

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПІДТРИМКИ ПРИЙНЯТТЯ
РІШЕНЬ ПРО ФУТБОЛЬНУ ТРАНСФЕРНУ АКТИВНІСТЬ
FOOTBALL TRANSFER DECISION SUPPORT SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-138
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Ліпихін С.О.

(прізвище та ініціали)

Керівник Льовкін В.М.

(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 ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
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
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Ліпіхіна Сергія Олександровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення підтримки прийняття рішень
про футбольну трансферну активність
Football Transfer Decision Support Software

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 27 ” квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблемної області. 2. Матеріали і методи.

3. Проєктування програмного забезпечення. 4. Розробка програми.

5. Експлуатація та тестування програми.

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
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання « 4 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження моделей для розв'язання проблеми	3 тиждень	Розділ 2
4	Проектування програми	4 тиждень	Розділ 3
5	Розробка програми	5 тиждень	Розділ 4
6	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
8	Захист роботи	8 тиждень	

Студент(ка)

(підпис)

Ліпихін С.О.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Льовкін В.М.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 184 с., 62 рис., 17 табл., 3 дод., 59 джерел.

ТРАНСФЕР ФУТБОЛІСТА, БАГАТОШАРОВИЙ ПЕРСЕПТРОН, МНОЖИННА РЕГРЕСІЯ, CIES, MVVM АРХІТЕКТУРА, FIFA.

Об'єкт дослідження – процес оцінювання трансферної вартості футболістів. Предмет дослідження – програмне забезпечення підтримки прийняття рішень про футбольну трансферну активність.

Мета роботи – створення програмного забезпечення підтримки прийняття рішень про футбольну трансферну активність щодо визначення ціни гравця на трансферному ринку для автоматизації цього процесу згідно з оцінкою його ігрових показників футбольними експертами.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мови програмування Python та C#, середовища розробки Spyder та MVS 19, парсинг даних, кореляційний аналіз, багатошарова нейронна мережа, навчання з учителем, бібліотеки Keras та NumPy, шаблон проєктування архітектури MVVM, технологія WPF, технологія WCF, персональний комп'ютер під керівництвом ОС Windows 10 з відеокартою GeForce GTX 1060.

Результати. Виконано об'єднання двох фундаментальних методів визначення трансферної вартості гравця – математичної моделі та моделі «мудрості натовпу».

Висновки. Розроблено програмний продукт для оцінювання гри футболістів та підрахунку на основі таких оцінок ринкової вартості гравців.

Галузь використання – футбольними функціонерами перед та в процесі трансферних перемовин, футбольними експертами для аналізу гри та звичайними вболівальниками на дозвіллі.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 184 pages, 62 figures, 17 tables, 3 appendixes, 59 sources.

FOOTBALL TRANSFER, MULTILAYER PERCEPTON, MULTIPLE REGRESSION, CIES, MVVM ARCHITECTURE, FIFA.

Object of study is the process of estimating the transfer value of football players.

Subject of the study is football transfer decision support software.

The purpose of the work is to develop decision support software on football transfer activity for determination the price of a player on the transfer market to automate this process according to the assessment of his performance by football experts.

Materials, methods and tools: object-oriented programming, programming languages Python and C#, development environments Spyder and MVS 19, data parsing, correlation analysis, multilayer neural network, supervised learning, libraries Keras and Numpy, MVVM architecture design template, WPF technology, WCF technology, personal computer with GeForce GTX 1060 under the control of the Microsoft Windows 10 operating system.

Results. The possibility of combining two fundamental methods of determining the transfer value of a player – mathematical model and the "wisdom of crowd" is implemented.

Conclusions. A software system has been developed to rate the game characteristics of football players and evaluate on the basis of such estimates of the market value of players.

The field of use is transfer negotiations between clubs' representatives and intermediaries, football analytics by experts and leisure of fans.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	9
Вступ	10
1 Аналіз предметної області	12
1.1 Трансферна вартість футболістів	12
1.1.1 Основні поняття та терміни. Процес укладання трансферу гравця	12
1.1.2 Основні ознаки для оцінки трансферу гравця	14
1.1.3 Масштаби футбольного трансферного ринку. Кількісний та якісний аналіз трансферного ринку	15
1.2 Огляд існуючих методів аналізу футбольного трансферного ринку та відповідних програмних засобів	21
1.2.1 Футбольна обсерваторія CIES	22
1.2.2 Сервіс Transfermarkt	25
1.3 Порівняльна характеристика існуючих та розроблюваного програмних продуктів	28
1.4 Постановка задачі	31
1.5 Висновки за розділом 1	32
2 Матеріали і методи	33
2.1 Визначення типу вирішуваної задачі. Вибір ознак	33
2.1.1 Вибір вхідних ознак моделі	33
2.1.2 Вихідна ознака. Тип моделі	38
2.2 Аналіз та обробка даних	39
2.3 Побудова математичної моделі	45
2.3.1 Нова концепція аналізу футбольної трансферної активності. Вербальна модель	45
2.3.2 Вирішення задачі регресії за допомогою багатоваріантної нейронної мережі прямого поширення	48
2.3.3 Структурний синтез моделі	48
2.4 Висновки за розділом 2	49

	7
3 Проєктування програмного забезпечення	51
3.1 Вибір інструментарію розробки програмного забезпечення	51
3.1.1 Аналіз технічного завдання та формування вимог до використовуваних програмних засобів	51
3.1.2 Вибір мови програмування	55
3.1.3 Вибір середовища розробки мовою Python.....	64
3.1.4 Особливості технології WPF у порівнянні з Windows Forms	66
3.2 Структурна схема програми та її опис	68
3.3 Проєктування структури бази даних	73
3.4 Висновки за розділом 3	76
4 Розробка програми	77
4.1 Алгоритми функціонування програмного забезпечення	77
4.2 Опис модулів реалізованої програми	82
4.2.1 Опис серверного додатку	82
4.2.1.1 Клас DataBase.....	83
4.2.1.2 Клас DataBaseLoad.....	84
4.2.1.3 Перелік DataActionResult.....	86
4.2.1.4 Клас NeuralNetwork	87
4.2.1.5 Клас PlayerService	89
4.2.2 Опис ViewModel компоненти клієнтського додатку.....	91
4.3 Висновки за розділом 4	93
5 Експлуатація та тестування програми.....	94
5.1 Призначення програми.....	94
5.2 Умови виконання програми	95
5.2.1 Файлова структура.....	95
5.2.2 Наявність зовнішніх залежностей	98
5.2.3 Мінімальні апаратні вимоги.....	100
5.3 Виконання програми	101
5.4 Тестування програми	109
5.4.1 Валідація параметрів	109

	8
5.4.2 Чеклист функціонального тестування	112
5.4.3 Результати навчання запропонованої моделі	114
5.4.4 Фінальна конфігурація моделі	117
5.4.5 Порівняння моделі з аналогічними рішеннями	118
5.5 Висновки за розділом 5	119
Висновок	120
Перелік джерел посилання	121
Додаток А Технічне завдання	127
Додаток Б Текст програми	138
Додаток В Слайди презентації	177

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IDE	– Integrated Development Environment;
MSE	– Mean Squared Error;
MVVM	– Model View ViewModel;
ReLU	– Rectified Linear Unit;
WCF	– Windows Communication Foundation;
WPF	– Windows Presentation Foundation;
БД	– база даних;
БНМ	– багатошарова нейронна мережа;
НМ	– нейронна мережа;
ПЗ	– програмний засіб;
ПП	– програмний продукт.

ВСТУП

Футбольний клуб – це в першу чергу бізнес-проект, а вже потім – спортивна команда, здатна проносити успіх на футбольному полі, вигравши титули. Таке твердження намертво вкоренилося у думках власників та менеджерів усіх провідних європейських футбольних клубів. Такий підхід превалює, і власники футбольних клубів спершу вирішують проблеми, пов’язані з бюджетом, а вже після цього – питання спортивної команди.

Рок за роком футбол, як породження бізнесу, розвивався, а разом з ним розвивалася його частина, пов’язана великою мірою з економічною стороною, – трансферні угоди. Вартість футбольних клубів, прав на трансляцію матчів, брендів, за останні роки досягла величезних масштабів, а відповідно і ціни на гравців теж. Якщо раніше, декілька десятиліть тому, перехід футболіста з одного клубу в інший за 5 мільйонів євро – це була варта уваги подія, то зараз подібна угода може не привернути уваги взагалі. Значущий трансфер вимірюється десятками, а іноді сотнями мільйонів євро.

Не дивно, що точне визначення ціни того чи іншого футболіста стало турбувати футбольну публіку. Під точною ціною розуміється така сума грошей, яку футболіст заслуговує. Заслуговує багатьма факторами, наприклад, своєю популярністю, однак на першому місці – своєю грою. Саме процес конвертації ігрових здібностей футболіста у свою економічну вартість вартий вивчення.

Під терміном вартість футболіста слід розуміти доволі складну концепцію: вартість – це ціна за розрив гравцем старого контракту, з колишнім клубом, та підписання нового, з новим клубом. Такий процес можливий тільки у відповідному правовому полі, а головне за певних економічних законів – законів трансферного ринку.

Процес оцінювання трансферної вартості футболістів – це об’єкт дослідження. Дослідження такого процесу могло б призвести до його формалізації, переведення до математичної моделі. У такому разі

оптимізувати процес ціноутворення можна б було відомими методами, обрахувавши найбільш точну ціну гравця.

Хоч подібні спроби, доволі успішно, були реалізовані, найбільш популярним засобом обчислення трансферної вартості гравця залишається нематематичний метод, що ґрунтується на аналізі думок експертів. Необхідно дослідити, чому це так, та як це можна використати при створенні власної моделі визначення ціни футболіста.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В даному розділі розглянуто та проаналізовано стан тематики на даний момент, з теоретичної та практичної точок зору.

Розглянуто основні поняття футбольного трансферного ринку та процес укладання трансферів; проаналізовано останні тенденції зміни ринку; розглянуто існуючі програмні засоби (ПЗ) вирішення проблеми ціноутворення на футболістів та деякі аналогічні програми для аналізу цього процесу. В решті-решт було визначено суть задачі, яка буде вирішуватися в ході виконання дипломної роботи.

1.1 Трансферна вартість футболістів

1.1.1 Основні поняття та терміни. Процес укладання трансферу гравця

Футбольна трансферна активність – це діяльність визначених акторів на футбольному трансферному ринку.

Ринок, у широкому сенсі, являє собою сукупність відносин (актів) купівлі-продажу товарів і послуг [1]. Такі відносини купівлі-продажу, у розглянутій спортивній сфері, прийнято називати трансферами. Цей термін підкреслює переміщення «товару» від одного власника до іншого. А таким товаром в свою чергу і є футболісти.

Контрагентами футбольного трансферного ринку є особи, що укладають договір – оформлюють трансфер. Такими особами є представники футбольних клубів, або з селекційного відділу, або самі спортивні директори [2]. Також суб'єктом ринкових відношень є футбольні агенти, які представляють та захищають інтереси футболіста. Агенти мають також приватний інтерес, а інколи і взагалі бувають каталізатором проведення трансферів.

Об'єктом подібних ринкових відносин є футболісти, а точніше їх контракти. Контракт – це форма трудового договору, який укладається між футбольним клубом, у особі його представників, та футболістом, як правило у особі його агента. Трансфер – це успішний розрив контракту з одним футбольним клубом та підписання футболістом контракту з іншим клубом, за умови повної узгодженості сторін.

Процес укладання трансферу гравця схематично зображено на рисунку 1.1.

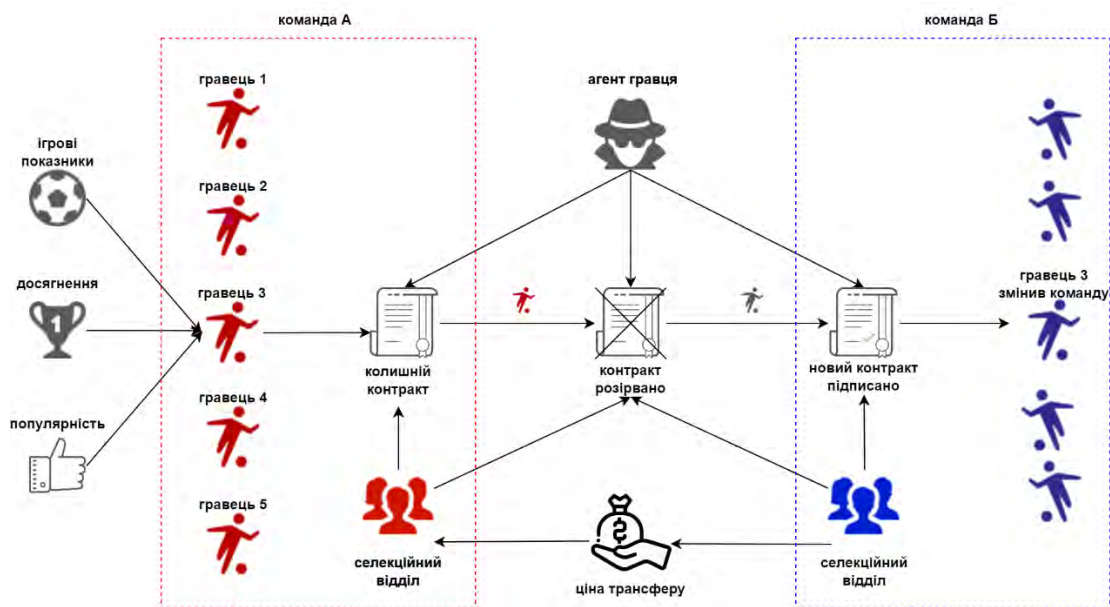


Рисунок 1.1 – Укладання трансферу гравця між двома командами

Загалом, повну правову оцінку проведення трансферу укладено та описано у «Регламенті по статусу і трансферу гравців» [3]. Саме цей документ не тільки регулює, а й закріплює структуру футбольного трансферного ринку. Наприклад, в ньому докладно пояснені поняття статус гравця, трансферне вікно, можливі види трансферів та інше.

Однак в цій роботі зосереджено увагу не стільки на правовому боці питання, скільки на економічному та ігровому.

Резюмуючи простими словами, трансферний футбольний ринок це абстрактне поле, головними гравцями якого є футбольні клуби. Їх ціллю є

задовільнення власного попиту – як вигідне придбання, так і продаж футболістів, які є центральним товаром. Оформлення актів придбання та продажу називаються оформленням трансферів футболістів, які у ринковій економіці традиційно мають під собою грошовий еквівалент.

1.1.2 Основні ознаки для оцінки трансферу гравця

Вигідність оформлення трансферу визначається за допомогою двох ознак: економічної та неекономічної (ігрової). Ігрові ознаки оцінюють кількість корисної дії, яку принесе гравець безпосередньо на футбольному полі. Такий фактор визначається як правило менеджером команди – в нашому регіоні замість терміну «менеджер» вживають «тренер». Подібна оцінка може бути отримана як до оформлення трансферу, так і після, коли футболіст проявить себе. Ігровий фактор часто виражається статистично, та наочно впливає у досягнення команди разом з цим футболістом, наприклад у перемогу в змаганні. Тим не менш, можна сказати напевно, що передбачається відносно збереження навичок футболіста при зміні його місця прописки.

Що ж до економічної оцінки здійсненого трансферу, то вона може мати кількісну характеристику, тобто мати грошову вартість. На цьому етапі з'являється дуже важливе зауваження – різниця між трансферною вартістю та ринковою вартістю гравця. Ринкова вартість – це оцінка трансферних зборів, які, швидше за все, будуть заплачені за футболіста [4]. Або ще точніше, ринкова вартість футболіста – це оціночна кількість грошей, які футбольний клуб готовий заплатити за те, щоб спортсмен підписав контракт, незважаючи на дійсну угоду [5]. Напроти ж, трансферна вартість – це фактична кількість грошей, які вже заплатили за гравця. Хоча існують концептуальні відмінності між ринковою та трансферною вартістю, вони є зіставленими [6].

Інтуїтивно зрозуміло, що існує певна кореляція між економічними та неекономічними показниками. Тобто певні ознаки, які характерні товару – футболісту, якимось чином впливають на ціноутворення і в кінцевому рахунку

впливають у вартості трансферу. Така залежність виявляється у процесі розглядання футбольного трансферного ринку – об’єкту дослідження; а сама така залежність ціни гравця від його футбольної активності є предметом дослідження. Відповідь на питання «яким чином ігрові показники впливають на економічні» є ключовою ціллю цієї та багатьох інших аналогічних робіт. Це передмова постановки задачі. Сама ж постановка задачі ще містить умову автоматизації процесу визначення ціни трансферу гравця.

1.1.3 Масштаби футбольного трансферного ринку. Кількісний та якісний аналіз трансферного ринку

Великих наслідків аналіз процесу ціноутворення набуває у футбольній індустрії – саме тут гострим стає питання найточнішого визначення реальної ціни на футболіста, згідно з його характеристиками. Чим точніше ця ціна буде визначена, тим більше у трансферних менеджерів та агентів буде аргументів на переговорах.

Звернемося до реальних прикладів на футбольному трансферному ринку для підтвердження актуальності розробки через аналіз обсягів грошового обороту ринку та його масштабів охоплення. Це дозволить виставити певні орієнтири, на які можна спиратися та до яких треба пристосуватися у процесі оперування грошовою масою.

Динаміку зміни обсягів трансферних витрат з часом (у фунтах стерлінгів) та графік зміни кількості трансферів з часом, на основі даних про трансфери гравців команд головних європейських чемпіонатів, зібраних з 1992 року по 2020 рік [7], зображено на рисунку 1.2.

