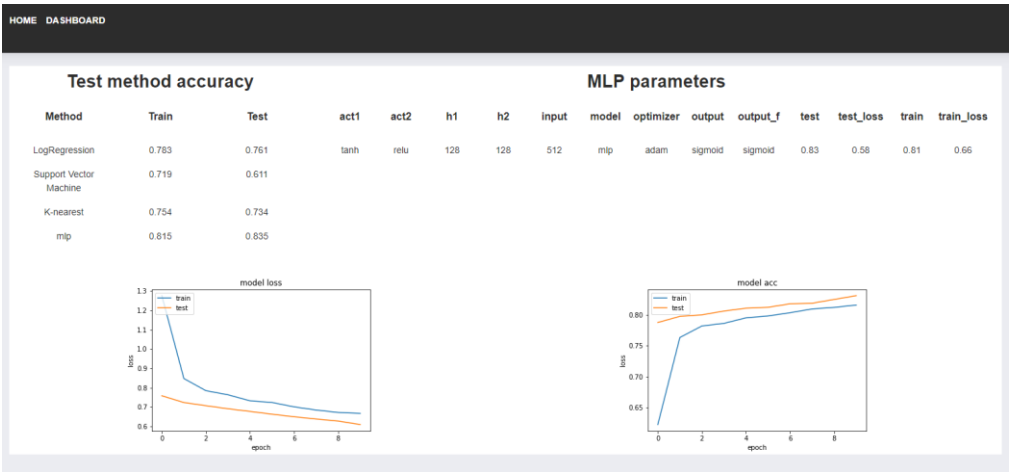


Рисунок 5.5 – Аналітична сторінка системи

На сторінці створення власних параметрів для гравця, користувач може заповнити необхідні поля та виконати прогнозування з ними. Ці значення будуть враховані при виконанні прогнозування позиції. Сторінку представлено на рис. 5.6.

The screenshot shows a web interface for entering player statistics. The top navigation bar includes 'HOME', 'DASHBOARD', and 'CUSTOM'. The main content area is titled 'ATTACK' and contains the following parameters:

- ATTACK:** Attacking Crossing (91), Attacking Finishing (96), Attacking Heading Accuracy (89), Attacking Short Passing (84), Attacking Volleys (98).
- TECHNIQUE:** Skill Dribbling (97), Skill Curve (78), Skill FK Accuracy (99), Skill Long Passing (97), Skill Ball Control (96).
- SPEED:** Movement Acceleration (91), Movement Spring Speed (96), Movement Agility (96), Movement Reactions (98), Movement Balance (83).
- PHYSICS:** Power Shot (83), Power Jumping (98), Power Stamina (96), Power Strength (97), Power Long Short (86).
- DEFENCE:** Defence Marking (43), Defence Standing Tackle (54), Defence Sliding Tackle (44).

At the bottom of the form is an orange button labeled 'PREDICT'.

Рисунок 5.6 – Введення своїх параметрів для гравця

5.4 Методика тестування системи

Тестування системи поділено на дві частини: перша – це тестування моделі для тестової та тренувальної вибірок, друга – це тестування серверної частини, а саме відповіді серверу на вебзапит.

5.4.1 Тестування моделі

Тестування створеної архітектури відбувалося за допомогою методів бібліотеки tensorflow та побудови графіків показника точності та показника

втрат. Результати представлені на рис. 5.7 – 5.8, а також на аналітичній сторінці системи.

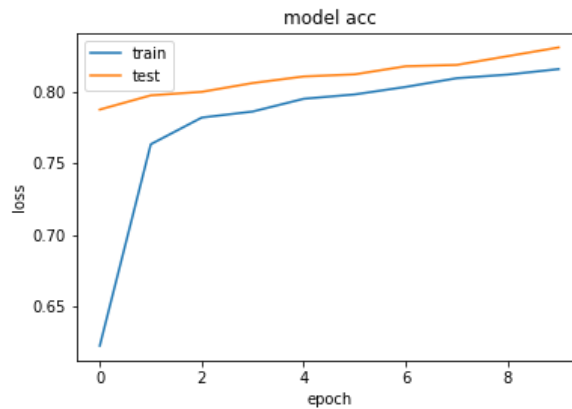


Рисунок 5.7 – Порівняння точності тренувальних та тестових даних

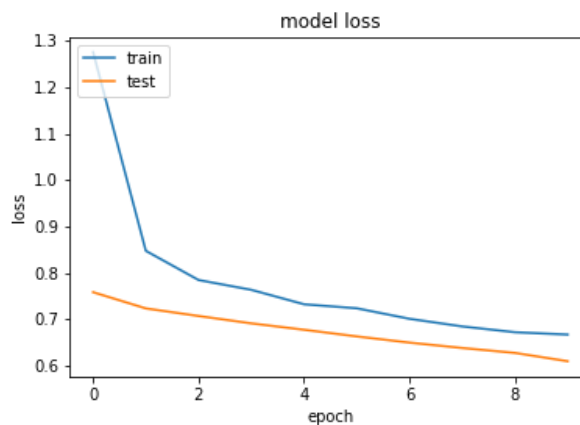


Рисунок 5.8 – Порівняння втрат тренувальних та тестових даних

Остаточні результати програми записані до бази даних та відображені у табл. 5.1 та на аналітичній сторінці.

Таблиця 5.1 – Порівняння точності методів прогнозування

Назва	Тренувальна	Тестова
Логістична регресія	0,78	0,76
Метод опорних векторів	0,71	0,61
К-найближчих сусідів	0,75	0,73
БШП	0,81	0,83

5.4.2 Тестування серверної частини

Тестування серверної частини полягає у перевірці роботи відповідей серверу на отримані URL-запити. Оскільки кількість запитів не є великою, для тестування було обрано програмний засіб Postman. Приклад роботи з програмою представлено на рис. 5.9.

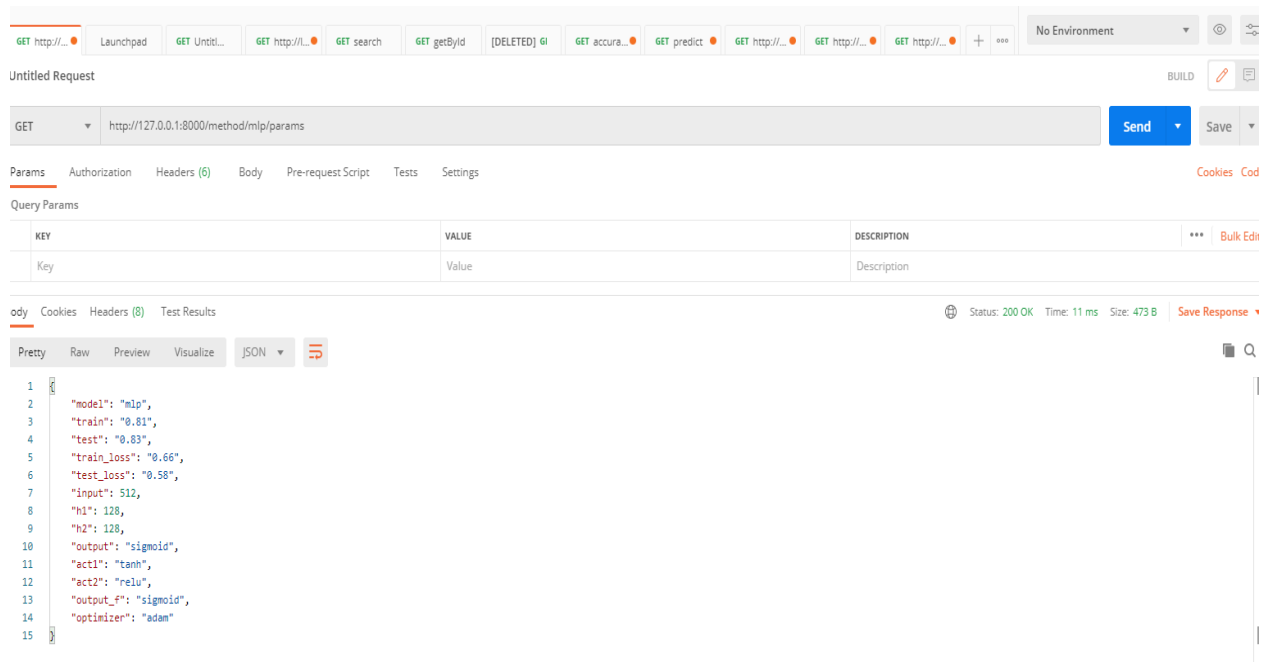


Рисунок 5.9 – Використання Postman для тестування вебзапитів

5.5 Висновки за розділом 5

В даному розділі було розглянуто призначення розроблюваної системи та описано основні підходи для взаємодії користувача. Окрім цього було описано види тестування для двох частин програми, безпосередньо створеної НМ та вебсерверу.

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

В даному розділі приводиться економічне обґрунтування створення програмного забезпечення (ПЗ), оцінка затрат на його створення, розрахунок економічної ефективності та терміну окупності [37].

6.1 Концепція економічного обґрунтування

Метою даної дипломної роботи є дослідження та використання нейронних мереж для прогнозування спортивної статистики. Точність та швидкість роботи системи є основною метою розроблюваного комплексу. Машинне навчання надає змогу швидко реалізовувати методи прогнозування з достатньою точністю, що впливає на прибуток компаній.

Машинне навчання базується на поєднанні біологічної, математичної та комп'ютерної наук, що вбачає застосування найкращих якостей з всіх наведених з метою емуляції роботи людського мозку. Нейронні мережі дозволяють відтворити алгоритми за якими людський мозок приймає рішення та в свою чергу автоматизувати різні процеси на виробництвах або в повсякденному житті.

Практична цінність роботи полягає в тому, що використання НМ за допомогою сучасних обчислювальних можливостей наближає якість прийняття рішень машин до людського. Прогнозування на основі спортивних даних вже застосовується в таких сферах як: спортивні аналітичні команди, комп'ютерні симулятори, гральний бізнес, тощо.

Застосування розроблюваного програмного продукту дозволить скороти час аналітики даних, які в багатьох випадках витрачаються на пошуку якісних джерел даних, побудові та налаштуванню методів прогнозування. Якщо розглядати систему, як засіб допомоги тренеру в підготовці до гри, то це дозволить зекономити час на розбір тактики та стратегії суперника. Що в свою чергу підвищує імовірність позитивного результату для команди.

Розроблена система повністю реалізує різні методи прогнозування позицій гравці, які базуються на статистичному аналізі та використанні нейронних мереж. Оптимізація роботи полягає у спрощенні аналізу суперника для тренера, розробки симуляція стартових складів в комп'ютерній грі, покращення коефіцієнтів в букмекерських конторах. Тобто, для побудови стартового складу для обраних команд скорочується час роботи спортивного аналітика. З економічної точки зору наведена вище оптимізація зменшить часові витрати на підготовку аналітичних звітів, тощо.

Якщо розглядати конкурентів, то слід відзначити проєкт «Opta – це потужна аналітична система, яка є одною із найвідоміших в спортивному світі. Система пропонує різні продукти від надання даних до власних аналітичних метрик та віджетів. Представляю собою вебсайт для таких видів спорту як: футбол, регбі, крикет, баскетбол, американський футбол, бейсбол. Також конкурентами можливо виділити різні букмекерські платформи, які пропонують розрахунок коефіцієнтів на різні події. В унікальність розробленої системи закладено гнучкості створеного методу для різних сфер, що дозволить охопити широкий простір ринку. Наведені вище аналоги працюють в окремо обраному напрямі.

6.2 Планування розробки програмного продукту

Для розрахунку витрат необхідно визначити етапи робіт та їх тривалість. У розробці беруть участь програміст протягом 3 місяців та консультант протягом 0,15 місяця. Розробка ПЗ починається третього вересня та повинна бути закінчена до двадцять сьомого листопада 2020 року.

Тривалість робіт визначають за формулою (6.1):

$$T_{ц} = \frac{Q}{R}, \quad (6.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

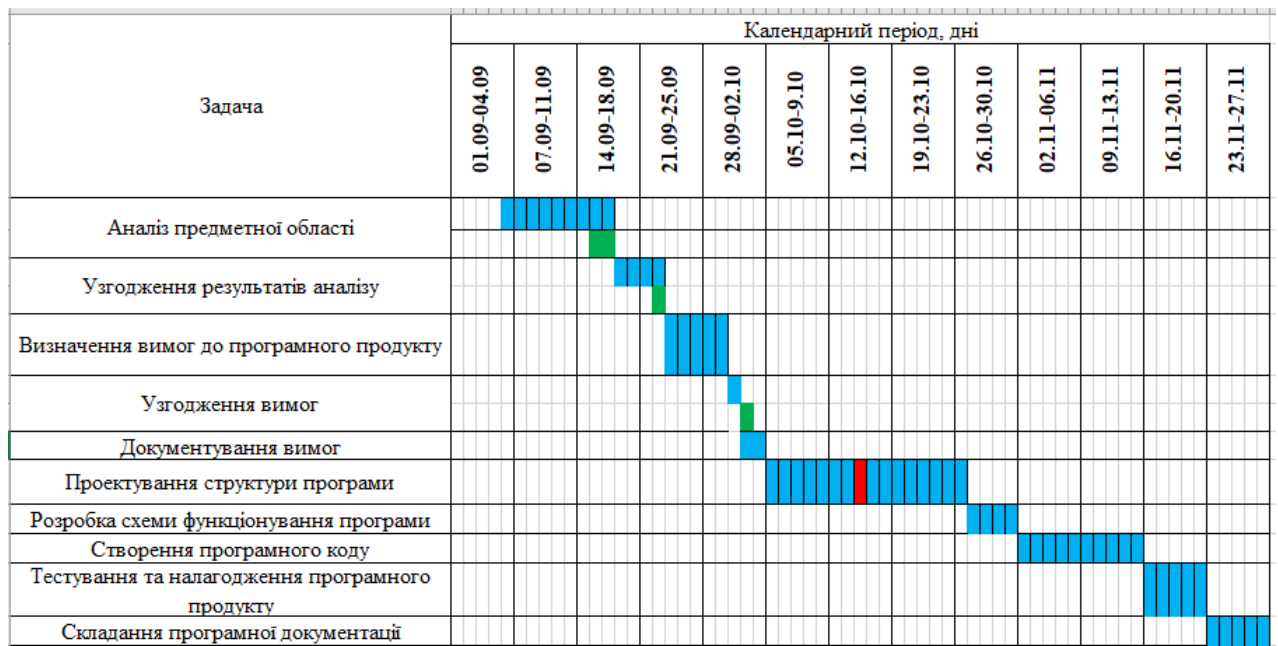
R – кількість виконавців, людей.

Трудомісткість проектування системи наведена у табл. 6.1.

Таблиця 6.1 – Характеристика робіт з розробки програми

Найменування роботи	Трудомісткість		Виконавці	Тривалість, дні
	люд.-дні	% до підсумку		
Аналіз предметної області	12	18%	Програміст, Консультант	10
Узгодження результатів аналізу	4	6%	Програміст Консультант	4
Визначення вимог до програмного продукту	4	6%	Програміст	4
Узгодження вимог	1	1%	Програміст Консультант	2
Документування вимог	2	3%	Програміст	2
Проектування структури програми	18	27%	Програміст	15
Розробка схеми функціонування програми	4	6%	Програміст	4
Створення програмного коду	14	21%	Програміст	10
Тестування та налагодження програмного продукту	4	6%	Програміст	5
Складання програмної документації	4	6%	Програміст	5
Підсумок	67	100		61

За даними табл. 6.1 складається зведений стрічковий графік розробки програмного продукту, який представляє собою таблицю, в першому стовпці якої розміщені в порядку збільшення строків початку виконання всі види роботи, а навпаки – календарний період їх виконання. Даний графік наведений на рис. 6.1.



Легенда: ■ – робота програміста, ■ – робота консультанта, ■ – святкові дні.

Рисунок 6.1 – Зведений стрічковий графік планування розробки

6.3 Розрахунок основної заробітної плати

Витрати за цією статтею складаються з планового фонду зарплати всіх категорій працівників, зайнятих у розробці програми. Розрахунок зарплати ведеться на підставі даних про трудомісткість, поданих у табл. 6.2.

Таблиця 6.2 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	9000	3	27000
Консультант	1	12000	0,15	1800
Підсумок	2			28800

6.4 Розрахунок додаткової заробітної плати

Додаткову заробітну плату приймають рівною 10% від основної заробітної плати працівників і розраховують за формулою (6.2):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1. \quad (6.2)$$

У нашому випадку отримаємо:

$$ЗП_{\text{дод}} = 28800 \cdot 0,1 = 2880 \text{ грн.}$$

6.5 Єдиний соціальний внесок

Єдиний соціальний внесок становить 22% і береться від основної та додаткової заробітної плати, формула (6.3):

$$ВТ_{\text{соц}} = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22. \quad (6.3)$$

Далі отримаємо:

$$ВТ_{\text{соц}} = (28800 + 2880) \cdot 0,22 = 6969,6 \text{ грн.}$$

6.6 Визначення витрат на матеріали

Використовується 3 найменування матеріалів: диск DVD+RW – 45 грн., картридж – 200 грн., і папір – 300 грн. (3 упаковка).

Витрати на матеріали вираховуються за наступною формулою (6.4):

$$ВТ_{\text{мат}} = \sum_{i=1}^n (Ц_i \cdot N_i \cdot (1 + K_{\text{т.з}}) - Ц_{io} \cdot N_{io}), \quad (6.4)$$

де $ВТ_{\text{мат}}$ – це витрати на матеріали, покупні напівфабрикати та комплектуючі вироби, грн.;

$K_{\text{т.з}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

$Ц_i$ – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн; N_i – потреба в i -му матеріалі, напівфабрикати і комплектуючому;

$Ц_{io}$ – ціна зворотних відходів i -го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

Визначено значення змінних:

$$C_{io} = 0; N_{io} = 0; K_{т.з} = 0,05.$$

Виконаємо обчислення:

$$BT_{\text{мат}} = (1 + 0,05) \cdot (45 + 200 + 300) = 572,25 \text{ грн.}$$

Разом, витрати на матеріали становлять 572,25 грн.

6.7 Витрати на спеціальне обладнання

Основною робочою машиною є ноутбук Dell Latitude 5500 в наступній конфігурації: Intel Core i5-8300U, 16 ГБ RAM, 512 ГБ SSD, Intel Graphics. Виходячи з даних, наданих виробником обладнання, а так само знятими системними показниками можна припустити, що середнє енергоспоживання ноутбука при подібній проєктній роботі складає 200 Ватт. Початкова ціна робочої машини – 30000 грн.

Амортизаційні відрахування визначають за формулою (6.5) [38]:

$$A = \Phi_6 \cdot \frac{H_a}{100}, \quad (6.5)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки %.

Норма амортизації для ПК становить 25%.

Балансова вартість обчислювальної техніки становить:

$$\Phi_6 = 7500 \text{ грн.}$$

Тоді отримаємо амортизаційні відрахування:

$$A = 7500 \cdot 0,25 = 1875 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію та допоміжні матеріали, формула (6.6):

$$C_e = N_n \cdot \Phi_{\text{еф}} \cdot K_{\text{зч}} \cdot K_{\text{зп}} \cdot C_{\text{ел.е}}, \quad (6.6)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{еф}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{\text{зч}}$ – середній коефіцієнт завантаження по часу;

$K_{\text{зп}}$ – коефіцієнт завантаження по потужності;

$C_{\text{ел.е}}$ – ціна одного кВт·год електроенергії, грн/(кВт·год).

Номінальна потужність робочої машини – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і по потужності рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,80 грн.

Отримуємо ціну експлуатації обладнання:

$$C_e = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,80 = 544,32 \text{ грн.}$$

Зарплата обслуговуючого персоналу розраховується відповідно до формули (6.7):

$$ЗП_{\text{обсл}} = ФЗП_r \cdot (1 + K_{\text{відр}}) \cdot \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обсл}}}, \quad (6.7)$$

де $ФЗП_r$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн;

$K_{\text{вдр}}$ – коефіцієнт, що враховує відрахування на соціальне страхування й в інші фонди;

$t_{\text{обсл}}$ – час протягом року, необхідне на технічне обслуговування ЕОМ, год/рік;

$\Phi_{\text{еф.обсл}}$ – річний ефективний фонд часу обслуговуючого персоналу, год/рік.

Місячна заробітна плата обслуговуючого персоналу становить 4723 грн., а річний фонд заробітної плати відповідно дорівнює 56676 грн. Річний фонд робочого часу обслуговуючого ПК працівника дорівнює 1750 год/рік. На обслуговування одного ПК витрачається 12 год в рік.

Отримуємо:

$$ЗП_{\text{обсл}} = 56676 \cdot (1 + 0,22) \cdot \frac{12}{1750} = 474,14 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і становить 225 грн.

Стаття «Інші витрати» приймається рівною 5% від суми всіх попередніх статей витрат на утримання і експлуатацію обладнання. Сума всіх попередніх статей дорівнює 3118,46 грн., 5% від суми становлять 155,92 грн.

Розраховані статті витрат на утримання та експлуатацію обладнання внесена до табл. 6.3.

Таблиця 6.3 – Кошторис витрат на утримання та експлуатацію обладнання

Найменування статей витрат	Сума, грн.
Амортизація обладнання	1875
Експлуатація обладнання (крім витрат на поточний ремонт)	544,32
Поточний ремонт обладнання	225

Продовження таблиці 6.3

Найменування статей витрат	Сума, грн.
Заробітна плата основна та додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	474,14
Інші витрати	155,92
Разом	3273,40

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (6.8):

$$C_{\text{МО}} = B_{\text{екс}} \cdot t_{\text{МО}}, \quad (6.8)$$

де $C_{\text{МО}}$ – витрати на оплату машинного часу, грн;

$B_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн/машино-год;

$t_{\text{МО}}$ – машинний час цифровий ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат за кошторисом «Витрати на утримання та експлуатацію устаткування (ЕОМ)» (табл. 6.4) на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. В результаті експлуатаційні витрати на одну годину машинного часу рівні [40]:

$$B_{\text{екс}} = \frac{3273,40}{1800} = 1,8 \text{ грн / машино-год.}$$

ЕОМ експлуатується 61 днів в одну зміну, що становить в сумі 488 годин. Таким чином, витрати на оплату машинного часу складуть:

$$C_{\text{МО}} = 1,8 \cdot 488 = 878,4 \text{ грн.}$$

6.8 Інші прямі витрати

До інших прямих витрат включаються витрати на використання при розробці системи комерційне ПЗ:

- пайове ПЗ, що використовується постійно при роботі ПК (Windows 10 Professional) – 5 620 грн. без ПДВ;
- цільове ПЗ, що для даної конкретної задачі (PyCharm) – 5 620 грн. без ПДВ;
- Цільове ПЗ, що для даної конкретної задачі (VsCode) – 0 грн. без ПДВ;

Розрахунки представлені на формулах (6.9) та (6.10):

$$S_{\text{пай.ПО}} = \frac{C_{\text{ПОWindows}} \cdot T_{\text{КТС}}}{\Phi_{\text{еф.КТС}} \cdot T_{\text{с.ПО}}}, \quad (6.9)$$

$$S_{\text{цел.ПО}} = C_{\text{ПОPyCharm}} + C_{\text{ПОVsCode}}, \quad (6.10)$$

де $S_{\text{цел.ПО}}$ – витрати на пайове ПЗ при розробці ПЗ з розрахунків, грн;

$C_{\text{ПОWindows}}$ – ціна за Windows (без ПДВ), грн;

$C_{\text{ПОPyCharm}}$ – ціна за PyCharm (без ПДВ), грн;

$C_{\text{ПОVsCode}}$ – ціна за VsCode (без ПДВ), грн;

$T_{\text{КТС}}$ – машинний час КТС, необхідний користувачеві для розробки системи, машино-год/рік;

$\Phi_{\text{еф.КТС}}$ – річний ефективний фонд часу роботи КТС, машино-год/рік;

$T_{\text{с.ПО}}$ – термін служби пайового ПЗ, років.

Таким чином розрахунки наведено нижче:

$$S_{\text{пай.ПО}} = \frac{5620 \cdot 488}{1800 \cdot 5} = 304,73 \text{ грн.},$$

$$S_{\text{цел.ПО}} = 5620 + 0 = 5620 \text{ грн.},$$

$$S_{\Sigma} = 304,73 + 5620 = 5924,73 \text{ грн.}$$

6.9 Розрахунок накладних витрат

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т. д.), утримання та експлуатацію будівель.

Накладні витрати включаються у вартість розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. В даному випадку накладні витрати складають 40% до основної заробітної плати розробників, що становить 15600 грн.

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведені в табл. 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.	Питома вага до підсумку, %
Основна заробітна плата	28880	50,12
Додаткова заробітна плата	2880	5,00
Відрахування на соцстрах та ін.	6969,6	12,09
Матеріали та комплектуючі	572,25	0,99
Витрати на спец. обладнання	878,4	1,52
Інші прямі витрати	5924,73	10,28
Накладні витрати	11520	19,99
Разом	57624,98	100
ПДВ	11525	
Разом (з ПДВ)	69149,98	

Таким чином, вартість проекту оцінюється в 69149,98 грн.

6.10 Розрахунок економічної ефективності програмного продукту

Ефективність прикладних НДР визначається [38], як зіставлення річного економічного ефекту від застосування результатів розрахункових досліджень в умовах виробництва і використання капіталовкладень для здійснення втілення їх у виробництво, формула (6.11):

$$E_{\text{фак}} = \frac{\text{Економ.ефект.}}{\text{Капіталовклад.}} \quad (6.11)$$

Зіставляючи фактичну ефективність з нормативною ($E_n = 0,1 \dots 0,15$) робиться висновок про доцільність досліджень. Якщо $E_{\text{фак}} \geq E_n$, то дослідження визнаються економічно ефективними. У зворотному випадку дослідження не слід виконувати, тому що економічна вигода від таких досліджень мала. Економічну ефективність досліджень можливо оцінити за терміном окупності витрат на їх проведення та втілення [40], формула (6.12):

$$T_{\text{ок}}^{\text{фак}} = \frac{\text{Капіталовклад}}{\text{Економ.ефект.}} \quad (6.12)$$

Фактичний термін окупності капіталовкладень зіставляється з нормативним терміном окупності капіталовкладень, прийнятий для промисловості (приблизно 7 років), та на основі порівняння робиться висновок про доцільність досліджень.

Для теоретичних досліджень в більшості випадків важко або навіть неможливо розрахувати економічний коефіцієнт, тому доцільно визначити їх техніко-економічну ефективність з урахуванням наступних показників:

- важливість дослідження;
- складності розробки;
- результативності та можливості використання.

Важливість теоретичного дослідження оцінюють за його призначенням:

- вирішення проблемних питань;
- задоволення вимог спеціальної техніки;
- пошук принципово нових конструкцій і технологічних рішень та

т. п.

Складність виконання роботи визначають порівнянням отриманих результатів даного дослідження з результатами відомих аналогічних досліджень з обліком грошових і трудових витрат на їх проведення.

Результативність НДР можна визначити за повнотою рішень поставленого завдання: отриманий результат відповідає плановому, задовільний (часткове рішення) або негативний.

Аналіз залежності між цими показниками та витратами на їх досягнення дає можливість кількісної оцінки техніко-економічної ефективності теоретичних НДР по формулі (6.13):

$$K_{\text{НДР}} = \frac{J^n \cdot R \cdot T}{V_{\text{НДР}} \cdot t_{\text{НДР}}}, \quad (6.13)$$

де $K_{\text{НДР}}$ – рівень ефективності дослідження (коефіцієнт техніко-економічної ефективності НДР);

J – важливість роботи;

n – показник використання результатів НДР;

R – результативність роботи;

T – технічна складність виконання НДР;

$V_{\text{НДР}}$ – витрати на проведення НДР;

$t_{\text{НДР}}$ – час проведення НДР.