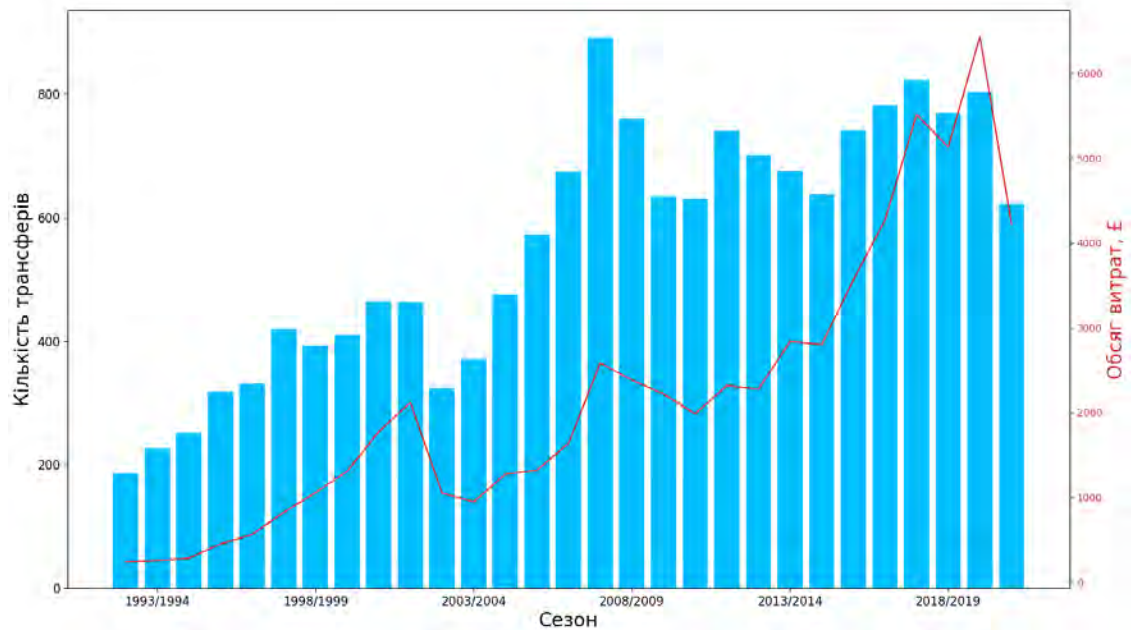


Рисунок 1.2 – Графік залежності кількості оформлених трансферів та обсягу трансферних витрат від часу

Аналізуючи отримані графіки, можна зробити висновок, що загальний показник обсягів трансферних витрат, майже за тридцять років, підвищився приблизно в шість разів. Хоч в останні роки такий показник і вийшов на плато, можна однозначно констатувати, що ринок росте суттєвими темпами. Це пояснюється неабиякими впливаннями в цю сферу бізнесу [8].

Саме так, футбол – це бізнес. Купівля та продаж гравців – не тільки один з головних інструментів досягнення спортивних успіхів у футболі, а ще й один з головних секторів бізнес політики клубу.

Для доведення першого судження звернемося до наступної статистичної викладки. Обсяги витрат на трансфери клубів в рамках окремо взятої ліги зображено на рисунку 1.3.

Клубні трансферні витрати

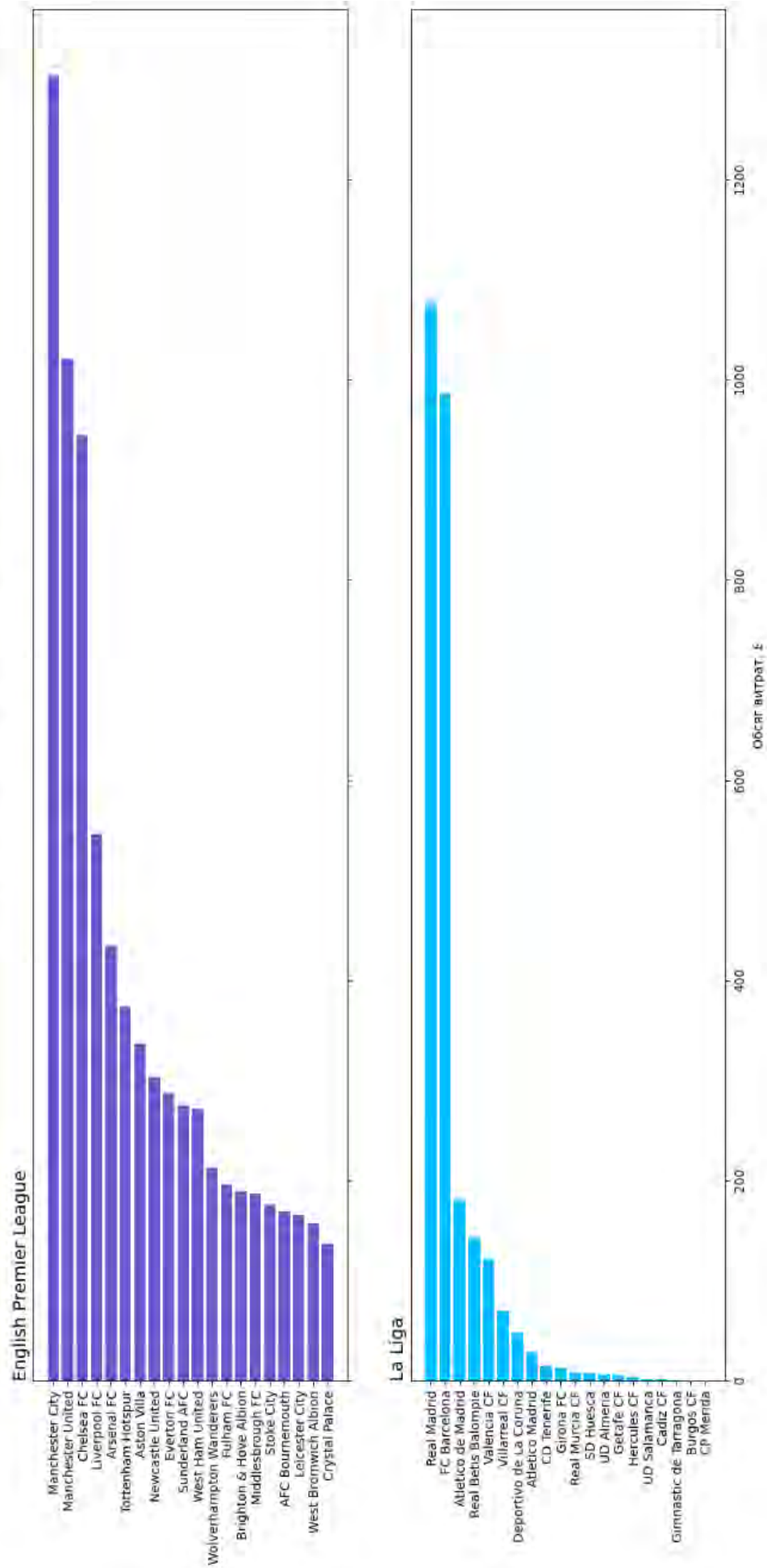


Рисунок 1.3 – Клубні трансферні чисті витрати для команд англійської Прем'єр ліги та іспанської Ла-Ліги

Показники трансферних витрат, зображених на рисунку 1.3, отримані шляхом віднімання суми затраченої на трансфери гравців від суми отриманої в результаті переходів. Дані взяті у період з 1992 року по 2020 рік. З рисунку можна побачити, що більше на трансфери витрачають команди, що займають високі позицій у турнірній таблиці. Переможців відповідних турнірів за роками зображено на рисунку 1.4.

English Premier League			La Liga		
2020/2021		Manchester City	2020/2021		Atletico de Madrid
2019/2020		Liverpool	2019/2020		Real Madrid
2018/2019		Manchester City	2018/2019		Barcelona
2017/2018		Manchester City	2017/2018		Barcelona
2016/2017		Chelsea	2016/2017		Real Madrid
2015/2016		Leicester City	2015/2016		Barcelona
2014/2015		Chelsea	2014/2015		Barcelona
2013/2014		Manchester City	2013/2014		Atletico de Madrid
2012/2013		Manchester United	2012/2013		Barcelona
2011/2012		Manchester City	2011/2012		Real Madrid
2010/2011		Manchester United	2010/2011		Barcelona
2009/2010		Chelsea	2009/2010		Barcelona
2008/2009		Manchester United	2008/2009		Barcelona
2007/2008		Manchester United	2007/2008		Real Madrid
2006/2007		Manchester United	2006/2007		Real Madrid
2005/2006		Chelsea	2005/2006		Barcelona

Рисунок 1.4 – Переможці англійської та іспанської футбольної ліги за останні п'ятнадцять років

З рисунку 1.3 та рисунку 1.4 видно, що між першими двома командами у іспанській Ла-Лізі та першими трьом командами у англійській лізі, у

порівнянні з усіма іншими клубами, утворилась різниця як у показниках трансферних затрат (рис. 1.3), так і у показниках спортивних досягнень (рис. 1.4).

Це доводить, що чим відповідальніше клуб підходить до комплектування своєї команди гравцями, саме через їх придбання за реальні гроші, тим вищі позиції у змаганнях така команда посідає. Цей факт спонукає клуби проводити якомога ефективнішу трансферну політику.

Співвідношення трансферних витрат і доходів клубів європейської ліги «великої п'ятірки» за роками 2013/14 – 2017/18 зображено на рисунку 1.5.

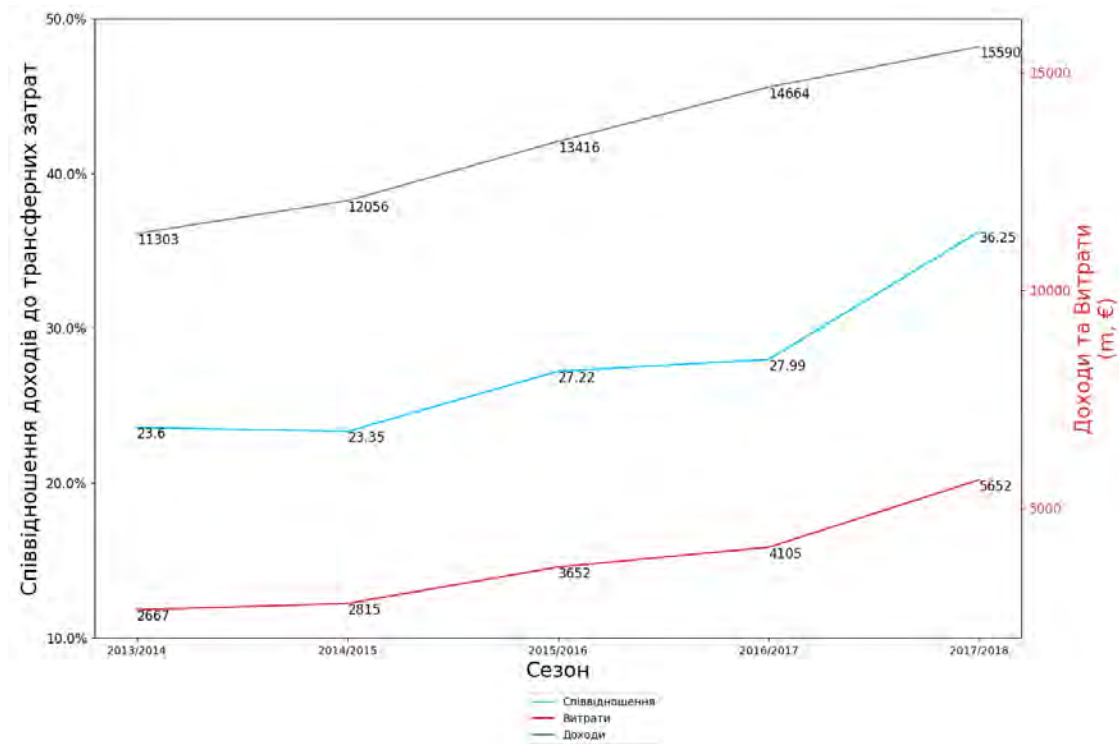


Рисунок 1.5 – Співвідношення трансферних витрат і доходів клубів європейських ліг «великої п'ятірки»

У п'яти найкращих європейських лігах (Бундесліга, англійська Прем'єр-ліга, Ла Ліга, Ліга 1, Серія А) як доходи, так і трансферні витрати стабільно зростали у сезонах 2013/14 – 2017/18. Витрати на трансфери зросли значно більше ніж сукупний річний дохід, і, як наслідок, відсоток доходу, витраченого на трансфери, збільшився більш ніж на 50%. Ця тенденція

свідчить про те, що деякі аспекти футбольного трансферного ринку змінилися, що змушує клуби виділяти більше доходів на трансферні витрати, ніж раніше [9].

Якщо раніше селекція футбольних клубів була направлена на виховання власних, доморощених гравців, то тепер головною її задачею є придбання та продаж футболістів. Тобто за останні роки відбулася трансферна революція, пов'язана з приходом у футбол колосальних фінансових засобів. Її головний маніфест – це переважання фінансового засобу досягнення результатів над спортивним. Це найяскравішим чином підкреслю масштаб важливості проведення ефективної трансферної політики.

Проаналізуємо суму, яку витрачають команди для придбання одного окремо взятого гравця. На рисунку 1.6 зображено зміну показника середньої та максимальної трансферної вартості футболіста з сезоном.

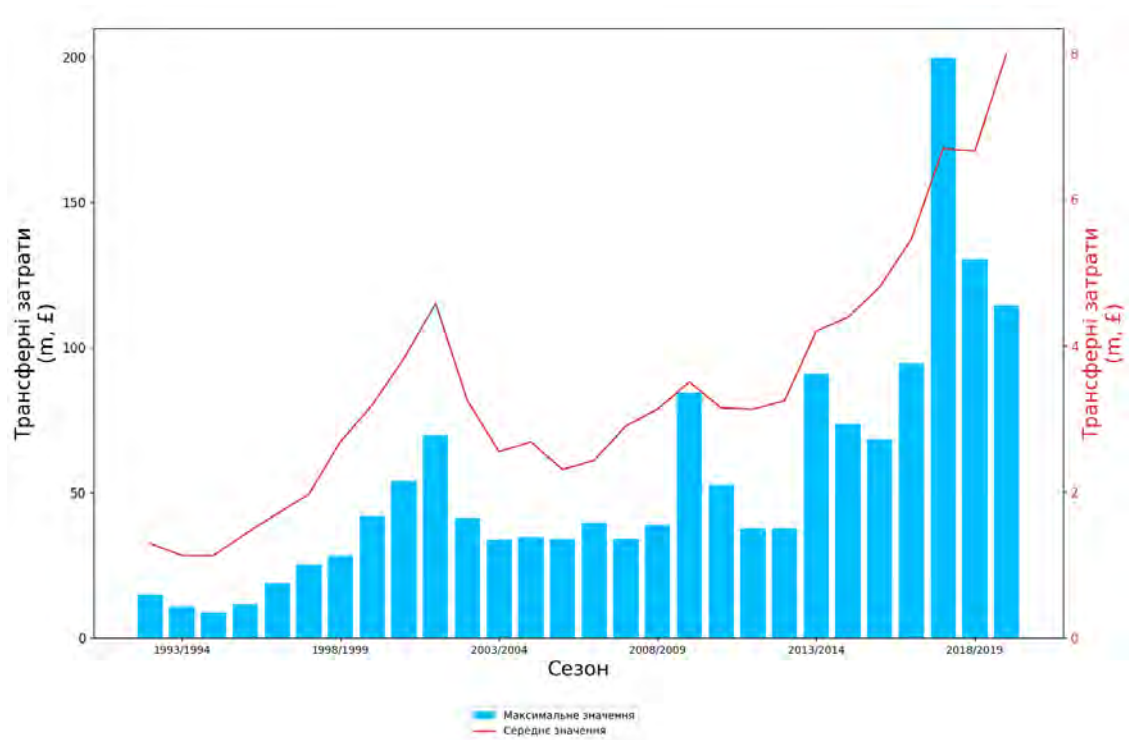


Рисунок 1.6 – Зміна середнього та максимального значень трансферної вартості футболіста для кожного сезону

За 25 років показник середнього значення оформленого трансферу збільшився майже у 5 разів, від 1.7 млн. фунтів стерлінгів до 8 млн. фунтів стерлінгів за футболіста; максимальна сума, яку клуб сплатив за гравця раніше не переважала і 40 мільйонів, а тільки за останні три роки максимальна сума трансферу склала більше 100 мільйонів; у 2017 році було поставлено рекорд – футболіст Неймар перейшов до клубу ПСЖ за 222 млн. євро [10].

Окрім швидкого досягнення командою спортивного результату, проведення здорової трансферної політики важливо для функціонування бізнес моделі клубу в цілому.

Світова фінансова криза у 2020 році [11] та доволі очікувана криза футбольного трансферного ринку на початку 2000-х років [12] довели, що деякі клуби не мають необхідного «запасу міцності». Обидві ці події можна побачити на рисунку 1.2 – обсяг трансферних витрат в ці роки значно зменшився. Кризи визивають як проблеми фінансові, так і спортивні [13], а такі коливання можуть взагалі призвести до знищення всієї справи – починаючи з перепродажу клубу власником, закінчуючи повним його банкрутством. В таких провалах критичну роль відіграє провальна трансферна політика – навіть найкращі клуби терплять суттєві невдачі через некомпетентність селекційного відділу чи спортивного директора [14]. Дивлячись на це, клуби все більше і більше приділяють увагу ефективності прийняття рішень, пов'язаних з трансферною активністю, та відповідним ПЗ.

1.2 Огляд існуючих методів аналізу футбольного трансферного ринку та відповідних програмних засобів

На сьогоднішній день для оцінки гри футболістів, існує багато можливостей, починаючи від скаутингу, коли зацікавлений у аналізі експерт приїжджає на матч та наживо передивляється гру потрібного футболіста, та закінчуючи необмеженим доступом до онлайн статистики гравця. Однак, усі ці параметри виступів футболіста складно аналізувати окремо, через що при

проведені трансферів відповідні працівники клубу звертаються до сервісів, що надають вже проаналізовану та виведену вартість на цих футболістів. Така вартість є ринковою, як вже було зазначено в пункті 1.1.2, тобто подібні дані носять лише орієнтовний характер. На цьому етапі принципово визначити, за яким принципом розробники таких сервісів та програм формують ринкову ціну та на які дані спираються.

1.2.1 Футбольна обсерваторія CIES

Футбольна обсерваторія CIES [15] – це дослідницька група, що функціонує в рамках Міжнародного центру спортивних досліджень (CIES), незалежного навчального центру.

Футбольна обсерваторія CIES спеціалізується на статистичному аналізі футболу. Складається група з чотирьох осіб: співробітника, відповідального за збір даних (Андреа Пессіна) та трьох дослідників (доктора Рафаеле Полі, Роджера Бессона та Лоїка Равенеля) [16]. Саме авторству останніх належать наукові статті та публікації, згідно до яких можна встановити методику роботи CIES.

CIES систематично створює математичні моделі, які дозволяють аналізувати футбольну сферу під різними кутами, однак так чи інакше усі дослідження зводяться до оцінки футболістів та їх активності. Наприклад, досліджується імовірність того, що гравець залишить поточний клуб та перейде до довільного іншого [17], аналізується трансферний ринок по окремих роках чи періодах, з урахуванням останніх тенденцій у суміжних сферах, наприклад, проведено аналіз змін викликаних пандемією 2020 року [11], [19], проводяться демографічні дослідження на тлі складів європейських команд [19]. Так чи інакше, центральною темою подібних досліджень, які проводяться обсерваторією CIES, є вивчення трансферного ринку, а конкретно процесу формування ціни на гравців.

Суми трансферних виплат не є ірраціональними, вони фіксуються учасниками ринкових відношень на основі критеріїв, які здебільшого об'єктивні та, таким чином, можуть бути переведені в статистичну модель [20]. CIES встановлюють, що трансферні вартості можуть бути розглянуті як цільові ознаки, значення яких залежать від певних, об'єктивних умов.

У своїй моделі CIES застосовують показник трансферної вартості, віддаючи йому перевагу над показником ринкової вартості (пункт 1.1.2). Тобто при зборі даних увагу зосереджено на вже проведених трансферних угодах, в яких чітко зафіксовано ціну трансферу. Саме цей показник використовується як залежна змінна моделі.

Вирішальна складність моделі полягає у відборі інформативних незалежних ознак. Такі ознаки можна умовно поділити на три групи:

- ознаки, що відносяться до клубів. Вони в свою чергу теж діляться на дві категорії. Ознаки першої категорії описують рівень команди зі спортивної точки зору, в яких футболісти грали до переходу. Такі показники відбивають досягнення команди в усіх змаганнях, в яких вона приймала участь. Ознаки другої категорії описують економічні можливості команди, наприклад який в команди бюджет та скільки в неї було інвестовано засобів;

- ознаки, що відносяться до гравців. Це тривалість дійсного контракту, вік гравця, міжнародний статус, кар'єрний прогрес та статистика виступів (кількість проведених матчів, забитих голів, точних передач, тощо).

- «контекстуальна» ознака – сезон, упродовж якого здійснюється перехід.

Така класифікація вхідних ознак зображена на рисунку 1.7.

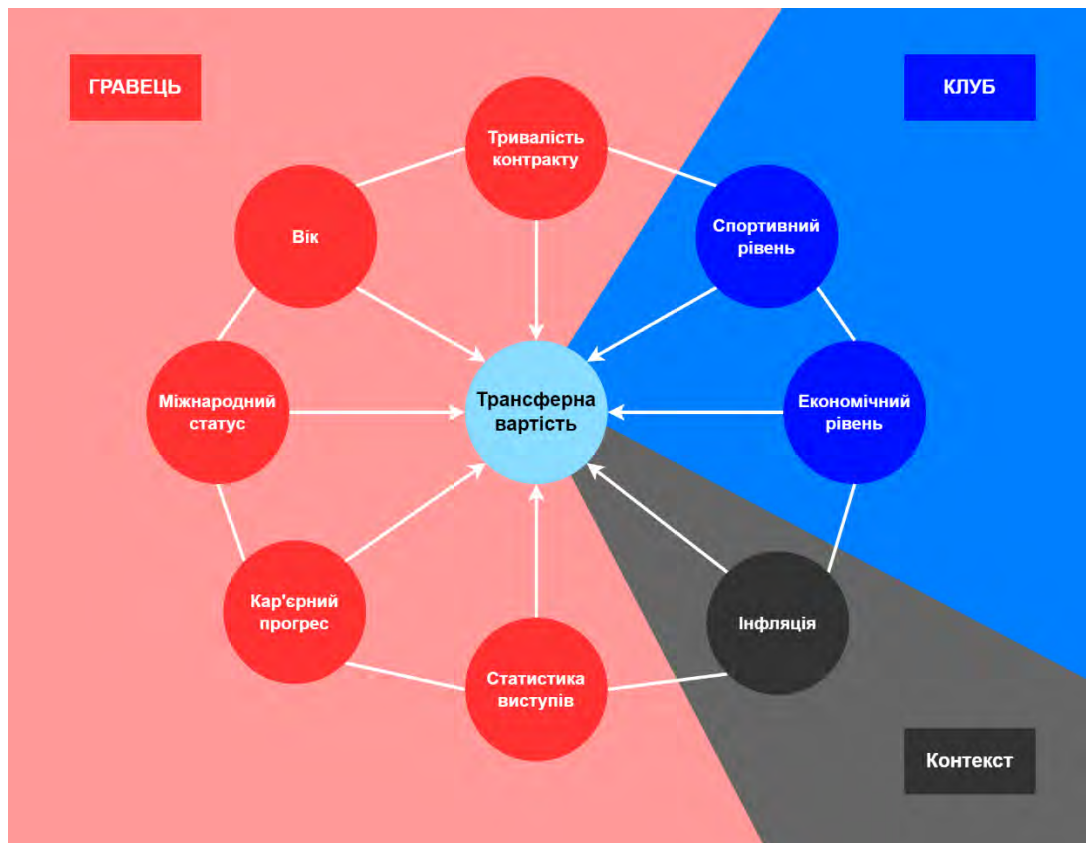


Рисунок 1.7 – Класифікація незалежних ознак у математичній моделі від CIES для визначення трансферної вартості футболістів

Кожна з ознак робить значущий внесок у визначенні цінності гравця. Високий ступінь кореляції між розрахованими сумами за допомогою математичної моделі та реально виплаченими сумами вказує на те, що застосовані ознаки добре відображують те, як учасники ринкових відношень домовляються про суму трансферної угоди [20].

Тим не менш, попри широку вибірку представлених вхідних ознак, деякі обставини формування трансферної вартості або не враховані, або не можуть бути виражені в формалізованому вигляді. Наприклад, розбіжності між тренером та гравцем, чи іншими членами колективу, підвищена травматичність чи інші недоліки фізичної форми гравця, висока конкуренція на позиції в клубі.

Для перевірки побудованої статистичної моделі було оцінено трансферні вартості гравців п'яти провідних європейських чемпіонатів на основі вибірки з 1790 оплачених трансферів, здійснених у період з 2012 року

по 2020 рік. Прогнозовані трансферні вартості було порівняно з реальними, укладеними трансферами. Результати роботи моделі представлено на рисунку 1.8.

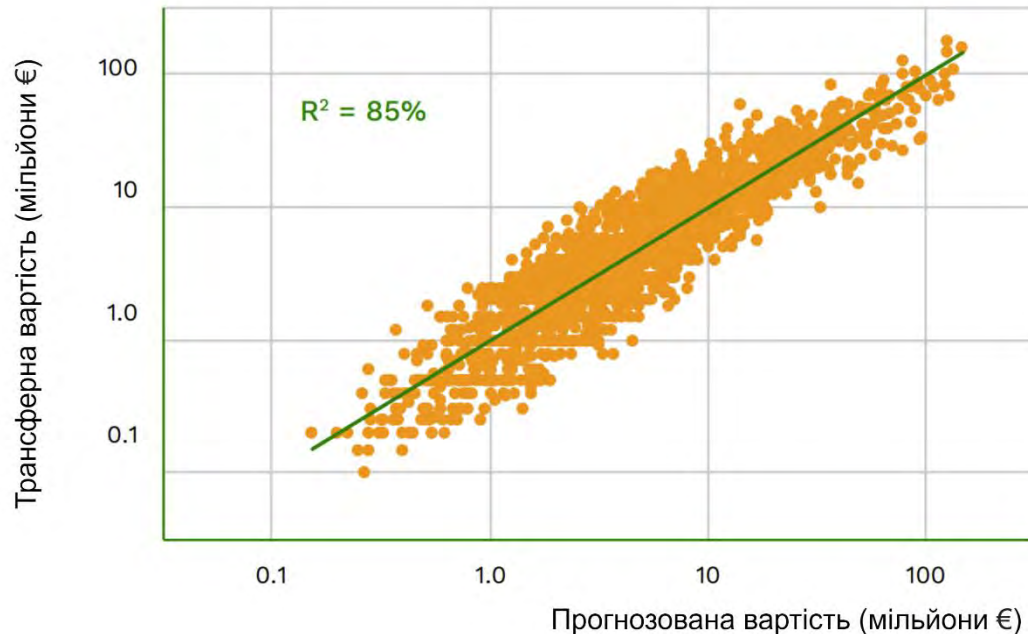


Рисунок 1.8 – Порівняння прогнозованих трансферних вартостей футболістів з реальними [20]

Кореляція між прогнозованими та сплаченими трансферами дорівнює 85 %. Це вказує на те, що незалежні ознаки пояснюють більше чотирьох п'ятих оформлених переходів гравців з однієї команди в іншу.

1.2.2 Сервіс Transfermarkt

Transfermarkt [21] – це компанія, яка спеціалізується на збиранні футбольної статистики та оцінці цін на футболістів. Усі дані публікуються на однойменному сайті, який став неофіційним оцінювачем гравців з усього світу.

Transfermarkt був створений в 2000 році як результат роботи Маттіаса Зайделя, німецького комп'ютерного інженера та за сумісництвом вболівальника футбольного клубу «Вердер» [22]. Протягом наступних років

сайт перетворився на професійну організацію, результати роботи якої затребувані серед кращих футбольних функціонерів світу у питанні оцінки вартості футболістів. Така суттєва метаморфоза змушує дослідити їх застосовуваний метод перетворення великого масиву розрізнених статистичних даних у впорядковану інформаційну одиницю – вартість гравця на трансферному ринку.

Головна особливість методики Transfermarkt, у процесі формування ціни на футболіста, – це відсутність автоматизованого алгоритму [22].

Transfermarkt оцінює очікувану трансферну вартість футболіста, яку прийнято називати ринковою (пункт 1.1.2). Ринкову вартість ні в якому разі не слід ототожнювати з трансферною вартістю – фактично виплаченою сумою грошей за гравця. Визначення ринкової вартості відбувається через детальне обговорення та за активної участі спільноти Transfermarkt [23]. Такий метод отримав назву «мудрість натовпу».

Коли окремі судження про якийсь факт усереднюються, спільна думка, як правило, точніша, ніж більшість індивідуальних оцінок [24]. Саме на цьому принципі базується метод «мудрість натовпу», використовуваний командою Transfermarkt. Аналіз передбачає краудсорсингове визначення значень певних ознак, а потім переходить до емпіричного аналізу отриманих оцінок, для визначення загальноприйнятих значень від Transfermarkt [25].

Використання мудрості натовпу у проєкті Transfermarkt має свої особливості. Фактори, які рекомендується брати до уваги експертам, при оцінюванні гравця можна умовно поділити на три групи:

- найбільш важливі фактори: вік, спортивні досягнення в клубі та національній збірній, значимість ліги, як спортивна, так і фінансова, потенціал та перспективи на майбутнє, досвід, кількість та репутація зацікавлених сторін, схильність до травм, загальний розвиток трансферних платежів;

- індивідуальні фактори трансферу: перехід через опцію викупу, прописана сума відступних, обмін гравцями, тривалість контракту, бонуси;

– ситуаційні фактори трансферу: «тиск», такий як конкуренція чи фінансовий тиск, воля/бажання/інтереси гравця, гравець оголошує страйк чи щось подібне, висока зарплата, клуб хоче/вимушений продати гравця.

Класифікацію факторів, на базі яких експерти спільноти Transfermarkt оцінюють трансферну вартість футболістів, зображено на рисунку 1.9.

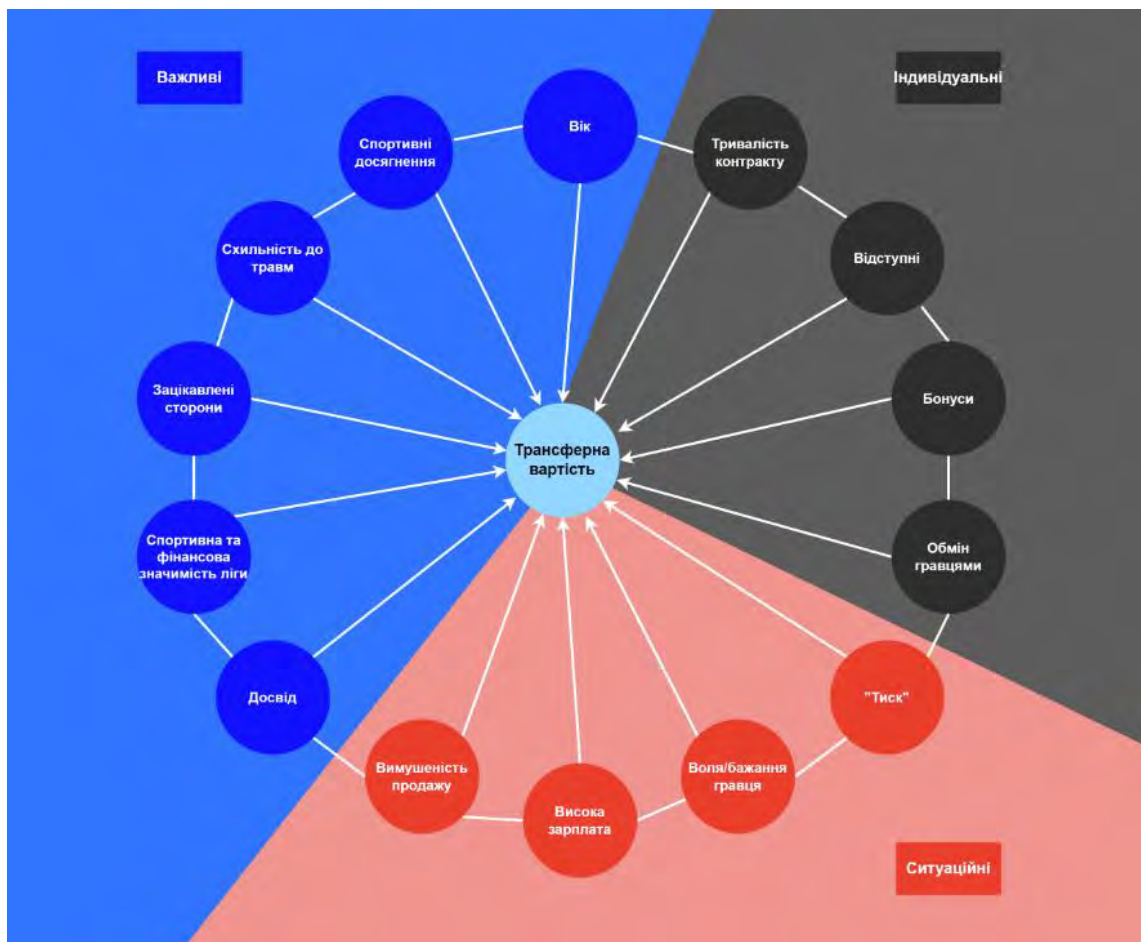


Рисунок 1.9 – Класифікація факторів, на базі яких експерти спільноти Transfermarkt оцінюють трансферну вартість футболістів

Оцінка подібних факторів пояснює попит на гравця на трансферному ринку та впливає у вартості в грошовому еквіваленті. Таку оцінку може дати окремо взятий учасник спільноти Transfermarkt, базуючись на власних аргументах. Сукупність оцінок подібних експертів зі знань аналізує так званий «хрещений батько» [22]. На внутрішньому сленгу Transfermarkt так називають закриту невелику групу людей, які і визначають остаточні ціни на футболістів.

Через велику кількість гравців, їх приблизно 800 тисяч, кожна футбольна ліга містить свого власного «хрещеного батька» – така структура дозволяє, через делегацію прав підпорядкованим групам, опрацьовувати великі обсяги даних та оцінити кожного футболіста.

Реалізація принципу «мудрість натовпу» у проєкті Transfermarkt зображено на рисунку 1.10.

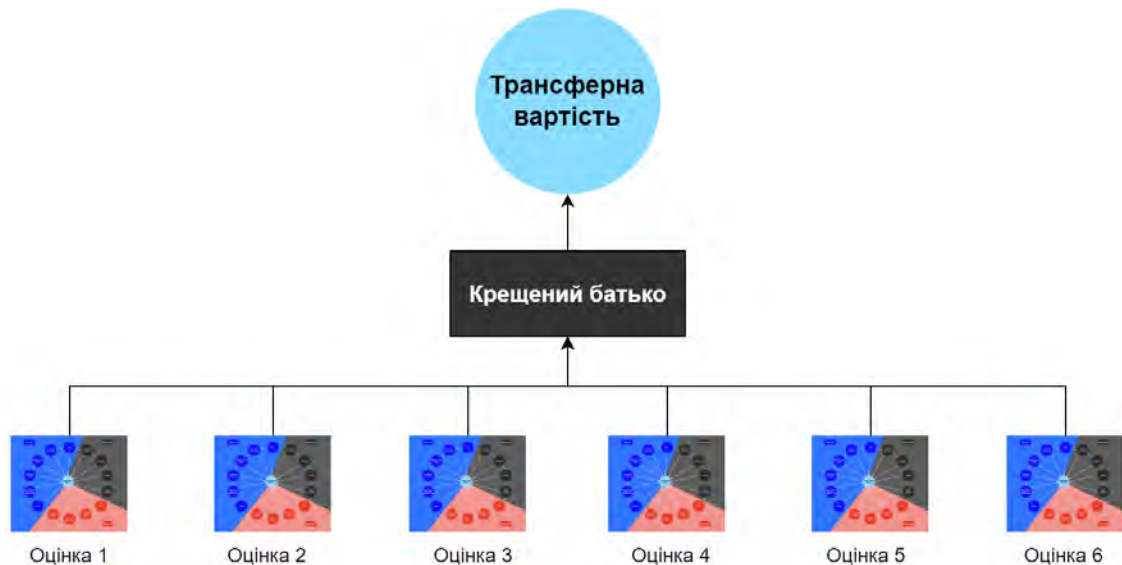


Рисунок 1.10 – Реалізація принципу «мудрість натовпу» у проєкті Transfermarkt – усереднення думок

Гравці в футбольних лігах вищого рівня переоцінюються чотири рази на рік: два основних оновлення перед трансферними періодами і два незначних між ними. Для кожної країни цей процес відбувається однаково, за деякими винятками [22].