При $K_{\text{НДР}} \geq 1$ НДР вважається ефективною, розрахунки наведено нижче:

$$K_{\text{НДР}} = \frac{2,5^2 \cdot 1 \cdot 2,5}{69,15 \cdot 0,22} = 1,03.$$

Після розрахунку отримуємо рівень ефективності дослідження вищий за 1, тому можна зробити висновок, що дана дослідницька робота є ефективною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Незважаючи на розвиток технічних засобів у сфері виробництва та підходів до забезпечення якості, безпеки роботи працівників – місце праці залишається небезпечним та шкідливим для здоров'я людини.

Розробка програмного забезпечення в більшості сферах ведеться в офісному приміщенні, в якому людину оточують обчислювальні пристрої, системи зв'язку, генератори струму та інші електричні прибори, що створюють електромагнітне випромінювання. Нажаль, на сьогоднішній день ми не достатнього наукових робіт для підтвердження шкідливості та впливу випромінювання на людину.

Охорона праці – це система законодавчих актів, соціально-економічних, організаційних, технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, що забезпечують безпеку збереження здоров'я та працездатності людини в процесі праці [39].

Основою охорони праці є системний та послідовний підхід до дослідження організації праці з точки зору безпеки, людино-машинної взаємодії, аналізу різних фізичних, хімічних, психологічних, соціальних чинників та їх вплив на організм людини.

Виконання правил техніки безпеки та санітарних норм залежить не тільки від виконання компанією своїх обов'язків перед працівником, а й від особистої відповідальності люди до вимог та правил, відношення до своїх колег.

Програмне рішення, що розробляється для аналізу спортивних подій передбачає роботу в обчислюваному центрі невеликої команди програмістів. Розміри приміщення $9 \cdot 6 \cdot 4$ м.

Площа приміщення становить $9 \cdot 6 = 54 \text{ м}^2$, об'єм – $9 \cdot 6 \cdot 4 = 216 \text{ м}^3$.

За природою виникнення шкідливі та небезпечні виробничі фактори поділяються на такі групи:

- фізичні (підвищення і знижена температура повітря, надмірна загазованість та запиленість повітря та ін.);
- біологічні (підвищений вміст у повітрі мікроорганізмів та ін.)
- психофізіологічні (нервово-емоційні перевантаження, розумове напруження та ін.)

Зазвичай небезпеки пов'язані з порушення правил пожежної безпеки, наприклад наявність необхідних інструментів для гасіння пожежі безпосередньо працівниками, або організації заходів з пожежної безпеки, що можуть призвести до розповсюдження пожежі.

У даному розділі проаналізовано наступні питання:

- електробезпека та пожежна безпека;
- склад повітря робочої зони та мікроклімат;
- шум та виробничі випромінювання;
- безпека технологічного обладнання та процесу;
- виробниче освітлення.

7.1 Заходи забезпечення безпеки

При роботі в обчислювальному центрі найбільшою небезпекою для здоров'я працівника є електричний струм.

Питання правильного електроживлення при об'єднанні комп'ютерів в локальні мережі мають важливе значення як для забезпечення електробезпеки користувачів, так і для безаварійної і безперебійної роботи обладнання.

Обладнання, що використовується в приміщенні обчислювального центру є споживачем електроенергії, що живиться від змінного струму 220 В, частотою струму 50 Гц від мережі з заземленою нейтраллю, та відноситься до електроустановок до 1000В закритого виконання.

Комп'ютери мають власні блоки живлення, Для перетворення змінного струму мережі у постійний комп'ютери використовують блоки живлення (5 В, 12 В), потужністю 350 – 1000 Вт. Так як усі прилади при підключенні до

електричної мережі яка заземляється, використовують систему TN, тому обчислювальні пристрої у приміщенні відносяться до категорії I захисту від ураження електричним струмом.

Для зменшення ризику ураження людини електричним струмом живлення всіх електричних пристроїв здійснюється спеціальним кабелем, корпуси всіх електричних пристроїв виготовляються з неструмопровідних матеріалів. У приміщенні для забезпечення безпечності всі кабелі сховані в спеціальні коробки, штепсельні розетки розташовані на висоті одного та 0,2 метри від підлоги, вимикачі на стінах знаходяться на висоті 1,2 метра від підлоги, використовується електропроводка трифазна, чотирьохжильна, з подвійною ізоляцією, електрична мережа вмикається і вимикається за допомогою пускової апаратури (рубильником). Відповідно до міжнародного стандарту СЕ зазначені міри електробезпеки відповідають вимогам, пропонованим до побутових приладів.

Відповідно до [40] в приміщенні розроблено заходи щодо забезпечення електробезпеки.

Також для запобігання ураження людини струмом в приміщенні використовується:

- захисне заземлення;
- окремими розетками по 220 В для кожного робочого місця з обчислювальним пристроєм;
- триполюсної вилки та розетки, для підключення обчислювальних пристроїв до струму, у яких відповідний полюс з'єднаний з шиною заземлення;
- мережевий фільтр до якого підключений кожен блок живлення обчислювального пристрою або периферійного пристрою;
- конденсатори для шунтування високочастотних перешкод живлячої мережі на землю через дріт захисного заземлення в мереживом фільтрі та при з'єднанні двох пристроїв (комп'ютера та принтера) інтерфейсним кабелем;

- окремий провід на загальний контур при з'єднанні пристроїв для ліквідування проблеми різниці потенціалів;
- ізолююча – лінолеум підлога.

Опір захисного заземлення в обчислювальному центрі становить 3 Ом (за стандартом – не більше 4 Ом) згідно з "Правилами улаштування електроустановок" (ПУЕ) також приміщенні встановлено автоматичні вимикачі, пристрій заземлення контуру згідно [41].

При виконанні робіт по ремонту і обслуговуванню ПЕОМ обслуговуючий персонал зобов'язаний керуватися "Правилами техніки безпеки при експлуатації електроустановок споживачами". До роботи не допускаються особи, які не пройшли навчання з техніки безпеки

Втома та синдром тунельний синдром також є розповсюдженим небезпечним фактором для здоров'я люди при роботі в обчислювальному центрі.

Для профілактики втоми робітників застосовуються специфічні методи, до яких можна віднести засоби відновлення функціонального стану зорового та опорно-рухового апарату, зменшення гіподинамії, підсилення мозкового кровообігу, оптимізацію розумової діяльності.

Також для профілактики втоми та запобігання статистичного навантаження робітників встановлені такі правила організації робочого місця:

- оптимальна висота клавіатури від підлоги – 65-75 см;
- можливість регулювання висоти і нахилу клавіатури (відстань від поверхні стола до середини клавіатури – не більше 30 мм, кут підйому клавіатури – від 2° до 15°);
- наявність у клавіатури підставки для рук та килимка для миші з захистом від тунельного синдрому;
- наявність стільця або крісла з підлокітниками;
- рекомендовано використовувати перерви в роботі 10 хв. через кожні дві години.

7.2 Заходи з забезпечення виробничої санітарії та гігієни праці

Відповідно до вимог [42] розроблені заходи щодо забезпечення виробничої санітарії та гігієни праці для обчислювального центру обладнаного обчислювальними пристроями з ВДТ.

Гігієнічна класифікація праці спрямована на гігієнічну оцінку умов та характеру праці на робочих місцях працівників та застосовуються на підприємствах, в установах, організаціях усіх форм власності у випадках, передбачених законодавством.

Так як роботи програмістів в обчислювальному центрі виконуються сидячи і не потребують фізичного напруження, то вони відносять до типу операторської роботи, даний тип роботи відносить до категорії 1а. Робоче місце програміста є постійним, робота проводиться у холодний та теплий періоди року тому рекомендується підтримувати оптимальні показники мікроклімату.

Для забезпечення оптимального рівня параметрів повітряного виробничого середовища використовуємо ДСН 3.3.6-042-99 «Санітарні норми мікроклімату виробничих приміщень» [43].

Встановлені наступні оптимальні значення параметрів мікроклімату:

- у холодний період року: температура 22-24°C; відносна вологість: 40 – 60%; швидкість переміщення повітря: 0,1 м/с;
- у теплий період року: температура 23-25°C; відносна вологість: 40- 60%; швидкість переміщення повітря: 0,1 – 0,2 м/с.

Оптимальні рівні позитивних (n+) і негативних (n-) іонів у повітрі приміщення з ПК нормовані згідно ГН 2152-80 «Санітарно-гігієнічні норми допустимих рівнів іонізації повітря виробничих та громадських приміщень» і становлять: $n+ = 1500-30000$ (шт. на 1 см^3); $n- = 3000-5000$ (шт. на 1 см^3). Підтримка оптимального рівня легких позитивних і негативних аероіонів у повітрі на робочих місцях забезпечено за допомогою біполярних коронних аероіонізаторів. В обчислювальному центрі передбачено: устрій системи

водяного опалення приміщення – для забезпечення необхідної температури повітря в холодний період року.

Дотримання вимог цих документів досягається оснащенням приміщень і транспортних засобів пристроями кондиціонування і вентиляції, дезодорації повітря, опалювання згідно вимог [43].

Зимою значення відносної вологості повітря часто можуть нижчими за встановлені норми і становити в середньому 30–40%, що приводить до надмірного висихання слизових оболонок очей, носа, горла та до нагромадження зарядів статичної електрики, що утворюються в процесі роботи комп'ютера. В приміщенні передбачена наявність 4 зволожувачів повітря, які допомагають підняти вологість до 60% для усунення таких проблем. Кількість зволожувачів зумовлена тим, що один такий апарат розраховано на приміщення площею до 15 м².

Літом для підтримки нормальної температури повітря в приміщенні використовуються три кондиціонери типу «спліт-система». Два кондиціонери розташовані у протилежних частинах та один по центру обчислювального центру дозволяють підтримувати температуру повітря не вище 24°C. Кількість кондиціонерів зумовлена тим, що один такий апарат розраховано на кондиціонування приміщення площею до 18 м².

Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється стомленість у зв'язку з підвищеними енергетичними витратами і нервово-психічною напругою, погіршується мовна комутація. Рівні звуку, рівні звукового тиску в октавних смугах частот, та еквівалентні рівні звуку на робочих місцях у приміщення нормуються згідно [44-45]. Оптимальний рів шуму в приміщення забезпечується за допомогою:

- використання блоків живлення ПК з вентиляторами на гумових підвісках;
- використання більш сучасного обладнання;

- розташування принтерів та різноманітного устаткування колективного користування на значній відстані від більшості робочих місць працівників;
- переведення жорсткого диска в режим сну (Standby), якщо комп'ютер не працює протягом визначеного часу.

Інтенсивне електромагнітне випромінювання постійно впливає на програміста в період роботи за дисплеєм, яке може стати причиною професійних захворювань. Комп'ютери потрапляють за визначенням в 5-й діапазон (низькі частоти). Для зниження потужності експозиційної дози рентгенівського випромінювання для роботи використовуються монітори з рідкокристалічними дисплеями.

Освітлення у приміщеннях з ЕОМ суміщене, при якому недостатнє за нормами природне освітлення доповнене штучним. Природне освітлення бічне, одностороннє. Вимоги до освітлення регламентовані [46].

Згідно з вимогами регламентованими в [46] в обчислювальному центрі коефіцієнт природної освітленості (КПО) має становити в середньому 1,1% (1,2% – у сонячний день та 1% – у хмарну погоду). Вікна в приміщенні обладнанні сонцезахисними металевими жалюзі з системою регульована для забезпечення відносної постійності природного освітлення. Коефіцієнт відбиття жалюзі становить 0,7. Робочі місця робітників розташовано так, щоб у поле зору робітника не потрапляли світлі поверхні світильників та вікон.

Люмінесцентні лампи ЛБ (білого світла) і ЛТБ (тепло-білого світла) потужністю 20, 40 або 80 Вт є найбільш прийнятними для приміщень, де працюють ПК.

Для освітлення приміщення розміром $9 \cdot 6 \cdot 4$ м. освітленістю $E_p = 400$ лк, коефіцієнтом відбиття стелі 70% і стін 50% розрахуємо кількість ламп необхідних для освітлення використовуючи люмінесцентні лампи типу ЛБ в світильниках ЛПО.

Далі наведено формулу (7.1), для знаходження індекс приміщення, формула (7.1):

$$i = \frac{A \cdot B}{h \cdot (A + B)} = \frac{9 \cdot 6}{4 \cdot (9 + 6)} = 0,9, \quad (7.1)$$

де A – довжина поміщення;

B – ширина поміщення;

H – висота поміщення.

Приймаємо коефіцієнт запасу $k = 1,6$ і коефіцієнт нерівномірності освітлення за формулою (7.2):

$$z = \frac{B_{\text{cp}}}{B_{\text{мін}}} = 1,1. \quad (7.2)$$

При індексі $i = 0,9$ отримаємо коефіцієнт використання світлового потоку 35%.

В два ряди ($N_p = 2$) розміщуємо світильники.

Використовуючи вище зазначені параметри розрахуємо необхідний світловий потік ламп в кожному ряду за формулою (7.3):

$$\Phi_p = \frac{E_n \cdot S \cdot z \cdot k}{N_p \cdot \eta} = \frac{400 \cdot 56 \cdot 1,1 \cdot 1,6}{2 \cdot 0,35} = 56320 \text{ лм}. \quad (7.3)$$

Далі розраховуємо кількість необхідних світильників за формулою (7.4), якщо в світильники встановити по дві лампи ЛБ ($n = 2$) потужністю 80 Вт і світловим потоком $\Phi_{\text{л}} = 5400$ лм.

$$N_p = \frac{\Phi_p}{n \cdot \Phi_{\text{л}}} = \frac{56320}{2 \cdot 5400} = 5 \text{ шт}. \quad (7.4)$$

Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски на поверхні екранів і клавіатури, використовуються сонцезахисні пристрої, на вікнах встановлені жалюзі або штори.