1.3 Порівняльна характеристика існуючих та розроблюваного програмних продуктів

Методи аналізу трансферної активності футболістів, які використовують вищезазначені сервіси, було проаналізовано у попередньому пункті. Тепер є необхідність порівняти засоби реалізації відповідних методів

із програмним забезпеченням, яке розроблюватиметься у цій дипломній роботі.

У таблиці 1.1 наведено порівняльну характеристику існуючих програмних продуктів (ПП) та програми, яка буде розроблена в ході виконання роботи.

Таблиця 1.1 – Порівняння програмних продуктів

Критерій	CIES	Transfermarkt	Власне
1	2	3	4
Клієнт-серверна архітектура	+	+	+
Робота через інтернет	+	+	±
Команда розробників	4	>40	1
Локалізація	ENG, FRA	20	ENG
Подання даних	+	++	+
Функціональність			
Подання «сирих» даних	-	+	+
Оцінка вартості гравця	+	+	+
«Доскональність» оцінки	+	-	-
Новинний портал	±	+	-
Стороння функціональність	-	+	+
Робота з користувачами			
Користувачі різних типів	±	+	+
Реєстрація	-	+	+
Інтерактивність	±	+	++
Актуальність			
Для функціонерів	+	+	+
Для вболівальників	-	+	+
Дані			
Доступність	-	++	+

Продовження таблиці 1.1.

1	2	3	4
Наявність власного масиву даних	-	+	-
Частота оновлення	±	±	±
Інформативний обсяг	-	+	±
Обсяг по гравцю	-	++	+
Різноманітність	+	+	+
Формалізованість	+	+	+
Ринкова вартість?	-	+	+
Точність вартості	±	++	+
Метод			
Тип	математичний	мудрість натовпу	гібридний
Консультативність	+	+	+
Гнучкість	±	±	+

Порівнюючи з оглянутими аналогами, розроблюваний ПП буде мати наступні переваги:

- повний доступ до усіх можливих даних, пов’язаних з предметною областю, на відміну від CIES, де дані скриті від звичайного користувача;
- користувачі з різними правами – деякі мають право на редагування інформації, а деякі ні;
- реєстрація та авторизація для обліку користувачів;
- інтерактивність, зокрема прямий вплив користувача на підрахунок власного варіанту ринкової вартості визначеного футболіста;
- широке коло користувачів – кожний вболівальник може долучитися до користування програмним продуктом, не маючи при цьому професійних навичок.

Тим не менш, слабкими місцями ПП, створюваного у рамках цієї роботи, у порівнянні з існуючими аналогами, є:

- дані зібрано чи згенеровано за допомогою сторонніх ресурсів;
- оновлення даних відбувається спорадично;
- невеликий інформативний обсяг, зосередженість на ігрових аспектах.

Також розроблюваний ПП характеризується такими особливостями:

– консультативний характер для користувачів у питанні ціноутворення на футболіста. Вартість гравців формується не стільки автоматизовано, скільки спираючись на думки експертів зі знань;

– можливе застосування ПП для оцінки гравців, що виступають як на професійному рівні, так і на аматорському;

– використовуваний метод при оцінці вартості – поєднання двох проаналізованих методів від CIES та Transfermarkt;

– розробка ведеться однією людиною – автором роботи;

– локалізацію не передбачено – мова програми англійська.

Так чи інакше, розроблюваний програмний продукт покликаний не стільки замістити недоліки інших продуктів, скільки реалізувати власний метод прийняття рішень, щодо футбольної трансферної активності.

1.4 Постановка задачі

Згідно до проаналізованих аналогів можна впевнено казати, що існує певна функціональна залежність між двома типами оцінки футбольної активності гравця. Оцінки першого типу, як правило, об'єктивні та виражені у формалізованих показниках, будь-то забиті футболістом голи, зроблені ним точні передачі, тощо. Друга оцінка виражена у грошовому еквіваленті, як прийнято у ринкових відносинах, та є залежною за певним функціональним законом від ігрових показників гравця.

Таким чином, задача, яка вирішується в ході виконання дипломного проекту, у широкому сенсі визначається як задача прогнозування, тобто

передбачення значення цільової змінної, згідно до значень незалежних змінних. Говорячи у термінах предметної області, необхідно прогнозувати значення вартості футболіста на трансферному ринку, в залежності від його здібностей та ігрових показників. Для цього необхідно побудувати певну математичну, функціональну модель.

Також, аналізуючи аналогічні програмні застосунки, було зроблено висновок, що будь-який сучасний продукт надає користувачу доступ до зручних функціональних можливостей та інтерфейсних рішень. Тому, для побудованої математичної моделі необхідно створити певну функціональну та графічну оболонку, для взаємодії з моделлю користувача.

1.5 Висновки за розділом 1

У розділі 1 було досліджено предметну область: розглянуто поняття трансферний ринок та процес укладання договору при переході гравця з одного клубу в інший. Було досліджено динаміку зміни показника трансферної вартості футболіста за останні 25 років та доведено актуальність роботи. Також було порівняно програмні засоби, які автоматизують процес визначення ціни на гравця, – CIES та Transfermarkt.

Завдання дипломної роботи визначені наступні:

- аналіз предметної області;
- дослідження методів і моделей оцінювання трансферної вартості футболіста;
- проектування програмного забезпечення підтримки прийняття рішень про футбольну трансферну активність;
- розробка програмного забезпечення підтримки прийняття рішень про футбольну трансферну активність.

2 МАТЕРІАЛИ І МЕТОДИ

В даному розділі наведено опис методу вирішення поставленої задачі. Наведено поступову класифікацію типу вирішуваної задачі. Проведений вибір ознак, робота з даними, вибір та проєктування математичної моделі.

2.1 Визначення типу вирішуваної задачі. Вибір ознак

Типом досліджуваних об'єктів предметної області є футболісти. Об'єкт такого типу – це окремо взятий футболіст з певними значеннями вибраних ознак у визначений момент часу. Наприклад, гравець N у першій половині футбольного сезону 2021/2022 має показник фізичної витривалості p одиниць. Таку сукупність ознак, будь-то фізична витривалість чи сила удару, можна легко вимірити та визначити значення кожної з них. Інший же показник – ціна гравця на трансферному ринку – залежить від навиків футболіста та є таким, який складно виміряти. Саме для встановлення чіткої залежності вартості гравця від набору певних оцінок його гри, необхідно створити функцію, яка б апроксимувала таку залежність.

Розглянуту задачу можна віднести до класу статичних задач, у процесі вирішення якої здійснюється лише збільшення інформації про футболіста – визначення його вартості, що не призводить до зміни раніше вибраних даних [26]. Більш того, задача визначення для розпізнаваного екземпляра значення розрахункової ознаки, за допомогою моделі залежності, називається задачею апроксимації.

2.1.1 Вибір вхідних ознак моделі

Перш за все, потрібно розглянути незалежні ознаки, які, за умовою задачі, повинні відбивати рівень гри футболіста та, водночас, бути

розробленими у вигляді певних метрик. Подібні неекономічні ознаки були коротко описані у пункті 1.1.2.

Для того, щоб у подальшому сформувати навчаючу вибірку, на основі своїх власних ознак, необхідно було б провести цілу акцію, з опитуванням великої кількості експертів, тим паче на вибірці щонайменше з тисячі футболістів. Однак через те, що організувати подібне соціальне опитування в рамках цієї роботи неможливо та не обов'язково, було прийнято рішення звернутися до вже існуючих критеріїв.

До незалежних інформативних ознак, які будуть застосовані у моделі, висувається ряд наступних вимог:

- велика кількість. Ознак повинно бути багато, тому що вони мають відбивати рівень футболіста у різних аспектах гри;
- суб'єктивність. Ознаки повинні бути оформлені як суб'єктивні оцінки. Якщо значення таких ознак не можливо виміряти у жодній існуючій фізичній одиниці виміру – тим краще;
- класифікація. Ознаки повинні бути категоріально згруповані, кожна група повинна нести відповідний сенс;
- вимірність. Ознаки повинні надаватися у певній метриці, кожна з яких повинна мати під собою прозору систему виміру. Шкала для виміру повинна бути або абсолютною, або, ще краще, інтервальною;
- зміна з часом. Ознаки повинні бути прив'язані до певного часового інтервалу.

Величезна кількість моделей, створених для прогнозування вартості гравця [4], [6], [27]-[29], використовує дані, отримані з сервісів Transfermarkt або Opta [17]. Застосовані ними метрики задовольняють усім переліченим вимогам, крім однієї – суб'єктивності. Тобто подібні ознаки як раз ті, від який в даній роботі було запропоновано принципово відмовитися.

Ще вартою уваги категорією є опитувальники, де експертами висуваються усі охочі. Гравців можуть аналізуватися за позиціями та під різними кутами – за встановленими до голосування критеріями. Така система

потенційного джерела ознак, та як наслідок початкових даних, є неповноцінною, тому що опитування проводяться ресурсами спорадично, а тому і побудувати модель на їх основі не є доцільним.

Напроти, ресурс WhoScored надає оцінку гравців за ігровий тур, яка зображена на рисунку 2.1, аналізуючи погляди експертів, а така оцінка оновлюється та вільно розповсюджується [30]. Однак недоліком є те, що такі оцінки не можуть бути застосовані як ознаки, тому що їх кількість замала для конструювання моделі.



Рисунок 2.1 – Оцінка гравців від WhoScored

Рішення проблеми підбору вхідних ознак було знайдено на основі набору даних комп'ютерної гри із серії FIFA. Насправді, пропоновані розробниками ознаки являють собою дещо середнє між реальною статистикою та суб'єктивною оцінкою людини. У мережі можна знайти негативні відгуки щодо невідповідності оцінок показників FIFA із реальними показниками, які демонструються гравці на футбольному полі [31, 32]. Це може зайвий раз свідчити про необ'єктивність таких даних, що і вимагається від ознак. Більш того, вибрані ознаки відповідають усім іншим вимогам: дані у наявності на

приблизно п'ятнадцять тисяч футболістів, ознаки згруповані за категоріями, числове представлення у інтервальній шкалі, оновлення кожний тур. Опис ознак FIFA наведено у таблиці 2.1.

Таблиця 2.1 – Ознаки використовувані FIFA [33]

№	Назва	Опис	Шкала
1	2	3	4
1	Загальний рейтинг	Середня оцінка навиків гравця	0-100
2	Потенціал	Можливий додатковий рейтинг	0-100
3	Міжнародна репутація	Показник гравця в залежності від його популярності	1-5
4	Майстерність	Відокремлює сильних гравців від великих	1-5
5	Слабка нога	Визначає потужність удару та контроль м'яча іншою ногою гравця, ніж бажана	1-5
6	Кроси	Визначає наскільки точно гравець виконує подачі, у грі та при штрафних ударах.	0-100
7	Завершення	Здатність гравця забивати голи	0-100
8	Гра головою	Точність під час використання голови для передачі чи удару	0-100
9	Короткі паси	Точність коротких передач	0-100
10	Удар з льоту	Здатність гравця бити по м'ячу на льоту	0-100
11	Дриблінг	Здатність гравця, контролюючи м'яч, провести його повз суперника	0-100
12	Підкручення м'яча	Здатність підкручувати м'яч	0-100
13	Штрафні удари	Точність вільних ударів	0-100
14	Дальні паси	Точність дальніх передач	0-100
15	Контроль м'яча	Здатність керувати м'ячом	0-100

Продовження таблиці 2.1

1	2	3	4
16	Прискорення	Швидкість при забігу на 30 метрів	0-100
17	Спринт	Швидкість при забігу на 100 метрів	0-100
18	Спритність	Швидкість руху з точки зору контролю м'яча	0-100
19	Реакція	Швидкість дії у відповідь на ситуації, що відбуваються навколо гравця	0-100
20	Баланс	Здатність залишатися стійким та займати вертикальне положення	0-100
21	Потужність удару	Сила ударів гравця	0-100
22	Стрибки	Ефективність у грі на другому поверсі	0-100
23	Витривалість	Здатність витримувати тривалі фізичні або емоційні напруження	0-100
24	Сила	Якість фізичної сили гравця	0-100
25	Дальні постріли	Точність ударів гравця з великих відстаней	0-100
26	Агресія	Рівень агресії гравця, коли він вступає у єдиноборство	0-100
27	Перехоплення	Здатність перехоплювати паси команди суперника	0-100
28	Позиціонування	Наскільки добре гравець займає позицію на полі	0-100
29	Бачення поля	Обізнаність гравця про позиційну гру	0-100
30	Пенальті	Влучність гравця при ударах з одинадцятиметрової позначки	0-100
31	Самовладання	Відчуття спокою та контролю над своїми емоціями при поразках	0-100
32	Гра по супернику	Здатність позначати гравців суперника	0-100
33	Відбір	Вміння відбирати м'яч стоячи	0-100
34	Підкат	Здатність виконувати підкат	0-100

Можна підсумувати, що наведені ознаки задовольняють усім висунутим вимогам. Обсяг даних, який надають розробники FIFA, є достатньо широким для налагодження функціональної залежності. На додачу, подібні ознаки легко осмислюються та можуть бути визначені для кожного гравця індивідуально.

2.1.2 Вихідна ознака. Тип моделі

Вихідна ознака не підлягає підбору, вона апріорно визначена темою роботи. Необхідно зазначити, що визначення, чи це трансферна вартість, чи це ринкова вартість гравця (п. 1.1.2), не є принциповим, через те, що результат роботи моделі буде мати лише консультативний характер. Той же Transfermarkt надає широку інформацію про переходи гравців та називає суми вартості трансферів. Однак для чистоти експерименту було прийнято рішення використати саме ринкову вартість як цільову ознаку.

Ринкова ціна гравця представлена у вигляді дійсного числа та подана у фунтах стерлінгів. Головне, що вартість гравця – це неперервне число, що може свідчити про те, що вирішувана задача є задачею оцінювання, або задачею регресії.

В заочну дискусію з таким рішенням вступає модель [34], де неперервну цільову ознаку було дискретизовано на сотню інтервалів, а задача із класу оцінювання перейшла до класу розпізнавання образів. Таке рішення хоч і не є оптимальним (що у подальшому буде доведено), однак є конкуруючим та вартим порівняння. Інші ж роботи [4], [6], [27]-[29], зокрема робота з використанням подібного набору ознак [35], що працює із подібними ознаками, теж вирішують проблему регресійного аналізу даних.

Таким чином, з проведеного аналізу вибраних ознак можна типізувати вирішувану інтелектуальну задачу, як регресійну. Стандартною практикою вирішення такої задачі є навчання з учителем. Воно передбачає на основі заданої навчальної вибірки $\langle x, y \rangle$, що описує залежність $f(w, x)$, знаходження

такої її структури f і таких значень параметрів w , що забезпечують оптимальне значення заданої функції якості розпізнавальної моделі $Q(f(w, x))$ [26].

Як критерій оптимальності Q традиційно використовують функцію помилки E , що в свою чергу залежить від різниці реального і прогнозованого значення вихідної ознаки, та в загальному випадку описується формулою (2.1):

$$Q: E = f'(y - \hat{y}) \rightarrow \min, \quad (2.1)$$

де y – реальне значення вихідної ознаки;

$\hat{y}$ – прогнозоване значення вихідної ознаки.

Формальна математична постановка задачі може бути сформульована наступним чином: знайдення оптимального значення заданої функції якості регресійної моделі, де критерій оптимальності Q залежить від різниці між реальною ринковою вартістю футболіста та прогнозованою вартістю футболіста.

2.2 Аналіз та обробка даних

На даному етапі необхідно виконати першу стадію опису залежності $f(w, x)$, а саме знаходження структури f . Можна стверджувати, що задача апроксимації такої залежності є задачею множинної регресії, адже кількість ознак, що впливають на залежну змінну, як видно з таблиці 1.2, становить 34 ознак.

Аналіз початкових даних дає змогу оцінити структуру залежності між ознаками. Спочатку необхідно сформувати датасет, завантаживши дані з відкритих джерел. Для цього було створено скрипт мовою Python, що витягує дані з офіційного сайту FIFA [36]. Такі дані є сирими та мають стандартний формат, тому їх необхідно попередньо обробити.

Перше, що потрібно зробити, це провести нормалізацію значень ознак, зокрема незалежних. Це звична процедура для вирішення задачі засобами

машинного навчання, що проводиться для пришвидшення навчання, при використанні нейронних мереж (НМ). Через те, що майже всі ознаки, представлені у інтервальній шкалі у границях від 0 до 100, нормалізація не є обов'язковою, однак є бажаною. Факт однорідності ознак може бути використаний при виборі методу нормалізації. Як правило, використовується стандартизація (Z-Score Normalization), для приведення значень ознаки відносно середнього значення [37]. Такий метод користується попитом через те, що на значення ознаки приведені таким засобом не впливають викиди – аномально великі або аномально малі значення. В розглянутій вибірці таких значень просто немає, тому є сенс скористатися іншим методом – нормалізацією вибірки, що виконується за формулою (2.2):

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} * (u - l) + l, \quad (2.2)$$

де x – значення ознаки для нормалізації;

x_{min} – мінімальне значення ознаки на вибірці;

x_{max} – максимальне значення ознаки на вибірці;

l – мінімальне значення нормалізованої вибірки;

u – максимальне значення нормалізованої вибірки.

Обробка ж залежної ознаки – ціни футболісту, носить більш актуальний характер. Гістограму розподілу кількості футболістів визначеної ціни зображено на рисунку 2.2.

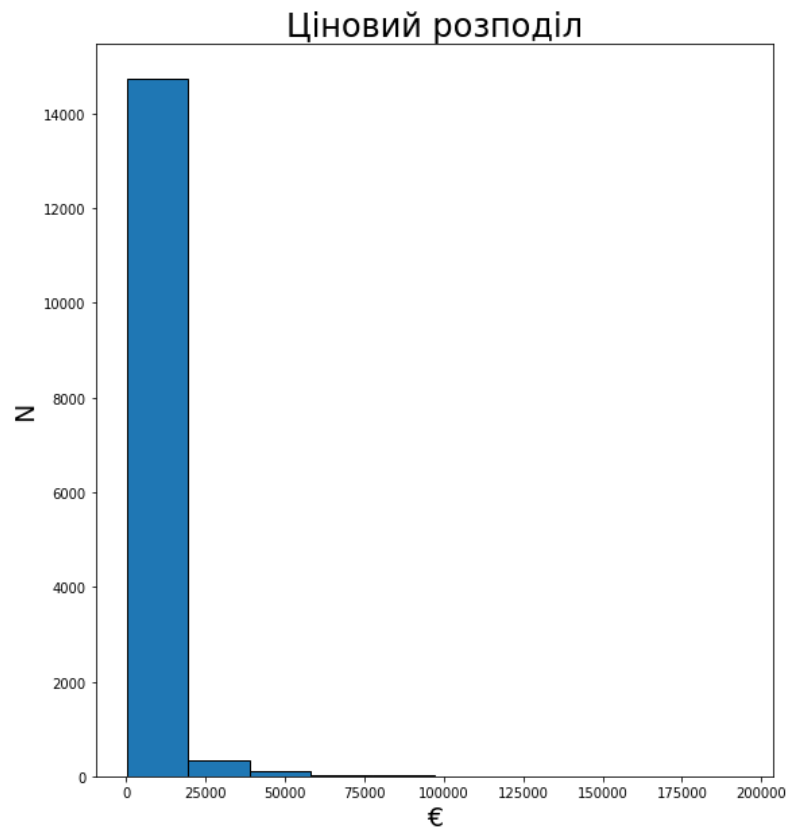


Рисунок 2.2 – Ціновий розподіл футболістів

Як видно з рисунка 2.2, більшість футболістів коштують на трансферному ринку до 25 мільйонів євро, тому медіанне значення ціни, відносно всього інтервалу, досить низьке. Такий розподіл випадкової величини, як видно з гістограми, асиметричний.

За такого розподілу модель може не точно прогнозувати ціну дуже дорогих гравців. Це до речі і стало головною проблемою дискретного підходу, у роботі [34], де до кошовних футболістів віднесено усіх, хто дорожче 36 мільйонів євро (при максимальній ціні 115 мільйонів).

Методів нормалізації подібної вибірки декілька [38], наприклад є сенс застосувати логарифмічне перетворення. Можна казати, що випадкова величина розподілена за логарифмічно нормальним законом (приблизно) [39], тобто логарифм такої випадкової величини має нормальний розподіл.

Після логарифмічного перетворення, розподіл залежної ознаки зображено на рисунку 2.3.

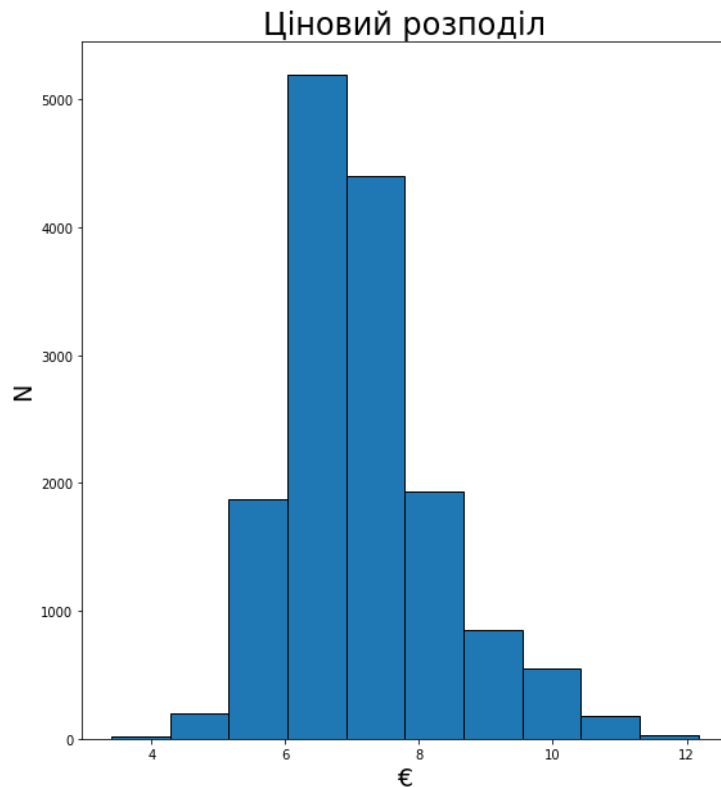


Рисунок 2.3 – Ціновий розподіл після логарифмічного перетворення

Як видно з рисунка 2.3, у порівнянні з рисунком 2.2, ціна на футболістів розподілена за нормальним законом. Єдиним незначним штрафом за це є те, що вартість футболіста повинна бути перетворена назад до звичної шкали.

Після нормалізації значень вхідних та вихідної ознак, з початкової множини необхідно провести відбір підмножини ознак, на основі їх значущості. Гіпотеза, на якій базується відбір говорить: добра підмножина ознак — це та, яка містить ознаки, які сильно корелюють з цільовою ознакою, але не корелюють між собою [40].

Для дослідження ступеню взаємозв'язку ознак застосовано коефіцієнт кореляції Пірсона. Він дозволяє визначити силу лінійного взаємозв'язку двох змінних. Попарне визначення коефіцієнту кореляції для всієї множини ознак представлено у вигляді теплової карти [41] та зображено на рисунку 2.4.

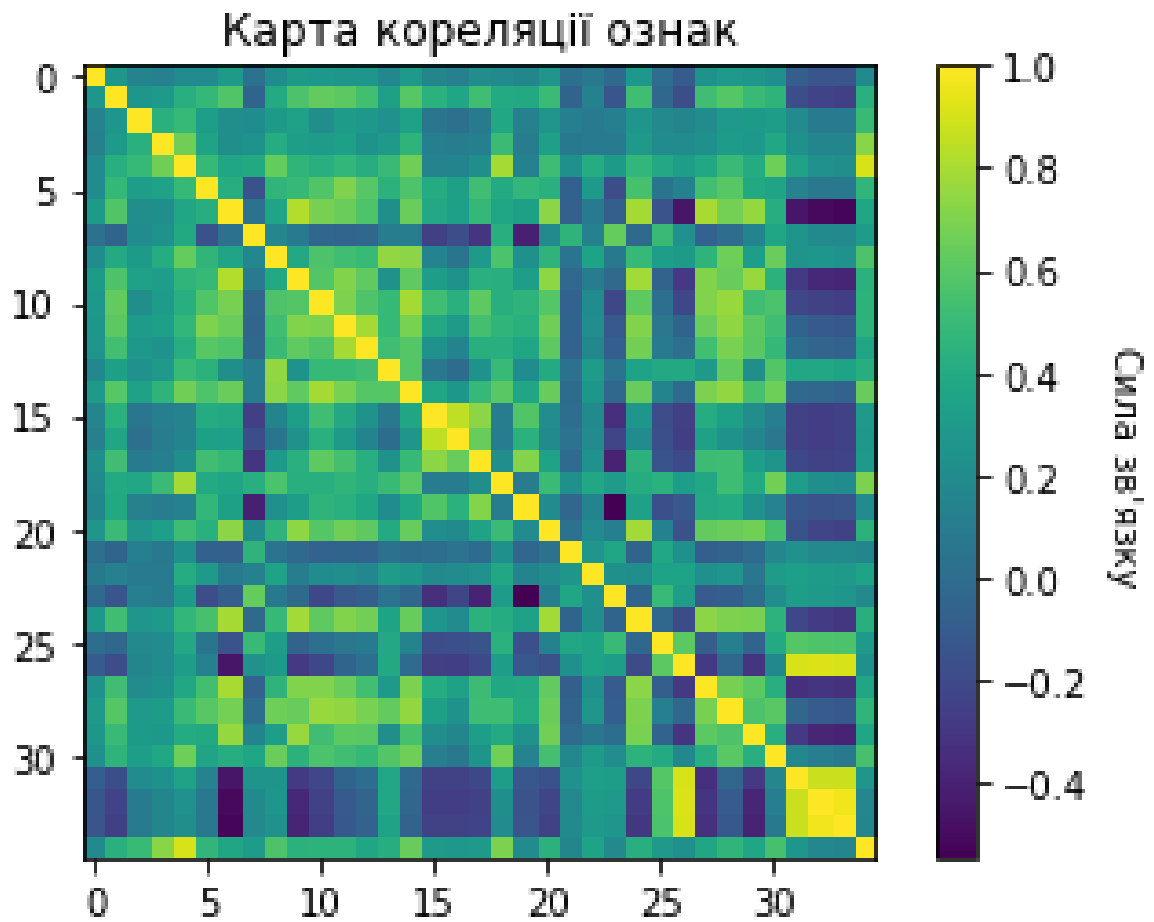


Рисунок 2.4 – Теплова карта кореляції ознак

Входячи з початкової гіпотези, необхідно відсіяти ті ознаки, які сильно корелюють між собою. Збільшення масштабу у місці проблемної області з рисунку 2.4 зображено на рисунку 2.5.

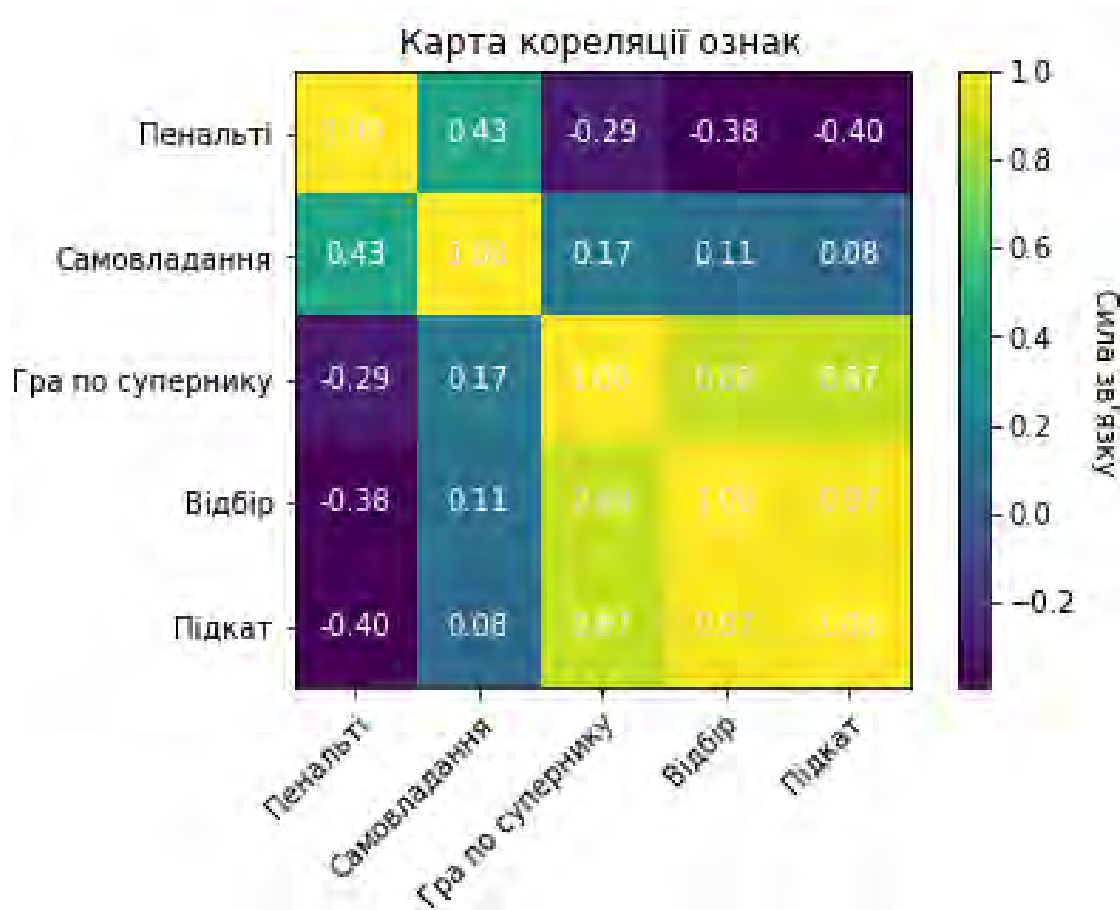


Рисунок 2.5 – Теплова карта кореляції ознак у приближенні

Як видно з рисунку, три незалежні ознаки сильно корелюють між собою, а саме: «гра по супернику», «відбір», «підкат». Це не дивно, адже з точки зору предметної області вони всі характеризують здатність гравця ефективно грати у обороні. Згідно з результатами математичного та предметного аналізу, було прийнято рішення ці три ознаки звести до однієї, з назвою «захист».

Графік залежності значень коефіцієнту кореляції між незалежними змінними та залежною змінною «ціна» зображено на рисунку 2.6.

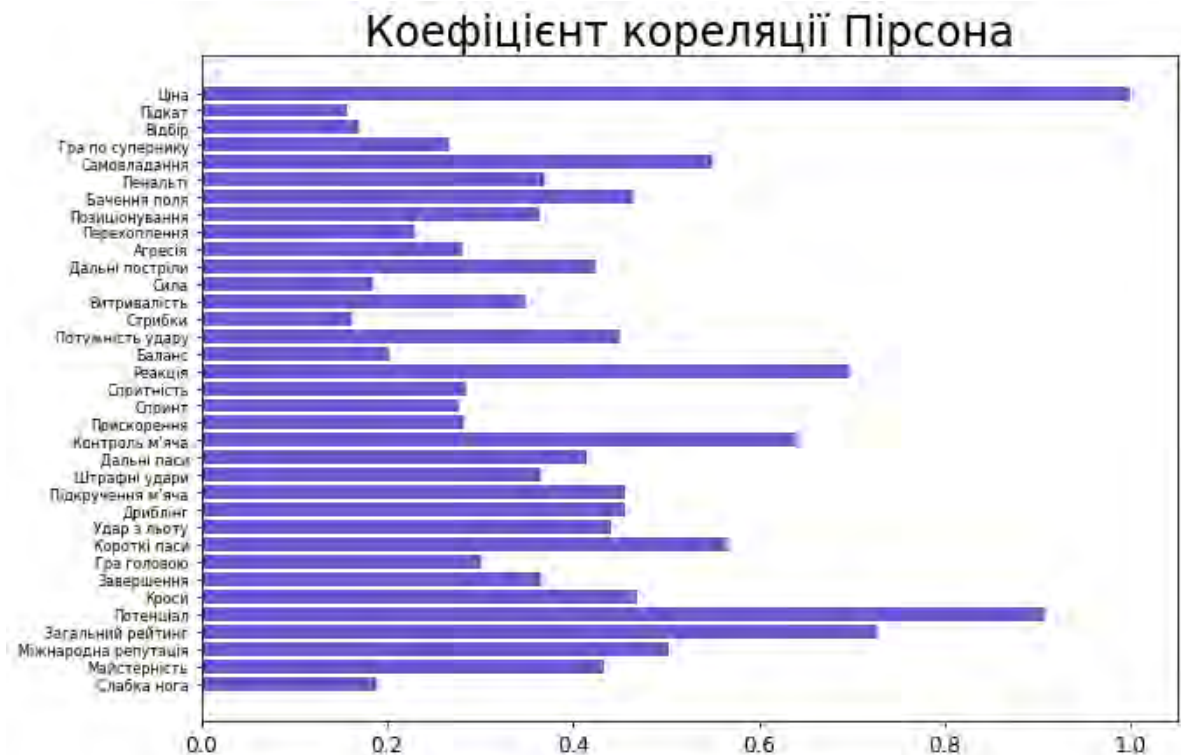


Рисунок 2.6 – Коефіцієнт кореляції між незалежними ознаками та ціною

Колінеарними називаються такі ознаки, які сильно корелюють одна з одною. У машинному навчанні це призводить до зниження рівня узагальнення даних та рівня інтерпретованості моделі [42]. Як правило, колінеарними ознаками можна вважати такі, кореляція яких переважає 0.7.

На рисунку 2.6 видно, що «загальний рейтинг» можна віднести до колінеарної ознаки, відносно цільової, тому таку ознаку необхідно видалити з вибірки. У предметній області загальний рейтинг є узагальнюючим показником усіх характеристик футболіста, який не є чітко визначеним, тому логічним буде від нього позбавитися.

2.3 Побудова математичної моделі

2.3.1 Нова концепція аналізу футбольної трансферної активності.

Вербальна модель

Як видно з таблиці 1.1, кожний програмний продукт будується навколо певного методу, метою якого є визначення вартості футболіста. Для CIES

такий метод реалізовано через побудовану математичну модель (пункт 1.2.1), для Transfermarkt використовується метод з назвою «мудрість натовпу» (пункт 1.2.2). Роботу обох схематично зображено на рисунку 2.7.