Організація та обладнання робочого місця в робочу приміщені має відповідати ергономічним вимогам з урахуванням особливостей трудовою дальності.

В приміщені обчислювального центру одночасно працюють 6 чоловік, кожний робітник має свій:

- обчислювальний пристрій, пристрою вводу (клавіатура, миш та інше), виводу(дисплей монітору, принтер, звукова колонка та інше) та набір периферійних пристроїв (принтер, сканер та інше);

- стіл;
- два ортопедичні стільці;
- тумбу.

Також кожний робітник може скористатися шафою для одягу, шафою з приладами, двома тумбами призначеними для використанню всіма робітниками.

Робоче місце працівника має площу 14 м², а об'єм – по 56 м³.

Приміщення має природне та штучне освітлення. Робочі місця розташовано так, щоб природне світло падало збоку. Робочі місця та взаємне розташування всіх його елементів відповідати антропометричним, фізичним і психологічним вимогам.

Також всі робочі місця організовані за такими правилами організації робочого місця:

- всім робітникам надані ергономічні та зручних миші і клавіатури;
- висота столу забезпечує розташування клавіатури від підлоги – 65-75 см, що позитивно впливає на працездатність робітника;
- кожна клавіатура має можливість міняти висоти і нахилу клавіатури;
- кожна клавіатура має підставку для рук;

- всі килимки для миші мають захист від тунельного синдрому (спеціальний виступ забезпечує правильне положення кисті);
- екрани монітору знаходяться на оптимальній відстані 700 мм від працівника;
- кожний стілець має підлокітник.

Загальний план обчислювального центру та робочих місць наведено на рис. 7.1.

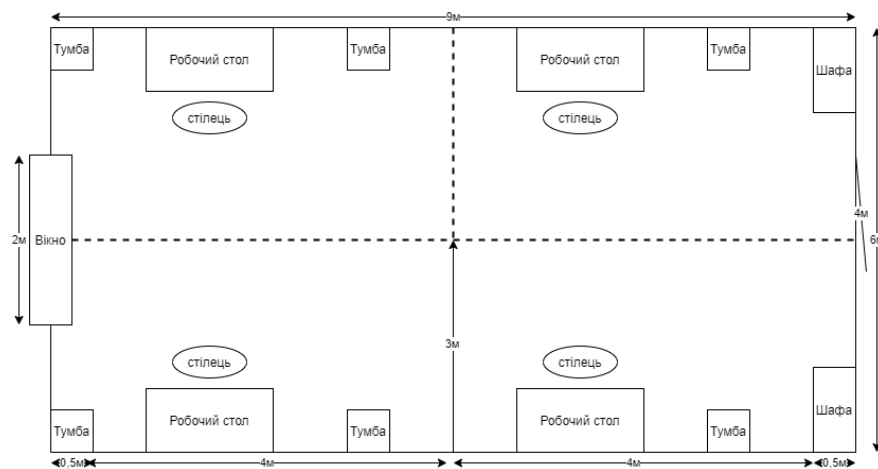


Рисунок 7.1 – Загальна схема обчислювального центру

7.3 Заходи з пожежної безпеки

Великою небезпекою для обчислювального центру є пожежа. Обчислювальний центр наповнений різноманітними електричними приладами, а пожежа основний фактор для їх знищення, що може призвести до великих матеріальних втрат. Так як в приміщенні обчислювального центру присутні всі три основні чинники, необхідні для виникнення пожежі: горючі речовини, окислення і джерела запалювання, то налагодження заходів пожежної безпеки в такому приміщенні є першочергово важливим.

Ймовірними причинами виникнення пожежу можуть бути несправність електрообладнання (кабелів, розеток), короткі замикання внаслідок виходу з

ладу чи експлуатації несправного електроустаткування (ПЕОМ, периферійних пристроїв), порушення правил протипожежної безпеки тощо.

Для забезпечення пожежної безпеки проводився пожежна профілактика, яка включає в себе комплекс організаційних і технічних заходів, спрямованих на забезпечення безпеки людей, на запобігання пожежі, обмеження її поширення, а також на створення умов для успішного гасіння пожежі.

Експлуатація ліній електромережі практично повністю унеможливорює виникнення електричного джерела загоряння в наслідок короткого замикання та перевантаження проводів. Застосовуються дроти з важко горючою і негорючою ізоляцією.

Для відводу теплоти від ПК у обчислювальному центрі постійно діє потужна система кондиціонування. Тому кисень, як окислювач процесів горіння, є в будь-якій точці обчислювального центру.

Обчислювальний центр відноситься до категорії «П-Па» згідно [47], а клас можливої пожежі визначається, як «А».

Для ліквідації пожежі у початковій стадії її розвитку силами персоналу об'єктів застосовуються первинні засоби пожежогасіння. До них відносяться: вогнегасники, пожежний інвентар (покривала з негорючого теплоізоляційного полотна, ящики з піском, пожежні відра, совкові лопати, ломы, сокири тощо), системи автоматичного пожежогасіння.

Так як приміщення (дослідницької лабораторії, конструкторського бюро, тощо) що обладнане ПК з ВДТ має площу 56 м², тому відповідно до вимог п. 3.8 [48] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості 3 штук (з розрахунку один вогнегасник с величиною заряду вогнегасної речовини 3 кг. і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 15 м.

Під час прийняття на роботу і в процесі праці усі працівники обчислювального центру повинні проходити протипожежний інструктаж,

перевірку знань з питань пожежної безпеки. Евакуаційні виходи і проходи утримуються постійно вільними та легкодоступними.

Відповідальний за протипожежний стан приміщень обчислювального центру повинен періодично (не рідше одного разу на місяць) перевіряти працездатність пристрою, який забезпечує автоматичне вимкнення системи вентиляції у разі пожежі, а також вогнедимозатримувальних пристроїв. Під час ремонту слід не допускати руйнування або порушення цілісності негорючих діафрагм у фальш-підлоговому просторі.

Забороняється працівникам проводити ремонт вузлів ЕОМ. Ремонтні роботи необхідно проводити в окремому, спеціалізованому для ремонту приміщенні.

Для своєчасного попередження пожеж та підвищення оперативності реагування при їх виникненні у приміщенні використовується такий комплекс заходів та правил:

- обов’язковий інструктаж персоналу з питань охорони праці;
- зокрема, правила пожежної безпеки у приміщеннях з ЕОМ;
- наявність системи автоматичної пожежної сигналізації з димовими пожежними оповіщувачами;
- наявність шляхів евакуації при виникненні пожежі;
- заборона залишати без нагляду електричну апаратуру, що використовується;
- заборона використання відкритого вогню у приміщенні;
- заборона зберігати постійно в залах ЕОМ носії інформації, запасні блоки та деталі в кількості більшій, ніж необхідно для поточного використання;
- заборона користуватися груповими розетками на горючій панелі;
- заборона використовувати килими та доріжки із синтетичних матеріалів;
- заборона ставити на вікна глухі ґрати (в разі потреби встановлюються ґрати, що розкриваються зсередини);

- заборона застосовувати електронагрівальні побутові прилади;
- розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу.

При виявленні ознак пожежі працівник повинен:

- негайно повідомити про це службу порятунку за номером телефону – 101 (112);
- вжити заходів щодо евакуації людей та матеріальних цінностей;
- вжити заходів до вжити заходів гасіння пожежі з використанням наявних вогнегасників та інших засобів пожежогасіння;
- Керівник підрозділу при виникненні пожежі зобов'язаний:
- упевнитися що викликані рятувальні служби;
- відключити подачу електроенергії в приміщення;
- проінформувати про пожежу керівництво;
- перевірити чи людей оповіщено про пожежу;
- організувати евакуацію працівників (за умови загрози життю людей)
- забезпечити дотримання техніки безпеки працівниками, які беруть участь у гасінні пожежі;
- організувати зустріч підрозділів Державної пожежної охорони;
- надати допомогу підрозділам Державної пожежної охорони під час локалізації та ліквідації пожежі.

7.4 Сили і засоби, які залучаються для проведення рятувальних та інших невідкладних робіт

Управління під час ліквідації наслідків надзвичайних ситуацій полягає у керівництві силами цивільного захисту (оперативна-рятувальна служба цивільного захисту, аварійно-рятувальні служби, формування цивільного захисту, спеціалізовані служби, пожежні-рятувальні підрозділи (частини),

добровільні формування цивільного захисту) при проведенні аварійно-рятувальних та інших невідкладних робіт (АРІНР) [49].

АРІНР – роботи, спрямовані на пошук, рятування і захист населення, уникнення руйнувань і матеріальних збитків, локалізацію зони впливу небезпечних чинників, ліквідацію чинників, що унеможливають проведення таких робіт або загрожують життю рятувальників.

Головна мета управління під час виконання АРІНР – забезпечити своєчасне та ефективне виконання завдань у зоні надзвичайної ситуації (НС) у найкоротші терміни та з мінімальними людськими й матеріальними втратами від наслідків НС наявними ресурсами (силами та засобами різноманітного призначення для виконання АРІНР).

Керівник аварійно-рятувального формування, що прибув у зону НС першим, бере на себе повноваження керівника робіт з ліквідації наслідків НС (керівника робіт) і виконує їх до прибуття призначеного у встановленому порядку керівника робіт.

За гострої необхідності керівник робіт має право самостійно вирішувати питання щодо:

- проведення заходів з евакуації;
- зупинки діяльності організацій, що перебувають у зоні НС;
- проведення АРІНР на об'єктах і територіях організацій, що перебувають у зоні НС;
- обмеження доступу людей у зону НС;
- розбронювання з метою ліквідації наслідків НС резервів матеріальних ресурсів організацій, що перебувають у зоні НС;
- використання у порядку, встановленому законодавством України, засобів зв'язку та транспорту, іншого майна організацій, що перебувають у зоні НС;
- залучення до проведення робіт з ліквідації наслідків НС позаштатних та громадських аварійно-рятувальних формувань, а також

рятувальників, що не входять до складу зазначених формувань, за наявності у них документів, що підтверджують їхню атестацію на проведення АРІНР;

- залучення на добровільній основі населення до проведення невідкладних робіт, а також окремих громадян, що не є рятувальниками, з їхньої згоди до проведення АРІНР;

- прийняття інших заходів, обумовлених розвитком НС і ходом робіт з її ліквідації;

Управління роботами починається з моменту виникнення НС і завершується після ліквідації її наслідків. Управління здійснюється, як правило, за добовими циклами, кожен із яких складається з таких етапів:

- збір даних про обстановку;
- аналіз і оцінка обстановки;
- підготовка висновків і пропозицій до рішення на проведення робіт;

- ухвалення (уточнення) рішення і доведення завдань до відома виконавців;

- організація взаємодії;
- забезпечення дій сил і засобів;
- організація управління.

Рішення на проведення АРІНР у зоні НС є основою управління; його ухвалює та організовує виконання керівник органу управління (керівник робіт з ліквідації НС).

Завдання підрозділам (формуванням), які беруть участь у ліквідації наслідків НС, доводять розпорядженнями, які заносяться у журнал (фіксуються засобами об'єктивного контролю).

Завдання щодо ліквідації наслідків надзвичайних ситуацій складаються із сукупності шести груп організаційних та інженерних заходів, які в умовах надзвичайної ситуації, стислих термінів виконання й обмежених місцевих ресурсів вимагають зусиль багатьох організацій:

- I група заходів (прогнозування надзвичайних ситуацій) вирішується силами державних і територіальних науково-дослідних установ, центрів та станцій спостереження.
- II група (локалізація або ліквідація надзвичайних ситуацій) та III група заходів (аварійно-рятувальні та інші невідкладні роботи) покладаються на територіальні та об'єктові аварійно-рятувальні формування, які створюються на підприємствах і в організаціях. За необхідності, можуть додатково залучатися військові формування і місцеве населення.
- IV група заходів (надання допомоги населенню постраждалих районів) перебуває у компетенції Уряду держави і здійснюється через місцеві органи влади й добровільні організації.
- V група заходів (відновлення уражених міст та інших об'єктів економіки) та VI група (інженерно-технічні заходи щодо стійкості роботи об'єктів економіки на випадок повторних дій стихійного лиха і їх запобігання) виконуються силами спеціалізованих формувань відповідних міністерств і відомств.

ВИСНОВКИ

Під час виконання дипломної кваліфікаційної роботи магістра було запропоновано новий метод прогнозування позицій гравців на основі модифікованого багатошарового перцептрона. Також було розроблено програмну систему, що реалізує роботу запропонованого методу.

Для цього були поставлені і вирішені такі задачі:

- виконано аналіз предметної області;
- досліджено методи прогнозування спортивної статистики;
- обрано мови програмування Python та Typescript;
- обрано середовища розробки PyCharm та Vs Code;

Виконано проєктування та розробку програмного забезпечення в результаті яких було отримано наступні результати:

- проведено аналіз, очищення, нормалізацію даних, формування тренувальної та тестової вибірок;
- реалізовано статистичні методи прогнозування;
- розроблено метод прогнозування на основі багатошарового перцептрона;
- створено програмну систему прогнозування позиції гравця на полі.

Розроблена програмна система, реалізує такі функції:

- робота з даними, підготовка та трансформація даних для використання в НМ;
- реалізація статистичних методів;
- реалізація методу прогнозування на основі багатошарового перцептрона;
- виведення проміжних та фінальних результатів роботи програми;
- виведення графічних результатів роботи програми;
- створення гравця за власними параметрами та прогнозування позиції для нього.

Розроблений програмний комплекс має практичну спрямованість та готовий до використання.

Під час розробки програми та аналізу розробленого програмного комплексу з'явилися ідеї з удосконалення, такі як: використання механізму паралелізації для зменшення часу роботи та більш ефективного використання ресурсів, також збільшення об'єму даних та тестування на реальних показниках гравців.

Нові можливості зможуть розширити функціональність програмного комплексу та підвищити його продуктивність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Carperz, A.M. Machine Learning With Big Data [Text] : Challenges and Approaches / A. L`heureux, K. Grolinger, A.M Carperz // IEEE Access. – 2017. – Vol. 5, Issue 1. – P. 7776 – 7797.
2. Muller, A. C. Introduction to Machine Learning with Python [Text] / A.C Muller, S. Guido. – Sebastopol : O`Reilly Media, 2017. – 2 p.
3. Stylianos, K. Predictive modeling of football injuries: dissertation ... doctor of philosophy / Kampakis Stylianos. – London: University College London, 2016.
4. Williams, J. Liverpool are using incredible data science during matches, and effects are extraordinary [Electronic resource]. – Access mode: <https://www.liverpool.com/liverpool-fc-news/features/liverpool-transfer-news-jurgen-klopp-17569689>
5. Skiera, B. Machine Learning With Big Data: Challenges and Approaches [Text] / A. Spann, B. Skiera // Journal of Forecasting. – 2009. – Vol. 28, – P. 55 – 72.
6. Stuart, K. Why clubs are using Football Manager as a real-life scouting tool [Electronic resource]. – Access mode: <https://www.theguardian.com/technology/2014/aug/12/why-clubs-football-manager-scouting-tool>.
7. Pitts. W. A logical calculus of the ideas immanent in nervous activity [Text] / W.S., McCulloch, W. Pitts // Bulletin of Mathematical Biophysics. – 1943. – Vol. 5, Issue 1. – P. 115 – 133.
8. Sukhadeve, A. Understanding Neural Network: A beginner's guide [Electronic resource]. – Access mode: <https://www.datasciencecentral.com/profiles/blogs/understanding-neural-network-a-beginner-s-guide>.
9. Rakhan, xG Model [Electronic resource]. – Access mode: <https://medium.com/@abhinavr8/xg-model-74c14ede6c58>.
10. Штучний нейрон [Електрон. ресурс]. – Режим доступу:

https://uk.wikipedia.org/wiki/Штучний_нейрон.

11. Mark, J. L. Introduction to Radial Basis Function Networks [Text] / J.L. Mark – Edinburgh: University of Edinburgh, 1996. – 9 – 11 p.
12. Advantages of Radial Basis Function Networks for Dynamic System Design [Text] / [Y. Hao, T. Xie, S. Paszczynski, B. Wilamowski] // IEEE Access. – 2011. – Vol. 58, Issue 12. – P. 5438 - 5450.
13. Kumar, R. Radial basis function network [Electronic resource]. – Access mode: <https://www.linkedin.com/pulse/radial-basis-function-network-rahul-kumar/>.
14. Kakaraparhi, V. Activation Functions in Neural Networks- What are they?, How they work? and Where to use them? [Electronic resource]. – Access mode: <https://prateekvishnu.medium.com/activation-functions-in-neural-networks-bf5c542d5fec>.
15. Activation Functions [Electronic resource]. – Access mode: https://mlcheatsheet.readthedocs.io/en/latest/activation_functions.html.
16. Rosenblatt, F. The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain [Text] / F. Rosenblatt // Psychological Review. – 1958. – Vol. 65, Issue 6. – P. 386–408.
17. Minsky, M Perceptrons. An Introduction to Computational Geometry. [Text] / M. Minsky, S. Papert // M.I.T Press. – 1969. – Vol. 165, Issue 3895. – P. 780–782.
18. The XOR Problem in Neural Networks [Electronic resource] Access mode: <https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b>.
19. Нейронные сети для начинающих. Часть 2 [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/313216/>.
20. Алгоритм обратного распространения ошибки (Back propagation algorithm) [Электрон. ресурс]. – Режим доступа: <https://wiki.loginom.ru/articles/back-propagation-algorithm.html>.

21. Banko, M. Scaling to very very large corpora for natural language disambiguation [Text] / M. Banko, E. Brill // Association for Computational Linguistics, Stroudsburg, July 2001. – P. 26 – 33.
22. Saini, R. Feature Scaling- Why it is required? [Electronic resource] / R. Saini. – Access mode: <https://medium.com/@rahul77349/feature-scaling-why-it-is-required-8a93df1af310>.
23. Pant, A. Introduction to Linear Regression and Polynomial Regression [Electronic resource] / A. Pant – Access mode: <https://towardsdatascience.com/introduction-to-linear-regression-and-polynomial-regression-f8adc96f31cb>
24. Arliss, W. Solving the SVM: Stochastic Gradient Descent and Hinge Loss [Electronic resource] / W. Arliss. – Access mode: <https://towardsdatascience.com/solving-svm-stochastic-gradient-descent-and-hinge-loss-8e8b4dd91f5b>.
25. Gandhi, R. Support Vector Machine — Introduction to Machine Learning Algorithms [Electronic resource] / R. Gandhi. – Access mode: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>.
26. Geron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. / Aurellen Geron. – Sebastopol, CA, USA: O'Reilly Media Inc., 2017 – 819 p.
27. Random Search [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Random_search.
28. Оптимізація гіперпараметрів [Електрон. ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Оптимізація_гіперпараметрів.
29. Hyperparameter Optimization With Random Search and Grid Search [Electronic resource]. – Access mode: <https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/>.
30. Jimmy, B. Adam: A Method for Stochastic Optimization [Text] /

P. Kingma, B. Jimmy // The 3rd International Conference for Learning Representations, San Diego, 7-9 May 2015.

31. Python 3.6.0 [Electronic resource]. – Access mode: <https://www.python.org/downloads/release/python-360/>.

32. Введение в Django [Электрон. ресурс]. – Режим доступа: <https://metanit.com/python/django/1.1.php>.

33. Angular 2 [Electronic resource]. – Access mode: <https://angular.io/>.

34. SQLite [Electronic resource]. – Access mode: <https://www.sqlite.org/index.html>.

35. The Python IDE for Professional Developers [Electronic resource]. – Access mode: <https://www.jetbrains.com/pycharm/>.

36. Microsoft Visual Studio Code [Электронный ресурс]. – Режим доступа: <https://code.visualstudio.com/>.

37. Остапенко, В.В. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальностей 7.080403 «Програмне забезпечення автоматизованих систем», 7.080402 «Інформаційні технології проєктування» і 7.091501 «Комп'ютерні та інтелектуальні системи і мережі» всіх форм навчання [Текст] / В.В. Остапенко, Е.М. Касьян, Є.М. Сердюк. – Запоріжжя : ЗНТУ, 2015. – 22 с

38. Методичні вказівки по економічному обґрунтуванню дипломних проєктів для студентів спеціальності 8.090801 «Мікроелектроніка та напівпровідникові прилади» [Текст] / Уклали В.О. Варинська, Т.А. Антоненко, – Запоріжжя: ЗНТУ, 2002 – 26 с.

39. Коробко О.В. Методичні вказівки до виконання розділу «Охорона праці та безпека у надзвичайних ситуаціях» у дипломних проєктах (роботах) студентів–магістрів спеціальностей 121 Інженерія програмного забезпечення (спеціалізація: інженерія програмного забезпечення), 121 Інженерія програмного забезпечення (спеціалізація: програмне забезпечення систем), 122 Комп'ютерні науки (спеціалізація: інформаційні технології

проектування), 122 Комп'ютерні науки (спеціалізація: системи штучного інтелекту), 123 Комп'ютерна інженерія (спеціалізація: комп'ютерні системи та мережі), 123 Комп'ютерна інженерія (спеціалізація: спеціалізовані комп'ютерні системи), 124 Системний аналіз (спеціалізація: інтелектуальні технології та прийняття рішень в складних системах) усіх форм навчання [Текст] / Укл.: О.В. Коробко – Запоріжжя: ЗНТУ, 2018 – 27 с.

40. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом: ДСТУ Б В.2.5-82:2016. – [Чинний від 2017-04-01]. – К.: ДП «УкрНДНЦ», 2016. – 109 с. – (Державний Стандарт України).

41. Правила безпечної експлуатації електроустановок: НПАОП 40.1-1.01-97. – [Чинний від 1997-10-06]. – К.: Держнагляд охорони праці, 1997. – 97 с. – (Нормативно-правовий акт охорони праці).

42. Санітарні норми мікроклімату виробничих приміщень: ДСН 3.3.6.042-99. – [Чинний від 1999-12-01]. – К.: МОЗ України, 1999. – 106 с. – (Державні санітарні норми).

43. Опалення, вентиляція та кондиціонування: ДБН В.2.5-67:2013. – [Чинний від 2014-01-01]. – К.: Мінрегіонбуд України, 2013. – 149 с. – (Державні будівельні норми України).

44. Санітарні норми виробничого шуму, ультразвуку та інфразвуку: ДСН 3.3.6.037-99. – [Чинний від 1999-12-01]. – К.: МОЗ України, 1999. – 132 с. – (Державні санітарні норми).

45. Державні санітарні норми виробничої загальної та локальної вібрації: ДСН 3.3.6.039-99. – [Чинний від 1999-12-01]. – К.: МОЗ України, 1999. – 29 с. – (Державні санітарні норми).

46. Природне і штучне освітлення: ДБН В.2.5-28-2018. – [Чинний з 2019-03-01]. – К.: Мінрегіон України, 2018. – 133 с. – (Державні будівельні норми України).

47. Визначення категорій приміщень, будинків та зовнішніх

установок за вибухопожежною та пожежною небезпекою: ДСТУ Б В.1.1-36:2016. – [Чинний від 2017-01-01]. – К.: Мінрегіонбуд України, 2016. – 66 с. – (Національний стандарт України).

48. Правила експлуатації та типових норм належності вогнегасників [Текст] – [Чинний від 2018-02-23]. – К.: МВС України, 2018. – 23 с. – (Правила).

49. Стеблюк М.І. Цивільна оборона та цивільний захист: підручник [Текст] / М.І. Стеблюк. – 3-тє вид., стер. – К. : Знання, 2013. – 487 с.

ДОДАТОК А
Технічне завдання

A.1 Вступ

A.1.1 Назва програми

Назва програми: «PositionPrediction»

A.1.2 Призначення та область застосування

Програма повинна прогнозувати позицій гравців на футбольному полі за їх характеристиками.

Програма представляє інтерфейс для зручної роботи з методами прогнозування.

A.2 Підстави для розробки

Підставою для розробки є завдання на дипломну роботу на тему: «Дослідження та програмна реалізація методів прогнозування спортивної статистики», затверджено наказом № 301 від 3 листопада 2020 року.

Актуальність роботи пояснюється тим, що програмно реалізовані методи будуть виконувати прогнозування, яке є основою для прийняття рішень під час участі зокрема футбольних команд у спортивних змаганнях, що в свою чергу впливає на результати, які досягає команда. В подальшому програма, що розробляється може бути використана в різних спортивних-аналітичних сервісах для розуміння тих чи інших подій спорту, а також для прибуткової діяльності в букмекерській сфері.

A.3 Призначення розробки

ПП буде спрямований на прогнозуванні результатів з використанням різних методів машинного навчання з відображенням проміжних та результуючих даних.

Програма буде виконувати наступні функції:

- реалізація декількох методів прогнозування;
- передбачення позиції гравця;
- виведення проміжних та фінальних результатів роботи програми;
- виведення інформації про запропонований метод прогнозування;
- обробку та аналіз даних;
- виведення графічних результатів прогнозування;
- порівняння реальної позиції гравця та прогнозованих.

A.4 Вимоги до програми

A.4.1 Вимоги до функціональних характеристик

ПП повинен забезпечувати виконання нижче перерахованих функцій:

- користувач може знайти гравця;
- користувач може спрогнозувати позицію гравця;
- користувач детально переглянути статистику;
- користувач може порівняти всі методи прогнозування для одного гравця;
- користувач може порівняти результати методів прогнозування для одного гравця;

A.4.2 Вимоги до надійності

Програма розміщена на комп'ютері. Працює додаток 24 години на добу, 7 днів на тиждень. Після оновлення функціоналу додатку треба перезапустити сервер.

A.4.3 Вимоги до технічних та програмних засобів

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

- наявність персонального комп'ютера (ПК);
- процесор із тактовою частотою 1.8 GHz та більше;
- оперативна пам'ять 8 ГБ та більше;
- вільний простір на жорсткому диску не менше, ніж 20 ГБ;
- операційна система Windows 10.

Умови до програмної частини, для нормального функціонування програми:

- наявність на ПК Angular 2+;
- наявність на ПК Python версії 3.6 (зі списком бібліотек, що будуть наведені в наступному розділі).

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- програмний код.

A.6 Стадії і етапи розробки

A.6.1 Стадії розробки

Розробка повинна бути проведена в три стадії:

- розробка технічного завдання;
- робоче проєктування;
- реалізації програмної системи;
- впровадження.

A.6.2 Етапи розробки

На стадії розробки технічного завдання повинен бути виконаний етап розробки ПЗ, узгодження та підтвердження сформованого технічного завдання.

Під час проєктування системи повинні бути проведені наступні роботи:

- створення програмних рішень, кодування;
- створення програмної документації;
- тестування створеної системи.

На стадії впровадження повинен бути виконаний етап розробки підготовка і передача програми.

A.6.3 Зміст робіт по етапах

На етапі розробки технічного завдання повинні бути виконані перераховані нижче роботи:

- постановка задачі;
- визначення і уточнення вимог до технічних засобів;
- визначення вимог до програми;
- визначення стадій, етапів і термінів розробки програми та документації на неї;
- узгодження і затвердження технічного завдання.

На етапі розробки програми повинна бути виконана робота з програмування (кодування) і налагодженні програми.

На етапі розробки програмної документації повинна бути виконана розробка програмних документів відповідно до вимог до складу документації.

На етапі випробувань програми повинні бути виконані перераховані нижче види робіт:

- проведення тестування моделей та серверної частини;
- проведення тестування графічного інтерфейсу (мануальне).