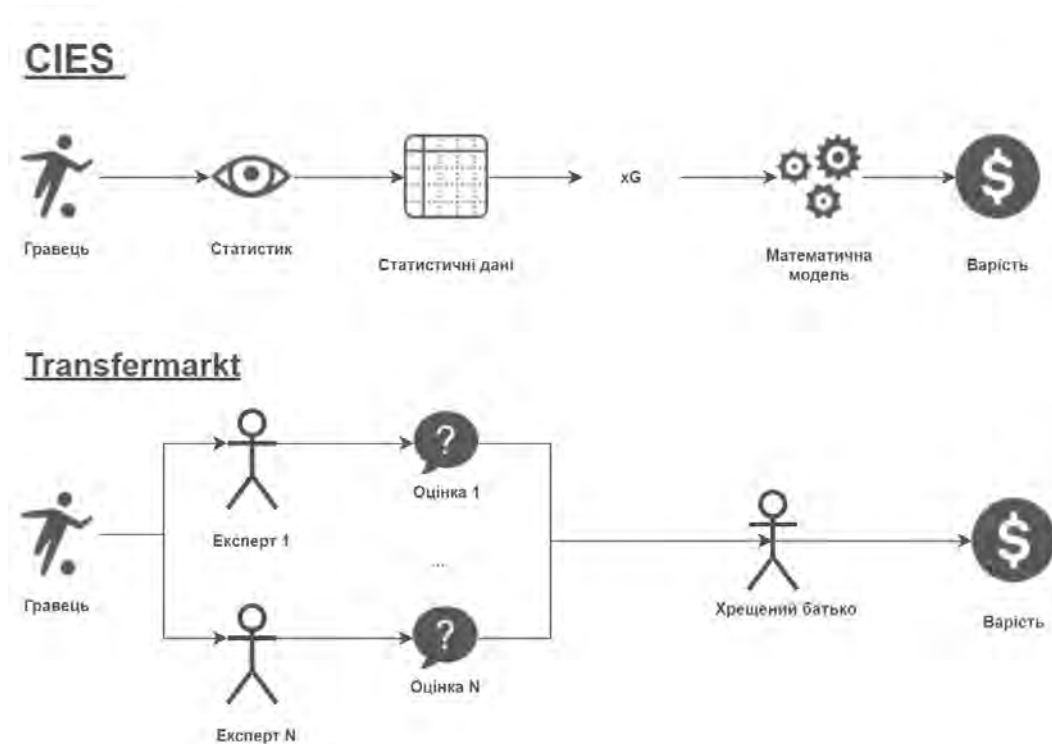


Рисунок 2.7 – Методи визначення ціни гравця

Як можна побачити з рисунку 2.7, метод CIES оцінює трансферну вартість гравця, судячи з об’єктивних статистичних показників про гру такого футболіста. Тобто результат роботи такого методу не залежить від людини (якщо ж звісно рахувати, що статистик, відповідальний за збір даних не помиляється), а рішення про ціну футболісту приймається виключно моделлю.

Напроти, Transfermarkt математичну модель не використовує, а замість неї рішення приймає експерт зі знань – «хрещений батько». При чому, при визначенні ціни, такий експерт не стільки спирається на статистичні дані, скільки усереднює думки інших експертів.

Метод, запропонований в рамках цієї роботи, покликаний поєднати обидва розглянутих принципи визначення трансферної вартості гравця. Роботу такого гібридного методу зображено на рисунку 2.8.

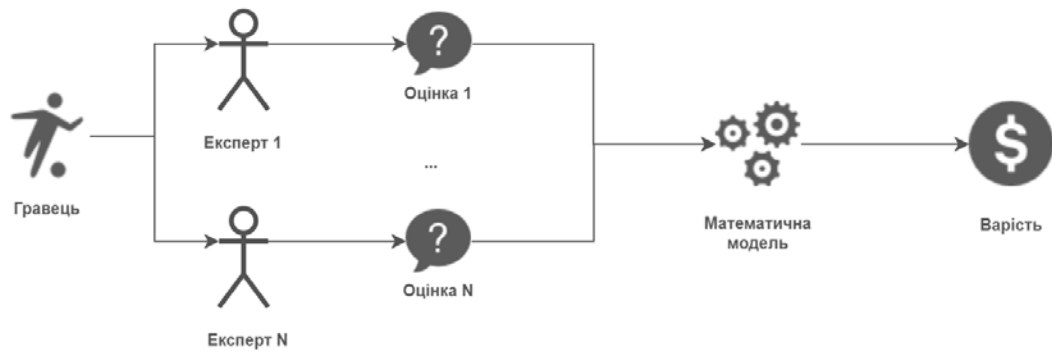


Рисунок 2.8 – Запропонований гібридний метод

Від Transfermarkt взято першу частину методу, а саме вибір та обробку даних. Як вже було сказано раніше, кожен експерт зі знань оцінює футболіста зі своєї точки зору, характеризуючи його гру. У подальшому такі дані повинні бути чітко формалізовані та подані до математичної моделі. Можливість побудову функціонуючої модулі було доведено обсерваторією CIES.

Генерацію такої моделі можна пояснити простим реченням: замість статистики – оцінка експертів, а замість «хрещеного батька» – математична модель.

Розроблювану модель у вербальній формі можна представити наступними тезами. Користувач програми повинен оцінити гру футболісту у спеціально підібраній системі ознак, зрозумілій будь-якій людині. Кожна така ознака має мати свою імпровізовану шкалу – метод обрахунку та умовну одиницю виміру. Значення кожної ознаки називається оцінкою.

На основі таких оцінок програма повинна розрахувати один, агрегований, фінальний показник – ціну гравця. Така ціна повинна бути представлена у грошовому еквіваленті у тих рамках, які сформовані трансферним футбольним ринком.

2.3.2 Вирішення задачі регресії за допомогою багат шарової нейронної мережі прямого поширення

Нейронна мережа – це математична модель, здатна вирішувати задачу регресії. У разі, якщо залежність між незалежними ознаками та цільовою змінною є лінійною, задача класифікується як задача лінійної регресії. У випадку коли множинна регресія не може бути описана лінійною функцією, задача визначається як задача нелінійної регресії.

НМ покликана вирішувати обидва види задач. Якщо на етапі структурного синтезу моделі вирішено, що вирішувана задача – це задача лінійної регресії, то в такому разі необхідно побудувати одношаровий персептрон, з лінійною функцією активації. Одношарового персептрона достатньо, тому що НМ з великою кількістю шарів з лінійною функцією активації, беручи до уваги таку властивість лінійності як адитивність, еквівалентна одношаровій мережі.

У разі класифікації задачі як задачі нелінійної регресії, застосовують багат шарову нейронну мережу (БНМ) з нелінійною функцією активації. Для такої мережі зазвичай достатньо одного прихованого шару, а подібна модель є універсальним апроксиматором, що і необхідно для вирішення задачі.

Користуючись тим, що на початку розв'язання задачі регресії немає жодних підстав на користь лінійності, задачу вирішено класифікувати як нелінійну. Це позбавить від можливої помилки специфікації, у протилежному випадку.

2.3.3 Структурний синтез моделі

Факт використання БНМ прямого поширення, для вирішення розглянутої задачі регресії, дає можливість висувати наступні твердження про модель:

- вхідний шар, що подає значення вхідного вектору обчислювальному слою, містить 31 нейрон, за кількістю вхідних ознак;
- один прихований шар, з нелінійною функцією активації;
- вихідний шар з одним нейроном та лінійною функцією активації, для отримання неперервного значення вихідної ознаки.
- метод градієнтного спуску для оптимізації критерію якості, в ході навчання моделі.

Деякі рішення, стосовно проєктування НМ, було прийнято спираючись на загальноприйняті відомості та на авторитети:

- функцією помилки, вираженою у загальному вигляді формулою 2.2, вибрано середньоквадратичну помилку (MSE);
- оптимізатор Adam [43].

Із загальних міркувань та за вимоги нелінійності, функцією активації для прихованого шару було вибрано Rectified Linear Unit (ReLU). Суттєвої різниці між логістичною функцією та ReLU немає, тим паче остання всього лише є лінійною апроксимацією сигмоїду. Через те, що модель має лише один прихований шар, проблем з насиченням нейронів не виникне.

Для визначення кількості нейронів прихованого шару можна скористатися наступною евристикою: така кількість повинна бути значно меншою за кількість вхідних ознак [44]. Для початку ініціалізовано значенням в 550 нейронів. Забігаючи наперед, різна конфігурація кількості нейронів прихованого шару моделі не приносила кращих результатів, за обрану – 550 нейронів.

2.4 Висновки за розділом 2

У розділі 2 було виконано побудову математичної моделі у вигляді нейронної мережі, згідно до поставленої задачі. Така математична модель може приймати персоналізовані оцінки гри окремих футболістів від

користувачів, диференційовані за певними ознаками, а на виході видавати прогнозовану вартість гравця – ціну на трансферному ринку.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

В даному розділі обґрунтовано вибір мови програмування, розглянуто основні особливості програмних засобів, за допомогою яких реалізовано дипломну роботу та розроблено структурну схему програми.

3.1 Вибір інструментарію розробки програмного забезпечення

Вибір програмних засобів, що будуть використовуватися при розробці програмного забезпечення, повинен бути обумовлений відповідними вимогам до продукту, які виділяються з технічного завдання.

Головною особливістю розробки є розділення можливостей програми на дві частини: здатність реалізації моделі за допомогою машинного навчання, зокрема робота з нейронною мережею, та реалізація клієнт-серверної архітектури. Справедливо буде припустити, що і інструментарій для різних цілей буде різний, а тому, розроблені за допомогою такого інструментарію компоненти повинні доречно взаємодіяти одна з одною у рамках цілісної конструкції програми.

3.1.1 Аналіз технічного завдання та формування вимог до використовуваних програмних засобів

Виділимо визначені вимоги, згідно з технічним завданням. З точки зору звичайного користувача, програма має задовольняти можливість виконання зазначених нижче функцій:

- проводити реєстрацію нових користувачів та авторизацію старих;
- здійснювати збір, обробку та збереження особистих даних про користувача;
- надавати можливість зміни особистих даних користувача;

- здійснювати чіткий контроль часу всередині системи та надавати користувачам інформацію про цей час;
- інформувати користувача про деякі останні події та помилки;
- надавати можливість переглядати список футболістів;
- виконувати пошук, рух за списком, сортування футболістів;
- надавати можливість користувачу виставляти гравцям власні суб'єктивні оцінки – змінювати ігрові параметри футболіста, що безпосередньо впливають на зміну його ціни;
- згідно до заданих користувачем параметрів гравця, виконувати обчислення ринкової вартості футболіста у грошовому еквіваленті;
- зберігати зміни, впроваджені користувачем, як то ігрових показників футболіста, так і його ціни;
- відміняти впроваджені зміни показників футболіста;
- приводити усі показники футболіста до їх мінімальних значень, в залежності від множини допустимих значень відповідних ознак;
- надавати історію зміни ринкової вартості футболіста, відносно користувача.

З точки зору адміністратора, програма повинна виконувати усі вищезазначені функції, а також виконувати низку додаткових:

- додавати до списку гравців нового футболіста;
- видаляти футболіста зі списку гравців;
- надавати можливість редагувати параметри гравців. До таких параметрів відносяться: ім'я, вік, національність, команда у якій грає футболіст, позиція на якій грає та трансферний статус;
- виводити повідомлення про успішне чи неуспішне проведення операцій додавання, видалення та редагування футболістів. У разі невдачі адміністратор повинен отримати відповідні вказання на вірний формат поля, з огляду на допущену помилку;

– у разі, якщо адміністратор має доступ до серверної інформаційної панелі, програма повинна виводити інформацію про дію, яку виконую сервер, клієнта, який цю дію запросив, час та результат роботи запиту.

На рисунку 3.1 зображено діаграму з основними функціями користувача.

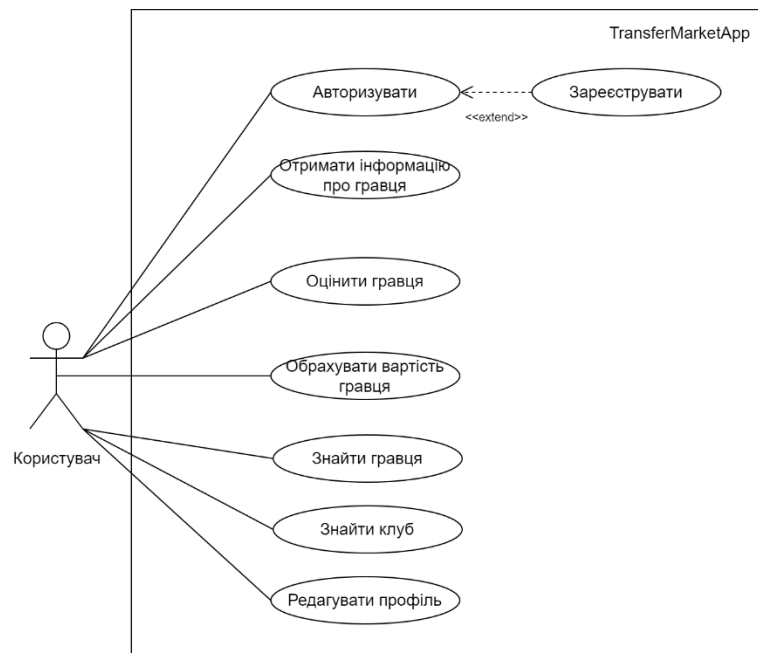


Рисунок 3.1 – Діаграма використання програми користувачем

На рисунку 3.2 зображено діаграму з функціями адміністратора.

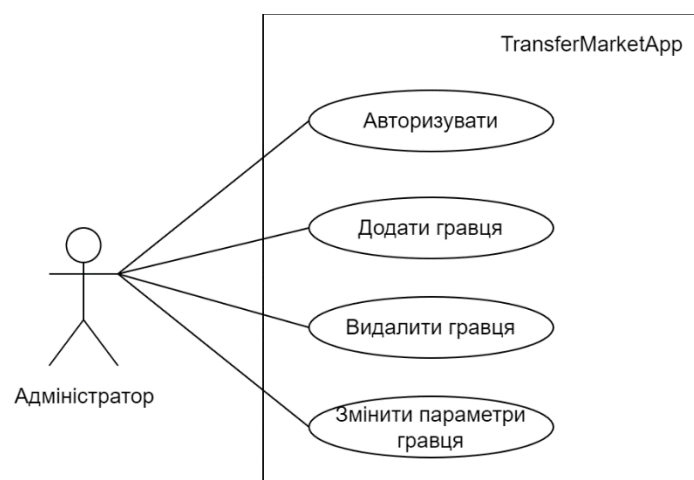


Рисунок 3.2 – Діаграма використання програми адміністратором

Програмні засоби, які будуть використані при розробці продукту, повинні, в кінцевому рахунку, задовольняти наступним вимогам:

- вбудована функціональність. Створення програми передбачує наявність вже готових функцій, які не повинен створювати розробник: наприклад робота з часом чи наявність функцій обробки та аналізу даних;

- робота з даними. Облік футболістів, з усіма їх необхідним параметрами, та безпосередній обрахунок вартості гравців рано чи пізно призведе до збільшення витрат пам'яті та часу, тому потрібно зберегти ресурси програми;

- широкі можливості створення графічного інтерфейсу. Все ж таки один з головних акцентів при створенні програми, робиться на розробку зручного та зрозумілого графічного інтерфейсу. Успіх розробки усього проєкту буде залежати від розкриття можливостей та особливостей проблемної сфери;

- створення розмітки інтерфейсу. Виконання такої вимоги дозволило б швидше та більш звично створити дизайн інтерфейсу, зробити його масштабованим та більш повноцінно використати потенціал мови програмування;

- створення та інтегрування бібліотек динамічного компонування. Деякі схожі за функціональністю можливості програми можна було б виділити у бібліотеки, підключення яких надавало б користувачу додаткових можливостей роботи. Така своєрідна абстракція обумовила б створення різних версій програми;

- підтримка клієнт-серверної архітектури. Більш того, було б добре, якщо б було декілька можливостей реалізації такого типу архітектури, з яких можна було б вибрати найліпший у даному контексті;

- робота під Windows. Згідно з технічним завданням, програма повинна функціонувати на ПК, на якому встановлена ОС Windows 10. Таке суворе обмеження у подальшому може бути розширене, наприклад, до створення мобільної версії програми;

- швидка розробка, для виконання в зазначені строки;
- контроль версій;
- зручність відлагодження проєкту. Для перегляду результатів та для швидкого отримання рішення необхідні зручні інструменти відлагодження.

Умови розробки математичної моделі теж висувають ряд додаткових вимог до застосованих програмних засобів:

- чисельний аналіз даних. Початкові дані, як правило, є сирими та перед тим як подати їх до математичної моделі, вони мають бути оброблені (формалізовані) та проаналізовані за допомогою відповідних функцій;
- структури даних. У ході налагодження моделі, дані повинні бути представлені за допомогою вихідних типів мови програмування чи стандартних розширених типів та, бажано, мати вбудовану функціональність, наприклад гнучку ітерацію;
- візуалізація, тобто можливості представлення формалізованих даних у простому для людини вигляді – графіка, діаграми чи теплової мапи;
- машинне навчання. Широкі можливості для реалізації машинного навчання, зокрема для нейронних мереж;
- робота з файлами, що зберігають дані у різних форматах;
- витягування даних. Звісно, що необхідний набір початкових даних може бути відсутній у вільному доступі для простого завантаження, тому необхідно мати інструменти для витягування даних, наприклад, з вебсторінок.

3.1.2 Вибір мови програмування

Сформуємо критерії аналізу мов програмування у відповідності до вимог до програмних засобів. Порівняння мов Python та C# за заданими критеріями наведено у таблиці 3.1.