ДОДАТОК Б
Текст програми

Б.1 Клієнтська частина

Б.1.1 Файл HomeComponent.html

```

<div class="main-home-content">
  <form class="main-controls-content" [formGroup]="playerGroup"
    (ngSubmit)="submit()">
    <div class="form-group">
      <label for="player">Player name</label>
      <input type="text"
        (input)="playerChange($event.target.value)"
        class="form-control"
        placeholder="Enter player name"
        id="player"
        formControlName="player"
        [matAutocomplete]="autoPlayer">
      <mat-autocomplete autoActiveFirstOption
        #autoPlayer="matAutocomplete">
        <mat-option *ngFor="let option of playerFilteredOptions"
          [value]="option.viewFiled">
          {{option.viewFiled}}
        </mat-option>
      </mat-autocomplete>
    </div>
    <button class="btn btn-primary btn-round w-100" > SELECT
  </button>
</form>
<div class="main-info-content" *ngIf="selected !== undefined">
  <div class="card-header">
    <h2>{{selected.short_name}}</h2>
  </div>
  <div class="wrap-div">
    <div class="card-body">
      <p>Age: <span>{{selected.age}}</span></p>
      <p>Club: <span>{{selected.club}}</span></p>
      <p>Attack: <span>{{selected.attack | number}}</span></p>
      <p>Technique: <span>{{selected.technique | number}}</span></p>
      <p>Speed: <span>{{selected.speed | number}}</span></p>
      <p>Physics: <span>{{selected.physics | number}}</span></p>
      <p>Defence: <span>{{selected.defence | number}}</span></p>
      <p>Mental: <span>{{selected.mental | number}}</span></p>
      <p>Goalkeeping: <span>{{selected.goalkeeping |
number}}</span></p>
    </div>
    <div class="image-wrap">
      
    </div>
  </div>

  <div class="card-header">
    <h2>Prediction Result</h2>
  </div>
  <div class="result">
    <table class="table" >
      <tr class="row" >
        <th *ngFor="let r of results$ | async"> {{r.method}}</th>
      </tr>
      <tbody>
        <tr class="row" >
          <td *ngFor="let r of results$ | async">{{r.result}}</td>

```

```

        </tr>
      </tbody>
    </table>

    <button class="btn btn-primary btn-round w-100"
      [disabled]="this.loading"
      (click)="predict(selected.id) " > PREDICT </button>
  </div>
</div>
</div>

```

Б.1.2 Файл DashboardComponent.html

```

<div class="main-content-dashboard">
  <div class="main-part1-content">
    <div class='column-left'>
      <h1 class="m-2 title">Test method accuracy</h1>
      <div class="my-table">
        <div class="t-header">
          <h3>Method</h3>
          <h3>Train</h3>
          <h3>Test</h3>
        </div>
        <div class="t-row" *ngFor="let r of results$ | async">
          <p>{{r.model}}</p>
          <p>{{r.train | number}}</p>
          <p>{{r.test | number}}</p>
        </div>
      </div>
    </div>

    <div class="column-right">
      <h1 class="m-2 title">MLP parameters</h1>
      <div class="my-table">
        <div class="t-header">
          <h3 *ngFor="let item of paramsMLP | keyvalue"> {{item.key}}
</h3>
        </div>
        <div class="t-row" >
          <p *ngFor="let item of paramsMLP |
keyvalue">{{item.value}}</p>
        </div>
      </div>
    </div>
  </div>

  <div class="main-part2-content">
    <div class='column-left'>
      <div class="image-wrap">
        <img [src]="graph.loss">
      </div>
    </div>

    <div class="column-right">
      <div class="image-wrap">
        <img [src]="graph.acc">
      </div>
    </div>
  </div>

```

```

    </div>

  </div>
</div>

```

Б.1.3 Файл NavBarComponent.html

```

<nav class="navbar navbar-absolute fixed-top bg-default">
  <div class="header-navbar">
    <div class="menu">
      <ul class="navbar-nav" *ngFor="let item of menuList">
        <li [routerLinkActive]='["active"]' class="nav-item"> <a
class="nav-link"

[routerLink]="item.route"> {{item.title}} </a> </li>
      </ul>
    </div>
  </div>
</nav>

```

Б.1.4 Файл AppComponent.html

```

<div class="main-content">
  <!--Header-->
  <app-navbar class="header"></app-navbar>

  <div class="content">
    <router-outlet></router-outlet>
  </div>
</div>

```

Б.1.5 Файл Method/Types.ts

```

export interface IAccuracyMethod {
  model: string;
  train: number;
  test: number;
}

export class IParamsMLP {
  model: string;
  train: number;
  test: number;
  train_loss: number;
  test_loss: number;
  input: number;
  h1: number;
  h2: number;
  output: string;
  act1: string;
}

```

```

    act2: string;
    output_f: string;
    optimizer: string;
}

export class IParamsGraph {
    loss: string;
    acc: string;
}

export interface IPredictionPosition {
    method: string;
    result: string;
}

```

Б.1.6 Файл Player/Types.ts

```

export interface IPlayerParams {
    id: number;
    age: number;
    short_name: string;
    club: string;
    url: string;
    position: string;
    attack: number;
    technique: number;
    speed: number;
    physics: number;
    defence: number;
    mental: number;
    goalkeeping: number;
}

export interface ISearchPlayer {
    id: string;
    short_name: string;
}

```

Б.1.7 Файл Utils/SearchPlayer.ts

```

export class SearchUser {
    id: number;
    name: string;

    constructor(id: number, name: string) {
        this.id = id;
        this.name = name;
    }

    public static getIdFromViewFiled(viewfield: string): number {
        return +viewfield.split(' ')[0].slice(1, viewfield.length);
    }

    public get searchFiled(): string {

```

```

        return `${this.name}`.toLowerCase();
    }

    public get viewFiled(): string {
        return `#${this.id} ${this.name}`;
    }
}

```

Б.1.8 Файл MethodService.ts

```

import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { environment } from '../../environments/environment';
import { IParamsGraph, IParamsMLP, IPredictionPosition } from
'../models/method/types';
import { HttpClient } from '@angular/common/http';
import { IAccuracyMethod } from '../models/method/types';

@Injectable({
    providedIn: 'root'
})
export class MethodService {

    constructor(private http: HttpClient) {}

    public predict(patientId: number): Observable<IPredictionPosition[]>
    {
        return
        this.http.get<IPredictionPosition[]>(`${environment.apiUrl}method/pred
ict/${patientId}`);
    }

    public getAccuracy(): Observable<IAccuracyMethod[]> {
        return
        this.http.get<IAccuracyMethod[]>(`${environment.apiUrl}method/accuracy
/`);
    }

    public predictMLP(patientId: number):
    Observable<IPredictionPosition> {
        return
        this.http.get<IPredictionPosition>(`${environment.apiUrl}method/predic
t_mlp/${patientId}`);
    }

    public getAccuracyMLP(): Observable<IAccuracyMethod> {
        return
        this.http.get<IAccuracyMethod>(`${environment.apiUrl}method/accuracy_m
lp/`);
    }

    public getParamsMLP(): Observable<IParamsMLP> {
        return
        this.http.get<IParamsMLP>(`${environment.apiUrl}method/mlp/params`);
    }

    public getGraphMLP(): Observable<IParamsGraph> {
        return
        this.http.get<IParamsGraph>(`${environment.apiUrl}method/mlp/graph`);
    }
}

```

```

    }
  }
}

```

Б.1.9 Файл PlayerService.ts

```

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { SearchUser } from '../models/utils/search-player';
import { environment } from '../../environments/environment';
import { map } from 'rxjs/operators';
import { IPlayerParams, ISearchPlayer } from '../models/player/types';

@Injectable({
  providedIn: 'root'
})
export class PlayerService {

  constructor(private http: HttpClient) { }

  public searchPlayer(word: string): Observable<SearchUser[]> {
    return
    this.http.get<ISearchPlayer[]>(`${environment.apiUrl}players/search/${
word}`)
    .pipe(map(items => {
      return mapSearchUserResponseToSearchUser(items);
    }));
  }

  public getPlayer(patientId: number): Observable<IPlayerParams> {
    return
    this.http.get<IPlayerParams>(`${environment.apiUrl}players/${patientId
}`);
  }
}

export function mapSearchUserResponseToSearchUser(response:
ISearchPlayer[]): SearchUser[] {
  return response.map(r => new SearchUser(+r.id, r.short_name));
}

```

Б.1.10 Файл HomeComponent.ts

```

import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormControl, FormGroup, Validators } from
'@angular/forms';
import { SearchUser } from '../models/utils/search-player';
import { PlayerService } from '../services/player.service';
import { BehaviorSubject, forkJoin, Observable } from 'rxjs';
import { MethodService } from '../services/method.service';
import { IPlayerParams } from '../models/player/types';
import { IPredictionPosition } from '../models/method/types';
import { mergeMap, tap } from "rxjs/operators";

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',

```

```

        styleUrls: ['./home.component.scss'],
        host: {
            class: 'w-100'
        }
    })
    export class HomeComponent implements OnInit {
        public playerOptions: SearchUser[];
        public playerFilteredOptions: SearchUser[];
        public selected: IPlayerParams;
        public playerGroup: FormGroup;
        public results$: BehaviorSubject<IPredictionPosition[]> = new
BehaviorSubject<IPredictionPosition[]>(null);
        public loading = false;

        constructor(private formBuilder: FormBuilder,
                    private playerService: PlayerService,
                    private methodService: MethodService) { }

        get f(): any {
            return this.playerGroup.controls;
        }

        public ngOnInit(): void {
            this.playerGroup = this.formBuilder.group({
                player: new FormControl('', [
                    Validators.required,
                ]),
            });
        }

        public submit(): void {
            if (this.playerGroup.invalid) {
                return;
            }

            const patientId =
SearchUser.getIdFromViewFiled(this.f.player.value);

            this.playerService.getPlayer(patientId)
                .subscribe(player => {
                    console.log(player);
                    this.selected = player;
                });
        }

        public playerChange($event: string): void {
            this.loading = false;
            if ($event.length < 3) {
                this.playerOptions = [];
                this.playerFilteredOptions = [];
                return;
            }

            if ($event.length === 3) {
                this.playerService.searchPlayer($event).subscribe(items => {
                    this.playerOptions = items;
                    this.playerFilteredOptions = items;
                });
            } else {
                this.playerFilteredOptions = this.playerOptions
                    .filter(u => u.searchFiled.includes($event.toLowerCase()));
            }
        }
    }

```

```

    }
  }

  public predict(id: number): void {
    forkJoin(
      this.methodService.predict(id),
      this.methodService.predictMLP(id),
    )
    .subscribe([request1, request2 ] => {
      let result: any = [];
      result.push({ method: 'Real position', result:
this.selected.position});
      result.push(request2);
      result = result.concat(request1);
      this.loading = true;
      console.log(result);
      this.results$.next(result);
    });
  }
}

```

Б.1.11 Файл DashboradComponent.ts

```

import { Component, OnInit } from '@angular/core';
import { PlayerService } from '../services/player.service';
import { forkJoin, Observable } from 'rxjs';
import { IAccuracyMethod, IParamsGraph, IParamsMLP } from
'../models/method/types';
import { MethodService } from '../services/method.service';
import { map } from "rxjs/operators";

@Component({
  selector: 'app-dashboard',
  templateUrl: './dashboard.component.html',
  styleUrls: ['./dashboard.component.scss'],
  host: {
    class: 'w-100'
  }
})
export class DashboardComponent implements OnInit {

  results$: Observable<IAccuracyMethod[]> = new
Observable<IAccuracyMethod[]>(null);

  paramsMLP: IParamsMLP;
  graph: IParamsGraph;

  constructor(private methodService: MethodService) { }

  ngOnInit() {
    this.results$ = forkJoin(
      this.methodService.getAccuracy(),
      this.methodService.getAccuracyMLP(),
    )
    .pipe(map([request1, request2 ] => {
      let result: any = [];
      result = result.concat(request1).concat(request2);
      return result;
    }));
  }
}

```

```

        this.methodService.getParamsMLP().subscribe(response => {
            this.paramsMLP = response;
        });

        this.methodService.getGraphMLP().subscribe(response => {
            console.log(response);
            this.graph = response;
        });
    }
}

```

Б.2 Серверна частина

Б.2.1 Файл Controllers/Method.py

```

from django.http import JsonResponse
from project.services.db_context import DbContext
from project.services.predict_service import PredictService

db = DbContext()

def prediction(request, id):
    pred = PredictService(id)
    result = pred.statistic_prediction()

    return JsonResponse(result, content_type="application/json",
        safe=False)

def accuracy(request):
    pred = PredictService(0)
    result = pred.get_stat_results()

    return JsonResponse(result, content_type="application/json",
        safe=False)

def accuracy_mlp(request):
    pred = PredictService(0)
    result = pred.get_mlp_results()

    return JsonResponse(result, content_type="application/json",
        safe=False)

def accuracy_mlp_params(request):
    pred = PredictService(0)
    result = pred.get_mlp_params()

    return JsonResponse(result, content_type="application/json",
        safe=False)

def predict_mlp(request, id):
    pred = PredictService(id)
    result = pred.mlp_prediction()

    return JsonResponse(result, content_type="application/json",
        safe=False)

def get_mlp_graph(request):

```

```

    pred = PredictService(0)
    result = pred.get_mlp_graph()
    return JsonResponse(result, content_type="application/json",
safe=False)

```

Б.2.2 Файл Controllers/Player.py

```

from django.http import JsonResponse
from project.services.db_context import DbContext
from project.services.player_service import PlayerService

def search(request, param1):
    playerService = PlayerService()
    search_result = playerService.search(param1)
    return JsonResponse(search_result,
content_type="application/json", safe=False)

def getById(request, id):
    playerService = PlayerService()
    player = playerService.getPlayerById(id)
    return JsonResponse(player, content_type="application/json",
safe=False)

```

Б.2.3 Файл Services/db_context.py

```

import sqlite3
from project.domain.data_module.attribute_factory import
AttributeFactory
from project.settings import DATABASES, STATIC_ROOT

def dict_factory(cursor, row):
    d = {}
    for idx, col in enumerate(cursor.description):
        d[col[0]] = row[idx]
    return d

class DbContext:
    def execute_query_like(self, query, param=''):
        with sqlite3.connect(DATABASES['default']['NAME']) as
connection:
            connection.row_factory = dict_factory
            cur = connection.cursor()
            cur.execute(query, (('{}%'.format(param),))
            return cur

    def execute_query(self, query, params=''):
        with sqlite3.connect(DATABASES['default']['NAME']) as
connection:
            connection.row_factory = dict_factory
            cur = connection.cursor()

            if params == '':
                cur.execute(query)

```

```

        else:
            cur.execute(query, ((params),))
        return cur

    def execute_query_2params(self, query, t):
        with sqlite3.connect(DATABASES['default']['NAME']) as
connection:
            connection.row_factory = dict_factory
            cur = connection.cursor()
            cur.execute(query, t)
            connection.commit()
        return cur

    def search(self, word):
        query = "select sofifa_id, player_url, age, short_name,
long_name, club, player_positions " + \
            "from season19 " + \
            "where long_name like ?"

        response = self.execute_query_like(query, word)

        players = []
        for p in response:
            temp = {
                'id': p['sofifa_id'],
                'short_name': p['short_name'],
            }
            players.append(temp)

        return players

    def getPlayerById(self, id):
        query = "select * " + \
            "from season19 " + \
            "where sofifa_id=?"

        pandas_query = "select * from season19"
        response = self.execute_query(query, id).fetchall()[0]
        skill = AttributeFactory(pandas_query, id)
        path = STATIC_ROOT + '/' + id

        result = skill.create_shape(path)
        t = (path, id)
        update_query = "update season19 " + \
            "set skill_graphic=?" + \
            "where sofifa_id=?"

        self.execute_query_2params(update_query, t)
        return {
            'id': response['sofifa_id'],
            'age': response['age'],
            'short_name': response['short_name'],
            'club': response['club'],
            'position': response['player_positions'].split(", ")[0],
            'url': path,
            'attack': result['Attack'],
            'technique': result['Technique'],
            'speed': result['Speed'],
            'physics': result['Physics'],
            'defence': result['Deffence'],
            'mental': result['Mental'],
            'goalkeeping': result['Goalkiping']
        }

```

Б.2.4 Файл Services/player_service.py

```
from project.services.db_context import DbContext

class PlayerService:

    def search(self, word):
        db = DbContext()
        return db.search(word)

    def getPlayerById(self, id):
        db = DbContext()
        return db.getPlayerById(id)
```

Б.2.5 Файл Services/predict_service.py

```
import project.domain.data_module.preprocessing as pre
from project.domain.data_module.attribute_factory import AttributeFactory

from project.domain.ml_module.mlp_prediction import MlpPrediction
from project.domain.ml_module.stat_prediction import StatPrediction
from project.settings import STATIC_ROOT

class PredictService:

    def __init__(self, id):
        self.id = id

    def data_processing_by_user(self):
        X_train_full, y_train_full, X_test_full, y_test_full = pre.loading_data()
        p = pre.Preprocess()
        X_train, y_train, X_test, y_test = p.preprocessing(X_train_full, X_test_full, y_train_full, y_test_full)
        id_player = [int(self.id)]

        user_set = list(map(lambda x: x in id_player, X_train_full.index))
        player_train = X_train[user_set]
        return player_train

    def statistic_prediction(self):
        data = self.data_processing_by_user()
        stat = StatPrediction()
        result = stat.prediction(data)

        return result

    def mlp_prediction(self):
        data = self.data_processing_by_user()
        mlp = MlpPrediction()
        result = mlp.prediction(data)

        return result

    def get_stat_results(self):
```

```

stat = StatPrediction()
acc = stat.get_accuracy()
res = []
for r in acc:
    res.append({
        'model': r['models'],
        'train': r['train'],
        'test': r['test']
    })
return res

def get_mlp_results(self):
    mlp = MlpPrediction()
    r = mlp.get_accuracy()[0]
    mlp.create_model()
    return {
        'model': 'mlp',
        'train': r['train'],
        'test': r['test'],
    }

def get_mlp_params(self):
    mlp = MlpPrediction()
    r = mlp.get_accuracy()[0]

    return {
        'model': 'mlp',
        'train': '{:0.2f}'.format(r['train']),
        'test': '{:0.2f}'.format(r['test']),
        'train_loss': '{:0.2f}'.format(r['train_loss']),
        'test_loss': '{:0.2f}'.format(r['test_loss']),
        'input': r['input'],
        'h1': r['h1'],
        'h2': r['h2'],
        'output': r['output'],
        'act1': r['act1'],
        'act2': r['act2'],
        'output_f': r['output'],
        'optimizer': r['optimizer'],
    }

def get_mlp_graph(self):
    return {
        'loss': 'model_loss.png',
        'acc': 'model_acc.png'
    }

```

Б.2.6 Файл ml_module/mlp_prediction.py

```

import pickle
from keras.engine.saving import load_model
from keras.models import Sequential
from keras.layers import Dense, Dropout

from project.domain.data_module.preprocessing import encode
from project.services.db_context import DbContext
from project.settings import ML_ROOT

class MlpPrediction:

```

```

files = [{'mlp.h5': 'MLP'}]

def create_model(self, neurons_input=1,
                 level1=1,
                 level2=1,
                 a1='relu',
                 a2='relu',
                 opt='adam',
                 a_output='softmax'):
    # create model
    model = Sequential([
        Dense(neurons_input, activation='relu'),
        Dense(level1, activation=a1),
        Dense(level2, activation=a2),
        Dense(16, activation=a_output)
    ])
    model.compile(optimizer=opt,
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
    return model

def get_params(self):
    neurons_input = [512]
    level1 = [128]
    level2 = [128]
    epochs = [10]
    a1 = ['tanh']
    a2 = ['relu']
    opt = ['adam']
    output_a = ['sigmoid']
    param_grid = dict(model__neurons_input=neurons_input,
                      model__level1=level1,
                      model__level2=level2,
                      model__a1=a1,
                      model__a2=a2,
                      model__a_output=output_a,
                      model__opt=opt,
                      model__epochs=epochs)

    return param_grid

def get_accuracy(self):
    db = DbContext()
    query = "select * from results_mlp"
    result = db.execute_query(query).fetchall()
    print(result)
    return result

def prediction(self, params):
    model = self.loading_model()
    value = model.predict_classes(params)
    return {
        'method': 'mlp',
        'result': encode(value[0])
    }

def loading_model(self):
    model = load_model(ML_ROOT + '\\\\' + 'mlp.h5')
    return model

```

Б.2.7 Файл ml_module/stat_prediction.py

```

from sklearn.linear_model import LogisticRegressionCV
import numpy as np
import pickle
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm

from project.services.db_context import DbContext
from project.domain.data_module.preprocessing import encode
from project.settings import ML_ROOT

def get_score(y_true, y_pred):
    bools = list(map(lambda x: x[0] == x[1], zip(y_pred, y_true)))
    return np.array(bools).sum() / len(y_true)

class StatPrediction:
    files = [{'log_reg_model.pkl': 'LogRegression'},
             {'svc_model.pkl': 'Support Vector Machine'},
             {'knn_model.pkl': 'K-nearest'}]

    def logistic_regression(self, x_train, y_train):
        model = LogisticRegressionCV(multi_class='multinomial',
solver='lbfgs', n_jobs=-1)
        model.fit(x_train, y_train)
        return model

    def knn_model(self, x_train, y_train):
        KNN_model = KNeighborsClassifier(n_neighbors=5)
        KNN_model.fit(x_train, y_train)
        return KNN_model

    def svc_model(self, x_train, y_train):
        SVC_model = svm.SVC()
        SVC_model.fit(x_train, y_train)
        return SVC_model

    def load_model(self, file):
        with open(file, 'rb') as file:
            pickle_model = pickle.load(file)
        return pickle_model

    def prediction(self, params):
        models = self.loading_models()
        result = []
        for file in self.files:
            for key in file:
                value = models[file[key]].predict(params)
                result.append({
                    'method': file[key],
                    'result': encode(value[0])
                })

        return result

    def loading_models(self):
        models = {}
        for file in self.files:
            for key in file:
                path = ML_ROOT + '\\\\' + key

```

```

        models[file[key]] = self.load_model(path)
    return models

def get_accuracy(self):
    db = DbContext()
    query = "select * from results"
    result = db.execute_query(query).fetchall()
    return result

```

Б.2.8 Файл tune-mode.ipynb

```

#%%

from keras.utils import np_utils
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
import loading_data as ld
from imblearn.pipeline import Pipeline
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv1D, MaxPooling1D
from sklearn.model_selection import RandomizedSearchCV,
TimeSeriesSplit, GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier
from imblearn.over_sampling import SMOTE
from ml.tune_model import create_model, getParams, create_model_one
from preprocessing import preprocess_B, preprocessing
from stats.stat_predictions import Prediction

#%%

X_train_full, y_train_full, X_test_full, y_test_full =
ld.loading_data()

#%%

X_train, y_train, X_test, y_test = preprocessing(X_train_full,
X_test_full, y_train_full, y_test_full)

#%%

print(X_train.shape)
print(y_train.shape)

#%%

model=KerasClassifier(build_fn=create_model, verbose=2)

pipe = Pipeline([
    ('oversample', SMOTE(random_state=12)),
    ('model', model)
])

#%%

params = getParams()

#%%

rs_keras = RandomizedSearchCV(pipe, params , cv=5, scoring='accuracy',

```

```
refit='roc_auc', verbose=3, n_jobs=1, random_state=42)
grid_result = rs_keras.fit(X_train, y_train)

#%%

from sklearn.metrics import classification_report, confusion_matrix
y_pred=rs_keras.predict(X_test)
clfreport = classification_report(y_test, y_pred)

#%%
```

ДОДАТОК В
Слайди презентації

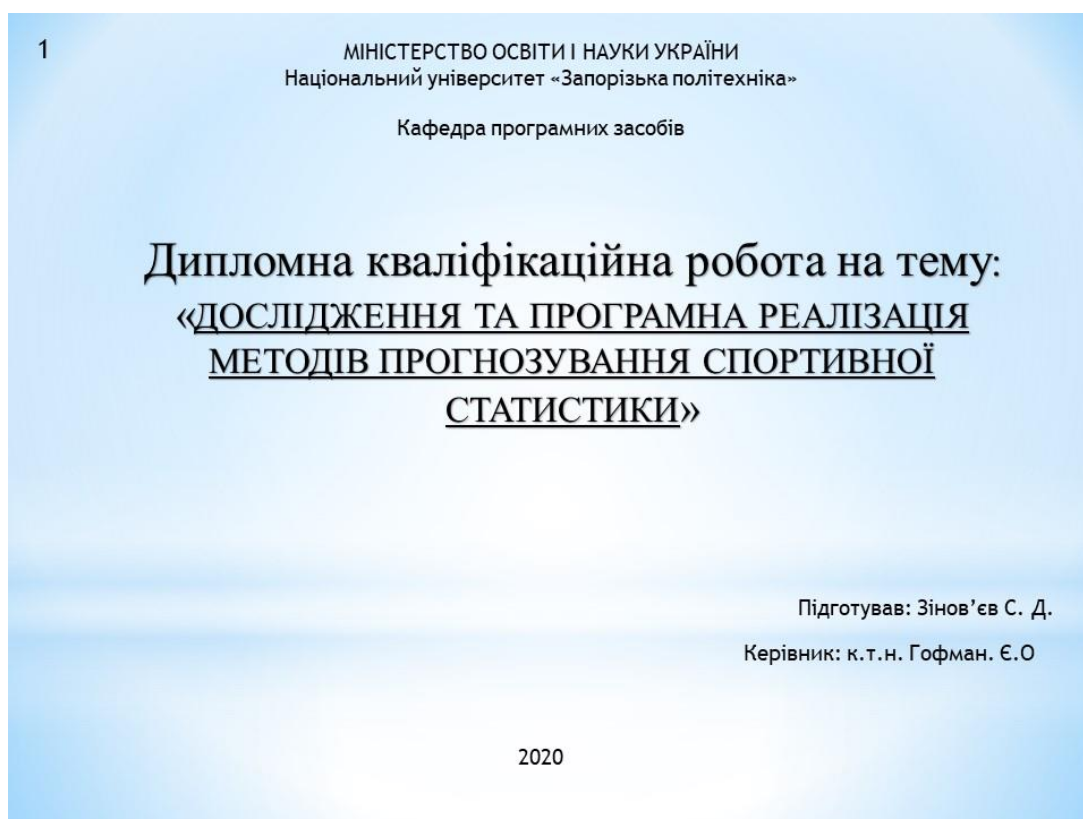


Рисунок В.1 – Дипломна кваліфікаційна робота

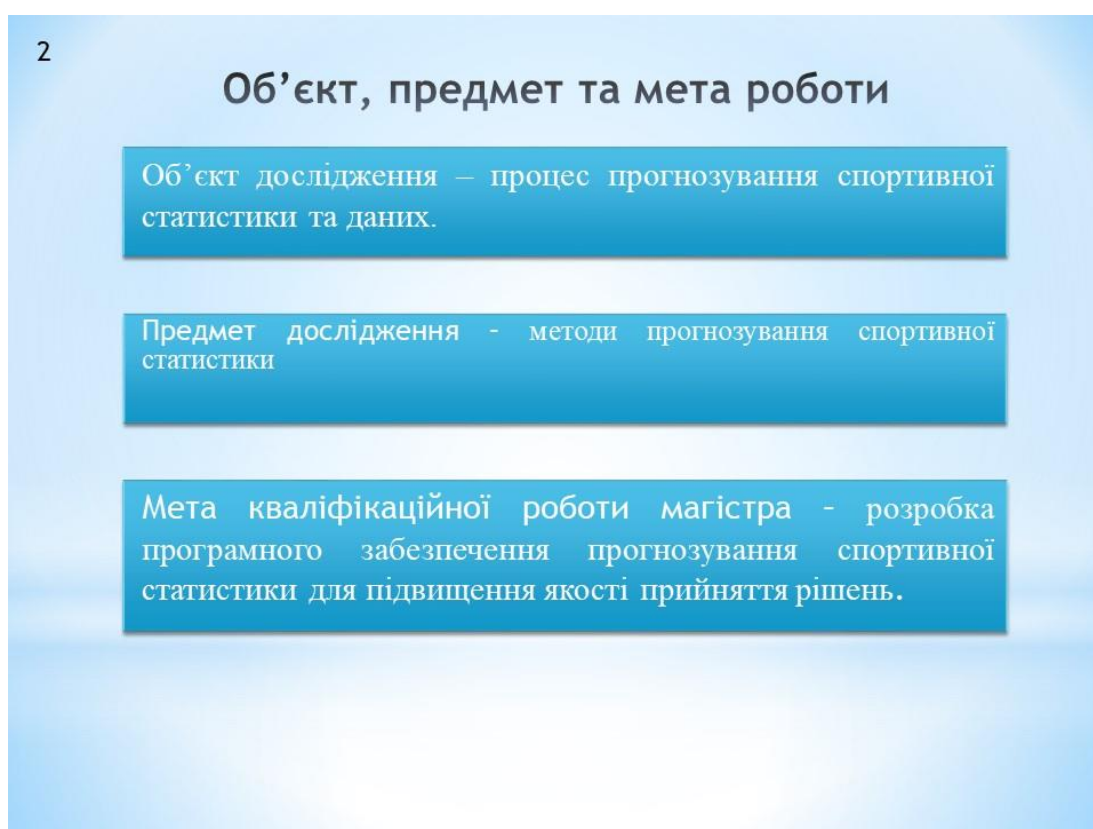


Рисунок В.2 – Об'єкт, предмет та мета роботи

3

Завдання до роботи



Рисунок В.3 – Завдання до роботи

4

Вирішувана задача

Прогнозування позиції гравця на футбольному полі за його характеристиками

Оскільки ми оперуємо позиціями, їх можливо представити як класи і завдання можливо сформулювати наступним чином: «передбачити клас гравця на футбольному полі за його параметрами».