Таблиця 3.1 – Аналіз мов програмування

Критерій	Python	C#
1	2	3
Вбудована функціональність	Величезна стандартна бібліотека, що поширюється разом з Python та сумісна з різними платформами [45].	Багата бібліотека C# заснована на .NET Framework.
Робота з даними. Підвищення продуктивності	Процес роботи з даними, зокрема як і наука про дані, зосереджений навколо процесу ETL (витяг-перетворення-завантаження). Ця особливість робить мову Python ідеально придатною для таких цілей. Наявні фундаментальні пакети: Pandas та NumPy.	Засоби мови C# дозволяють інтегрувати функції та бібліотеки, написані мовою C++, завдяки властивості C-подібності. Функціональність, створена засобами C++, може забезпечити підвищення продуктивності та часу виконання програми. Некерований код необхідно буде інтегрувати у керований, створений на C#.
Створення графічного інтерфейсу	Розробка користувацького графічного інтерфейсу мовою Python справедливо не є частим явищем. Однак все одно для такої потреби існує велика кількість відповідних бібліотек:	Розробка графічного інтерфейсу ґрунтується на двох головних компонентах операційної системи Windows: user32 та GDI+. Основана на них, технологія Windows Presentation Foundation (WPF) є частиною

Продовження таблиці 3.1

1	2	3
	вбудована Tkinter, wxPython, PySimpleGUI і PyQt. Усі бібліотеки об'єднує звична кросплатформеність, однак натомість і абстрагування від складних технічних засобів, для створення вдосталь складного інтерфейсу.	екосистеми платформи .NET і являє собою підсистему для побудови графічних інтерфейсів. Її особливостями є незалежність від розширення екрану, створення власних елементів керування та можливість декларативного проєктування інтерфейсу. Також є підтримка шаблону Model View ViewModel (MVVM).
Створення розмітки інтерфейсу	Розмітка інтерфейсу, наприклад у Tkinter, звичайно задається програмним засобом, однак навіть для створення десктоп додатку є можливість використання HTML [46].	Саме можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, заснованої на xml, і представляє альтернативу програмному створенню графіки та елементів управління, а також надає можливість комбінувати XAML і C#.
Створення та інтегрування бібліотек	В принципі, робота з dll це не дуже гарна ідея для мови Python, оскільки Python	У MVS dll легко створюється через проєкт відповідного типу. Після

Продовження таблиці 3.1

1	2	3
	робить акцент на незалежності платформи, а dll від неї залежить. Однак при необхідності, за допомогою технології маршалінгу [47], яка звичайно є і у мові C#, можна звернутися до вмісту такої бібліотеки.	створення проєкту виконується збірка. Інтегрування, вимагає лише додання до проєкту DLL та застосування звичайного синтаксису мови для звернення до матеріалів бібліотеки.
Створення та інтегрування бібліотек	В принципі, робота з dll це не дуже гарна ідея для мови Python, оскільки Python робить акцент на незалежності платформи, а dll від неї залежить. Однак при необхідності, за допомогою технології маршалінгу [47], яка звичайно є і у мові C#, можна звернутися до вмісту такої бібліотеки.	У MVS dll легко створюється через проєкт відповідного типу. Після створення проєкту виконується збірка. Інтегрування, вимагає лише додання до проєкту DLL та застосування звичайного синтаксису мови для звернення до матеріалів бібліотеки.
Підтримка клієнт-серверної архітектури	Найбільш стандартний спосіб створення додатка з клієнт-серверною архітектурою – це використання сокетів за допомогою модуля socket. В такому випадку буде налаштовано обмін між	Подібну логіку можна налаштувати і методами C#. Також можна створити Windows Communication Foundation (WCF) сервіс, який буде прошарком між базою даних (БД) і клієнтами. Тільки сервіс

Продовження таблиці 3.1

1	2	3
	сервером та клієнтом в звичному байтовому форматі.	буде зберігати налаштування підключення до БД; там же в сервісі будуть зберігатися усі інструкції до дії над даними у вигляді методів, а клієнти будуть їх викликати. Таким чином про базу даних буде знати тільки WCF, а клієнти вже отримувати готові дані.
Робота під Windows. Кросплатформеність	Ключовою особливістю мови Python є те, що її код спочатку трансліюється в спеціальний байт-код, сумісний із різними платформами. А потім цей байт-код виконується віртуальною машиною PVM (Python Virtual Machine). Така архітектура забезпечує кросплатформеність.	Мова С#, реально, виявилась не кросплатформеною, незважаючи на існування неофіційних .NET оточень під різними платформами і навіть потенційну бінарну сумісність між платформами.
Машинне навчання	Python має величезну кількість бібліотек для аналізу даних та обчислення за допомогою машинного навчання. Такі бібліотеки надають доступ	С# також підтримує машинне навчання за допомогою платформи машинного навчання .NET Core, ML.NET, яка дозволяє користувачам

Продовження таблиці 3.1

1	2	3
	до високопродуктивних структур даних та швидких методів. Чого лише варті NumPy для наукових обчислень, SciPy для більш складних обчислень, Pandas для загального аналізу даних, Scikit-learn для машинного навчання та TensorFlow для побудови нейронних мереж [48].	впроваджувати власні моделі машинного навчання в додатки. Також є підтримка пакету TensorFlowSharp, що надає доступ до функціоналу TensorFlow, сумісним з C#.
Візуалізація	Деякі бібліотеки, такі як Matplotlib, дозволяють розробникам створювати діаграми, гістограми та графіки для кращого представлення даних та візуалізації. Створення графіки дуже легке та майже не обмежене.	Побудова графіків теж доступна за допомогою спеціальних бібліотек, наприклад OxyPlot. Однак необхідно додати, що візуалізація мовою C# менш легка при програмному описі графіка, та частіше використовується не для тестування чогось, а як частина інтерфейсу готової програми.
Швидкість виконання	Python є мовою з повною динамічною типізацією, автоматичним керуванням пам'яттю, тому у порівнянні зі статичними	C# — одна з найшвидших мов, що використовуються настільки ж часто. І це досягається саме самою мовою, а не через її

Продовження таблиці 3.1

1	2	3
	мовами програмування Python програє за швидкістю виконання.	реалізацію, завдяки інфраструктурі Common Language Infrastructure (CLI). C# також забезпечує швидку компіляцію, є паралельним та запускає кілька потоків одночасно.
Простота та швидкість розробки	Підтримка складних алгоритмів машинного навчання компенсується простим синтаксисом мови. Важливо, що розробникам моделей не потрібно витрачати багато часу на технічні проблеми. Так, час розробки на Python доволі швидкий, а тому написаний код може бути меншим за обсягом в рази.	Мова програмування C# складніша, тому що вимагає розуміння більшої кількості концепцій та конструкцій. Це означає, що розробка програм займає більше часу, а тому доводиться виконувати більше роботи.
Спільнота та документація	Для проєктів машинного навчання та глибокого навчання переважна більшість розробників використовує Python. Це свідчить про те, що більшість «світливих розумів» світу можна знайти у спільнотах Python [49].	C# не поступається конкурентові у загальній кількості документації. Однак поступається у загальній популярності серед розробників у сфері науки про дані, а тому і має меншу кількість готових рішень чи рішень проблем у вільному доступі.

Роблячи висновки з проведеного аналізу, що наведено у таблиці 3.1, можна скласти інформацію для більш наочного порівняння можливостей Python та C#, що наведено у таблиці 3.2.

Таблиця 3.2 – Порівняння мов програмування

Критерій	Python	C#
Вбудована функціональність	+	+
Робота з даними	++	+
Підвищення продуктивності	-	+
Створення графічного інтерфейсу	±	+
Створення розмітки інтерфейсу	±	+
Створення та інтегрування бібліотек	-	+
Підтримка клієнт-серверної архітектури	+	++
Робота під Windows	+	++
Кросплатформеність	++	-
Машинне навчання	+	±
Візуалізація	++	-
Швидкість виконання	-	++
Простота та швидкість розробки	++	+
Спільнота та документація	+	+

З таблиці видно, що Python та C# тісно конкурують між собою, деś одна мова поступається іншій, деś вона навпаки краща, тому було прийнято рішення розподілити «обов'язки» між програмними компонентами, створеними різними мовами.

Опитування розробників (Python Developers Survey), проведене Stack Overflow у 2020 році, показало, що найчастіше розробники використовують мову Python для аналізу даних – 55%, а 40% опитуваних використовують цю мову програмування у сфері машинного навчання [50]. Для порівняння, друга за популярністю ціль використання мови, веброботка, має 50%.

Підсумовуючи дані, зібрані в процесі порівняння мов Python та C#, можна зробити висновок, що використання першої є більш оптимальним варіантом для створення математичної моделі для прогнозування ціни футболіста. За це рішення говорять наступні чинники:

- велика кількість різносторонніх бібліотек для аналізу даних та машинного навчання;
- прості засоби візуалізації результатів, без обмежень;
- прості правила написання коду, та, як наслідок, більш швидке досягнення бажаного результату;
- більш висока популярність ніж у C# у сфері машинного навчання, а тому і більш широкий обсяг документації та допоміжної інформації;
- автоматизований та простий у використанні інструментарій для витягування даних та проведення маніпуляцій над ними;
- розробка відбувається незалежно від платформи.

Для створення ж повноцінного багат шарового додатку з роботою з даними, логікою та графічним інтерфейсом будуть використані можливості мови C#:

- однозначна розробка від Windows;
- широкий набір стандартних бібліотек;
- зрозумілі засоби налагодження роботи з даними, зокрема з БД;
- висока продуктивність, код одразу компілюється у машинний код, без, наприклад, інтерпретатора як у Python;
- легкі засоби налагодження клієнт-серверної архітектури;
- можливість створення власних бібліотек, хоч мовою C++, та інтегрування їх у додаток;
- потужні засоби створення графічного інтерфейсу;
- багата документація, а спільнота генерує в достатній кількості допоміжних матеріалів.

3.1.3 Вибір середовища розробки мовою Python

До певного часу використання PyCharm, як інтегрованого середовища розробки (IDE) мовою Python, у рамках вирішення тривіальних завдань, не ставилось під сумнів. Не дивно, що якщо машинне навчання настільки поширена область для розробки Python [6], то мають існувати і дещо інші IDE, більш пристосовані для такої області. Одним з таких середовищ є Spyder. Порівняльну характеристику Spyder та PyCharm, загалом та з огляду вирішення задач машинного навчання, наведено у таблиці 3.3.

Таблиця 3.3 – Аналіз середовищ розробки мовою Python

Критерій	Spyder	PyCharm
Доступність	+	±
Кросплатформність	+	+
Універсальність	-	+
Встановлення	+	±
Відлагодження	+	±
Навантаження комп'ютеру	+	-
Інтерфейс	+	-
Декілька консолей	+	-
Редактору коду	±	+
Jupyter Notebook	+	-
Допоміжна інформація	+	+
Перегляд стану змінних	+	-
Перегляд графічних об'єктів	+	-
Контроль версій	-	+
Кастомізація	-	+
Зручність гарячих клавіш	-	+

Як можна побачити з таблиці, в загальному випадку обидва середовища дещо схожі. Вони обидва:

- кросплатформні;
- дозволяють створювати проекти з розгалуженою файловою структурою;
- мають зручний текстовий редактор з підсвічуванням синтаксису;
- мають можливість додавати зовнішні плагіни;
- наповнені зручною допоміжною інформацією.

Spyder, у порівнянні з конкурентом, має деякі загальні недоліки:

- Spyder створений виключно під розробку мовою Python, PyCharm же дозволяє редагувати файли різних мов, що можна вважати особливістю;
- відсутня вбудована функціональність Git, тому необхідно користатися командним рядком, що можна вважати суттєвим мінусом;
- PyCharm має значно потужнішу кастомізацію, що означає що середовище можна підлаштувати під себе. Більш того, за замовчуванням гарячі клавіші Spyder налаштовані невдало, а їх призначення потрібно змінити;
- редактор коду PyCharm більш чутливий та кросмодульний.

До переваг Spyder можна віднести:

- обидва середовища розробки безкоштовні, однак Spyder «більш безкоштовний»;
- Spyder значно простіше встановити;
- Spyder дозволяє працювати одразу у необмеженій кількості консолей, з різними встановленими середовищами, перемикаючись між ними під потреби розробника;
- достатньо велика кількість прикладів задач на тему машинного навчання створена у блокнотах Jupyter. Плагін Spyder-Notebook дозволяє працювати з такими блокнотами одразу у середовищі Spyder [51];

- Spyder надає розробнику можливість рядкового виконання програми (на основі клітинок – cell) або одразу всього коду;
- журнал всіх вхідних та вихідних даних, з можливістю його редагування, редагування типу відображуваних змінних та перегляду самого журналу. Такий браузер змінних поступово змінюється, в залежності від стадії чи місця виконання програми. Журнал змінних відображує масиви, колекції та будь-які структури даних [52];
- інтерфейс зовні та за можливостями нагадує MATLAB.
- візуалізація масивів Numpy [53];
- візуалізація графіків безпосередньо у середовищі Spyder;
- PyCharm для стабільної роботи потребує 8 Гб оперативної па'яті, у той час коли для роботи Spyder достатньо 4 Гб чи навіть менше [54].

Сама назва середовища Spyder розшифровується як The Scientific Python Development Environment, тому не дивно що таке IDE створене та використовується переважно для наукових обчислень чи у сфері науки про дані. І насправді, з порівняння середовищ можна зробити висновок, що PyCharm більш доречно використовувати у великих та тривіальних проєктах, однак для машинного навчання більш привабливе середовище Spyder. Саме воно і буде використано при створенні моделі прогнозування вартості гравця.

3.1.4 Особливості технології WPF у порівнянні з Windows Forms

Для створення клієнтського додатку буде використано мову C# та технологію WPF.

Фактично, існують дві головні технології побудови десктопного програмного продукту з користувацьким графічним інтерфейсом засобами C#: WPF та Windows Forms. Перша з них більш нова, а тому має дві визначні особливості, які роблять Windows Forms застарілим та фактично непридатним.

Перша особливість – це відображення графічних компонентів за допомогою DirectX. Розробка графічного інтерфейсу для Windows додатків ґрунтується на двох компонентах операційної системи:

- User32 – містить основні елементи вікна;
- GDI/GDI+ – надає функціонал для малювання фігур, тексту.

Еволюція графічних систем пройшла шлях від складної системи MFC до розробки мовами .Net і WinForms. Однак усі удосконалення технологій все одно спиралися на User32 і GDI/GDI+, а значить, не могли подолати фундаментальні обмеження базових компонентів.

Для вирішення цієї проблеми була розроблена технологія WPF. Її головна відмінність від Windows Forms в тому, що для відображення графічних компонентів використовується не GDI/GDI+, а DirectX – технологія, що розробляється Microsoft для створення ігор в тісній співпраці з виробниками відеокарт, і з цієї причини має підтримку апаратного прискорення [55].

Друга вирішальна особливість – декларативний підхід до створення інтерфейсу. Ще одне нововведення в WPF це мова розмітки XAML, що є підмножиною XML. Хоча весь призначений для користувача інтерфейс можна описати мовою C#, безпосередньо програмним шляхом, технологія WPF використовує інший підхід, завдяки якому інтерфейс описується мовою XAML, а робота з даними та бізнес-логіка визначається в коді.

Таке розділення обов'язків між елементами програми, та як наслідок між її розробниками – дизайнером та програмістом, дає привід говорити про використання MVVM – шаблону проєктування програми.

WPF також має і низку інших переваг, серед яких найбільш істотні:

- наявність опису стилів. В основу ідеї створення графічного інтересу покладена технологія HTML. Перехід до розробки на XAML дав можливість виводити властивості елементів вікон в окремі стилі;

- також присутні події, здатні визиватися за тих чи інших умов та реалізовувати певну функціональність. Незважаючи на те, що елементи в

основному були переписані, WPF зберіг принцип подій. Більш того було створено концепцію команди – дивлячись на прив'язку до даних, елемент керування ініціює ту чи іншу реакцію;

- прив'язка даних. Завдяки такій можливості можна не тільки взаємодіяти з даними ViewModel окремо від інтерфейсу, а й посилатися з самого інтерфейсу на інші елементи керування. Це зменшує кількість рядків коду та підвищує його читабельність;

- сторінки. Можна будувати додатки на основі сторінок та відповідних елементів навігації;

- модель малювання. Якщо в Windows Forms малювали пікселі, то в WPF працюють з примітивами – це вже готові базові фігури та інші графічні елементи. Крім того, є вбудована підтримка тривимірної графіки;

- анімація. У Windows Forms для того, щоб фігура чи елемент управління намалював сам себе, необхідно використовувати таймери. При розробці технології WPF було додано функціональність для створення анімації;

- незалежність від розширення екрану. Компонування елементів інтерфейсу, підлаштовуючи його під різні розширення екрану, стало значно гнучкішим у WPF;

- безпека. Поява мови XAML призвела до підвищення рівня відкритості та безпеки при роботі з даними;

- незалежність від розширення екрану. Компонування елементів інтерфейсу, підлаштовуючи його під різні розширення екрану, стало значно гнучкішим у WPF.

3.2 Структурна схема програми та її опис

Структурну схему програми, спроектовану для клієнтського додатку за шаблоном MVVM, продемонстровано на рисунку 3.3.

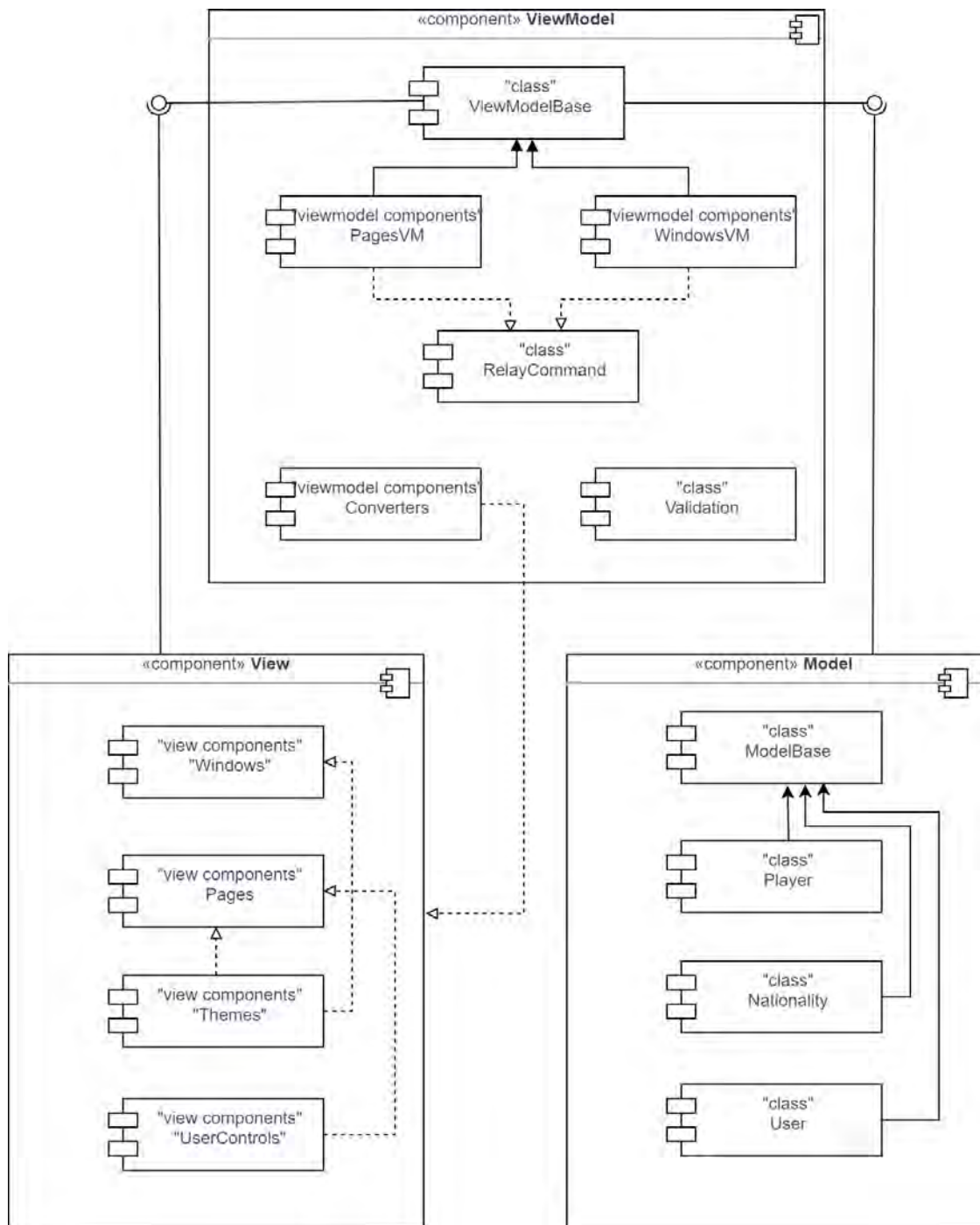


Рисунок 3.3 – Структурна схема програми, клієнтська частина

Опис структурних компонентів клієнтської частини додатку наведено у таблиці 3.4.