Прогнозування позицій гравця є актуальною задачею для підтвердження деяких гіпотез:

- ✓ впливають різні фізіологічні характеристики, де саме буде грати гравець
- ✓ знаходження зв'язку між його параметрами та позицією.

Практична цінність: подібна модель може бути використана в різних сферах, наприклад скаутами, аналітиками для прогнозування складу на матч, в відеоіграх для симуляції складу та в подальшому результаті матчу, тощо.

Рисунок В.4 – Вирішувана задача

5

Розроблений метод

ПІДГОТОВКА ДАНИХ

1. Очистка даних – видалення порожніх значень або несуттєвих характеристик.
2. Нормалізація даних – приведення даних до єдиного діапазону для зменшення викидів. В даний процес також було додано процес трансформації категоріальних даних;

ВІДБІР ГІПЕРПАРАМЕТРІВ

Процес знаходження найкращих комбінацій гіперпараметрів для НМ. Було виконано за допомогою `tensorflow.RandomizeSearchCV`

НАВЧАННЯ. ТЕСТУВАННЯ. ВІДОБРАЖЕННЯ РЕЗУЛЬТАТІВ

1. Побудова моделі за знайденими параметрами
2. Навчання моделі на тренувальній вибірці
3. Виконання тестування та побудова графіків точності та втрат

Рисунок В.5 – Розроблений метод

6

Основні модифікації

Гіперпараметри	Значення
Вхідна кількість нейронів	[512, 256]
Кількість прихованих шарів	1-3
Нейрони для шар1	[256, 128, 64]
Нейрони для шар2	[128, 64]
Функція активації вхідний шар	relu
Функція активації шар1	[relu, softmax, tanh, sigmoid, hard_sigmoid]
Функція активації шар2	[relu, softmax, tanh, sigmoid, hard_sigmoid]
Вихідна функція	[sigmoid, softmax]
Функція оптимізації	[adam, adamax]



Гіперпараметри	Значення
Вхідна кількість нейронів	2
Кількість прихованих шарів	512
Нейрони для шар1	128
Нейрони для шар2	18
Функція активації вхідний шар	relu
Функція активації шар1	tanh
Функція активації шар2	relu
Вихідна функція	sigmoid
Функція оптимізації	adam

Рисунок В.6 – Основні модифікації

7

Вибір інструментів розробки

Критерії порівняння	Мови програмування			
	C++	Java	C#	Python
Складність мови	++	±	±	-
Математичний апарат	++	±	±	++
Простота розроблення графічного інтерфейсу	±	±	++	±
Продуктивність роботи	++	+	++	±

Обрано:

Python + Django (Сервер) + Machine Learning Algorithm
Typescript + Angular2 (Інтерфейс)

Рисунок В.7– Вибір інструментів розробки

8

Можливості користувача

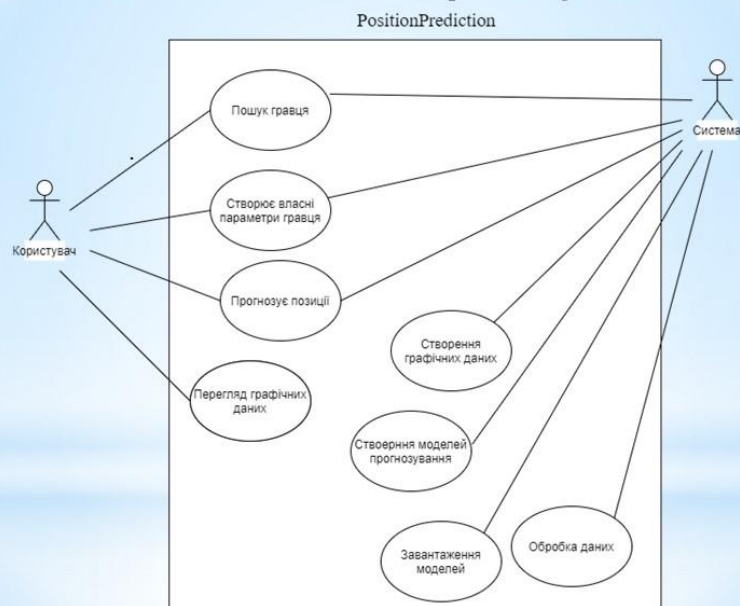


Рисунок В.8 – Можливості користувача

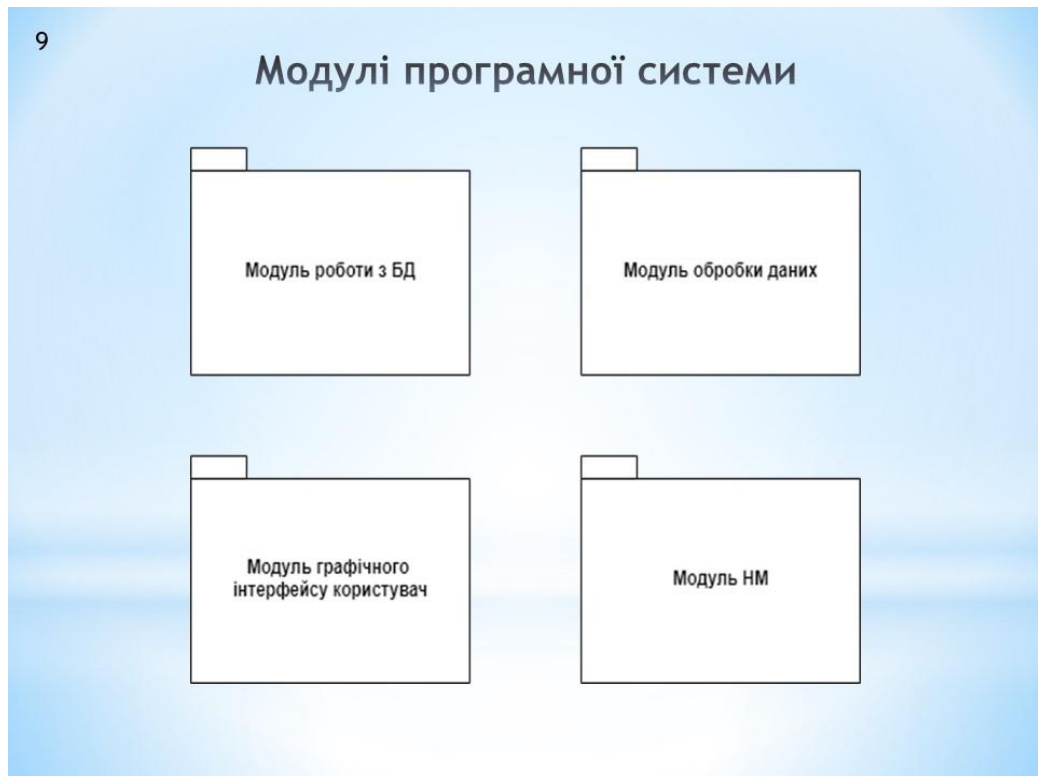


Рисунок В.9 – Модулі програмної системи

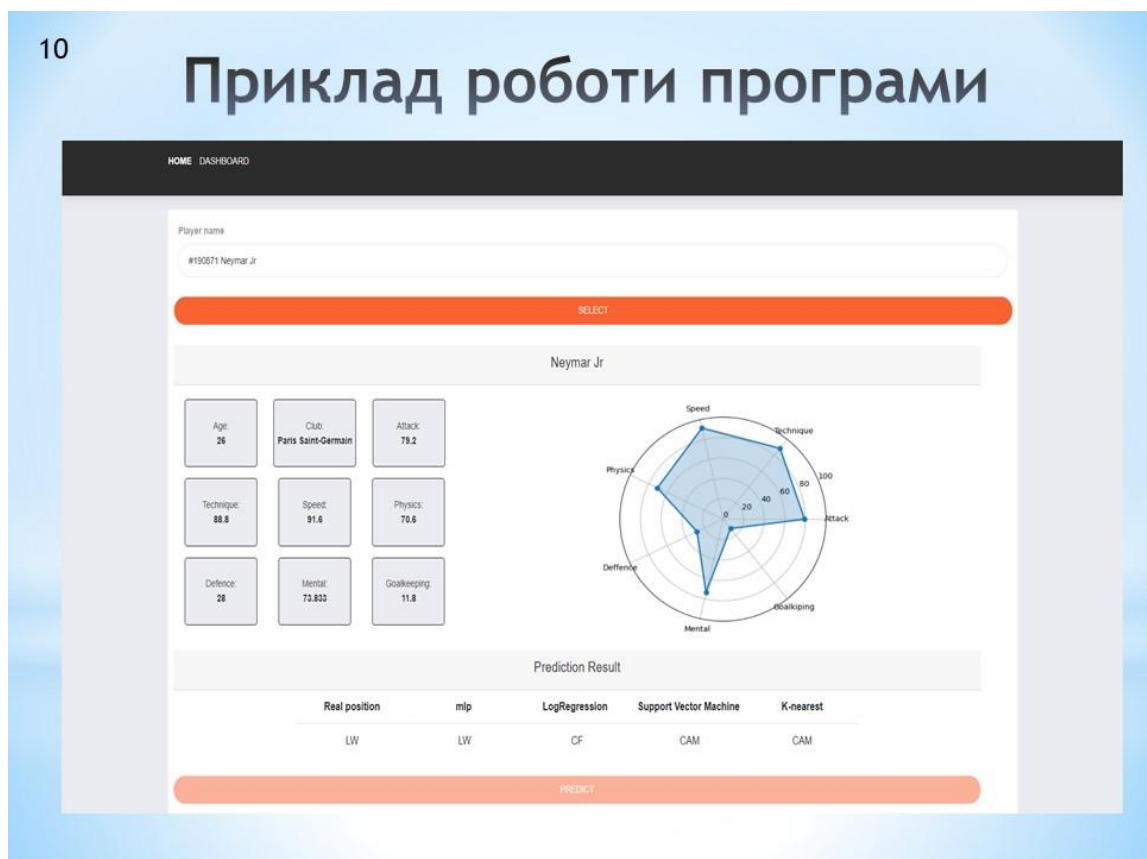


Рисунок В.10 – Приклад роботи програми

11

Показники модифікованого методу

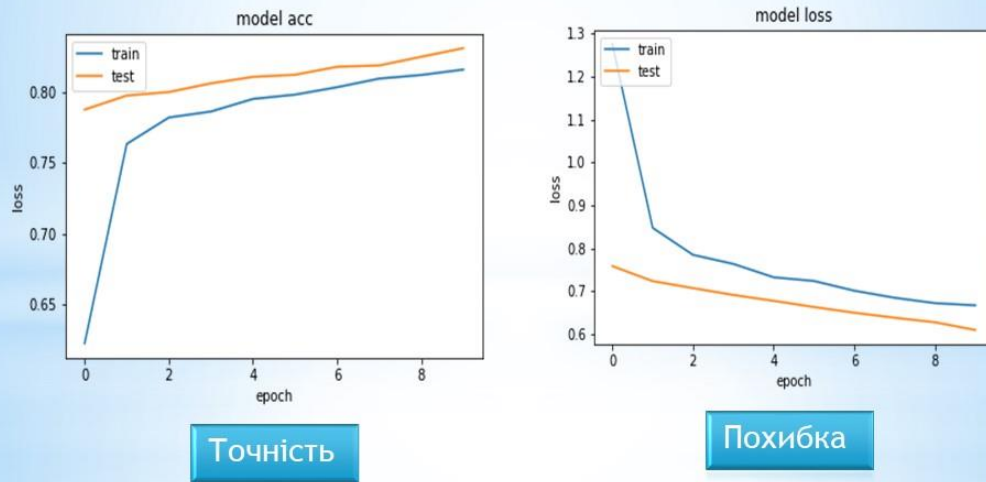


Рисунок В.11 – Показники модифікованого методу

12

Порівняння методів

Назва	Тренувальна	Тестова
Логістична регресія	0,78	0,76
Метод опорних векторів	0,71	0,61
К-найближчих сусідів	0,75	0,73
БШП	0,81	0,83

Рисунок В.12 – Порівняння методів



Рисунок В.13 – Висновок