Таблиця 3.4 – Опис компонентів клієнтського додатку

Назва компоненту	Опис
1	2
View	Відповідає за графічне представлення та збір «сигналів» від користувачів.
Windows	В цей компонент мають входити класи, що описують вікна та інші об'єкти користувацького інтерфейсу, пов'язані з ними, за допомогою декларативного стилю, як прийнято за принципами WPF.
Pages	Описи сторінок інтерфейсу та відповідної ним навігації. Сторінки вбирають опис визначних користувацьких елементів, а змінювати сторінки можна за допомогою функціоналу навігації.
Themes	Описи стилів, іншими словами теми або ресурси, які призначаються об'єктам інтерфейсу.
UserControls	Компоненти з відповідними описами розмітки та прив'язки даних, які створені безпосередньо розробником. Тобто тут повинні бути визначені налаштовані елементи та описана логіка їх взаємодії.
ViewModel	ViewModel, або модель уявлення, пов'язує модель і уявлення через механізм прив'язки даних. Для цього можна створювати для елементів представлення їх власний ViewModel, який буде успадковувати клас ViewModelBase саме для прив'язки. Для організації команд треба створити клас RelayCommand.
Validation	Такий клас містить опис правил валідації, для відповідних полів ViewModel. Тут необхідно використати компонент DataAnnotations, для опису таких правил. Якщо виникне така ситуація, що введені

Продовження таблиці 3.4

1	2
	дані не відповідають шаблону, виникне виключення, яке буде оброблено, а інформація по ньому буде відправлена до компоненту View. Таким чином користувач отримає необхідну інформацію про формат робочого поля.
Converters	У разі, якщо необхідно до інтерфейсу прив'язати одразу декілька параметрів (до одного елементу керування), або якщо необхідно зробити перетворення типів прямо у розмітці, або виконати порівняння значень параметрів елементів представлення, використовують конвертери. Усі вони повинні бути описані у відповідному компоненті, та доступні у View.
Model	Тут необхідно описати базові сутності моделі – визначити їх ознаки та поведінку. До головних сутностей відносяться: «користувач» та його похідні, «гравець» – центральний клас програми, «національність» чи «команда» – допоміжні. Усі сутності компоненти Model успадковують ModelBase.

Структурну схему програми, спроектовану для серверного додатку, продемонстровано на рисунку 3.4.

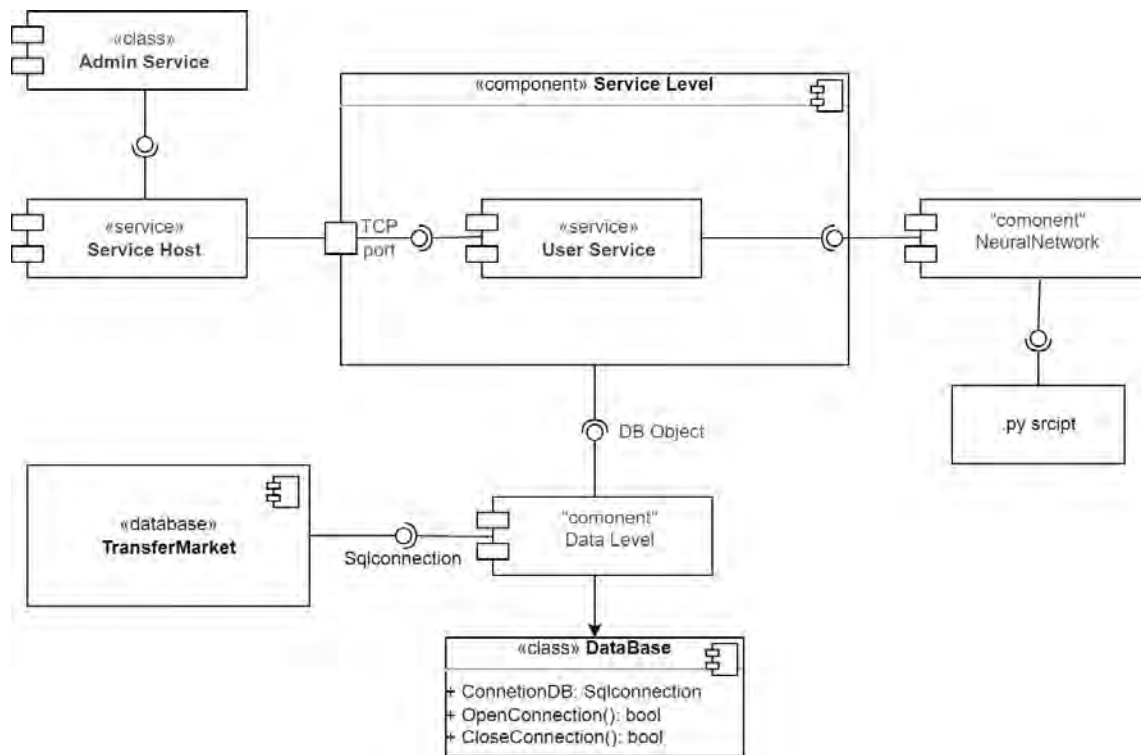


Рисунок 3.4 – Структурна схема програми, серверна частина

Опис структурних компонентів клієнтської частини додатку наведено у таблиці 3.5.

Таблиця 3.5 – Опис компонентів серверного додатку

Назва компоненту	Опис
1	2
TransferMarket	БД системи, створена за допомогою SQL Server.
DataBase	Підключення до БД.
DataLevel	Набір атрибутів та методів для роботи з даними, які зберігаються у БД.
NeuralNetwork	Компонент зберігає усі параметри та функції для інтегрування математичної моделі програми – моделі для прогнозування ціни. Тобто звідси викликається відповідний Python скрипт, подаються на обробку дані

Продовження таблиці 3.5

1	2
	– характеристики гравця, та зчитується результат роботи – прогнозована ціна.
UserService	Відповідає за розгортання сервісів, кожен з яких представлений у вигляді безперервно працюючої служби та має свою функціональність, якою може забезпечити клієнта. Кожен з сервісів реалізує деякий, визначений «контракт» та отримує власний порт. Сервіси зберігають логіку зміни даних у моделі. Звідси надається функціонал маніпулювання даними та функціональність роботи з нейронною моделлю.
ServiceHost	Відповідає за розгортання серверу, «збираючи» під своїм початком сервіси та надаючи клієнтам доступ до них.
AdminService	API, яким може користатися безпосередньо клієнт.

3.3 Проєктування структури бази даних

БД та її структура розроблені у відповідності до об'єктів предметної області. Концептуальну модель бази даних зображено на рисунку 3.5.

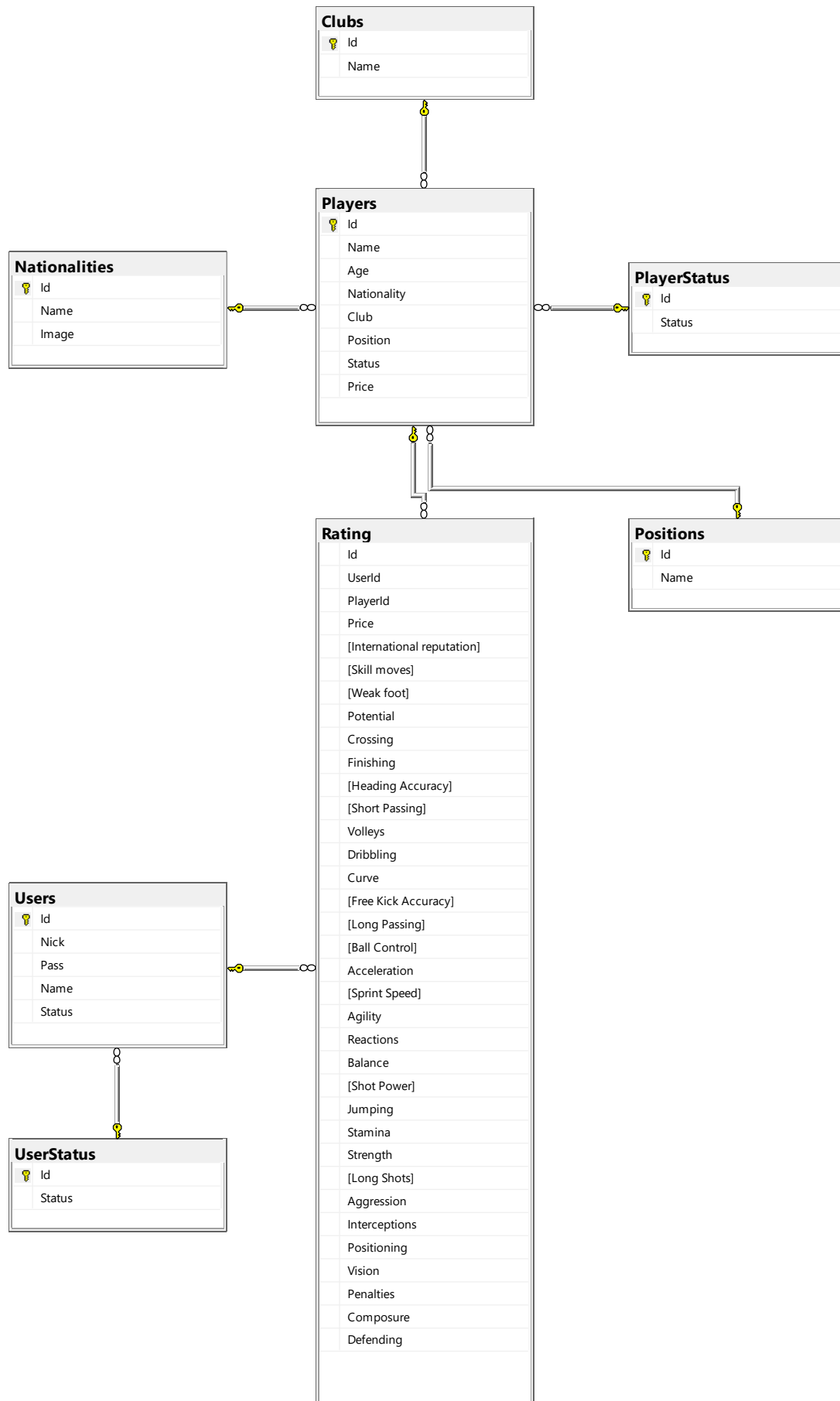


Рисунок 3.5 – Концептуальна схема БД програми

Опис таблиць, розроблених вході проєктування БД додатку, наведено у таблиці 3.6.

Таблиця 3.6 – Опис основних сутностей БД

Назва сутності	Опис
PlayerStatus	Зберігає трансферний статус футболіста: усі потенційні трансфери заблоковані клубом; навпаки, гравець виставлений на трансфер; ні те, не інше – гравець у «нормальному стані»; гравець закінчив кар’єру.
Clubs	Зберігає список усіх клубів.
Positions	Зберігає список усіх позицій на полі: воротар, захисник, півзахисник, нападник.
Nationalities	Зберігає список усіх національностей. Кожній нації повинно бути зіставлено зображення відповідної країни.
Players	Зберігає список усіх гравців.
Rating	Зберігає суб’єктивні оцінки, залишені користувачами, з огляду на гру відповідного футболіста. Також тут зберігається прогнозована, з огляду на оцінку, ринкова вартість гравця.
UserStatus	Зберігає тип зареєстрованого в системі користувача: або звичайний користувач, або адміністратор, або функціонер якоїсь команди.
Users	Зберігає інформацію про активних користувачів, зареєстрованих у системі, а також їх персональні дані.

3.4 Висновки за розділом 3

У розділі 3 було обґрунтовано вибір мови програмування – для побудови та навчання математичної моделі прийнято рішення застосувати Python, для створення клієнт-серверного додатку – застосувати C#.

У процесі проєктування було вирішено застосовувати технології Keras для машинного навчання та синтезу моделі, WPF – для створення інтерфейсу, шаблон MVVM для проєктування додатку в цілому, WCF – для розгортання сервісів та додержання клієнт серверної архітектури, SQL Server – для створення БД та запитів до неї. Робота має проходити у середовищі Spyder для розробки мовою Python та у середовищі MVS 2019 для розробки мовою C#.

Також було розроблено структурну схему програми та розглянуто обов'язки кожної з її компонент.

4 РОЗРОБКА ПРОГРАМИ

В даному розділі наведено визначні алгоритми функціонування програмного забезпечення та діаграми бізнес процесів, описано головні модулі реалізованої програми.

4.1 Алгоритми функціонування програмного забезпечення

Клієнтський додаток надає користувачу можливості досягти поставлених цілей, не вдаючись у програмні деталі. Основною такою ціллю, для чого власне і розроблювався програмний продукт, є визначення ринкової вартості гравця, в залежності від його ігрових параметрів.

Бізнес процес обчислення користувачем ціни на футболіста, починаючи від авторизації у системі і до самого отримання результату, зображено на рисунку 4.1.

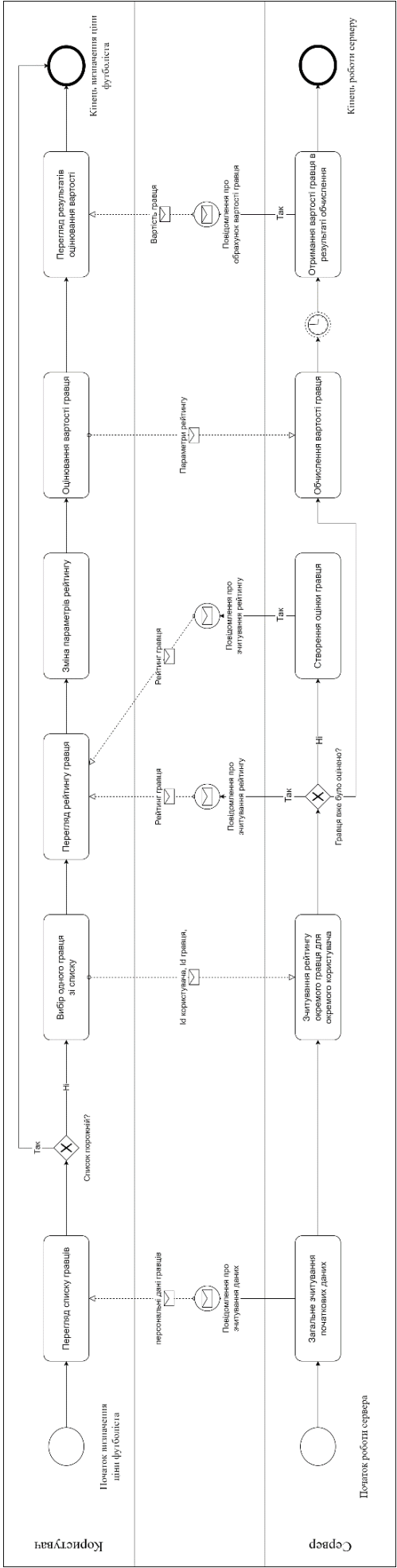


Рисунок 4.1 – Процес оцінювання вартості футболіста

Умовно бізнес процес можна поділити на три етапи, згідно до кількості серверних операцій у процесі:

- ініціалізація. Сервер вивантажує початкову інформацію з БД та, попередньо встановивши з'єднання з клієнтом, по відкритому порту передає інформації до клієнтського застосунку. Користувач у свою чергу отримає дані про гравців системи, перелік національностей та перелік клубів;

- вибір та оцінювання гравця. За допомогою інструментів пошуку користувач вибирає цільового гравця, якого бажає оцінити. Більш детальна інформація про гравця, точніше його рейтингові показники, вивантажується за допомогою запиту до сервера. У тому разі, якщо футболіста не було попередньо оцінено, значення кожного з параметрів дорівнює початковими значенням. Користувач у свою чергу працює з отриманими оцінками, змінюючи їх на свій розсуд;

- обчислення вартості. Коли стає необхідним підрахувати ціну футболіста потрібно теж викликати серверну операцію, передаючи сформований комплекс виставлених, необхідних оцінок. Усі обчислення вартості гравця відбуваються на сервері, після чого повертається фінальне значення ціни, яку бачить користувач на екрані.

Ще один процес наочно демонструє різницю у правах між адміністратором та користувачем. Перший має доступ до функціональності створення/видалення футболіста та редагування його персональної інформації, а другий – ні. Натомість користувач може дочекатися моменту, коли необхідний футболіст з'явиться в системі шляхом додавання адміністратором, та після цього оцінити здібності гравця.

Бізнес процес отримання доступу до інформації про нового футболіста, якого додав адміністратор, зображено на рисунку 4.2.

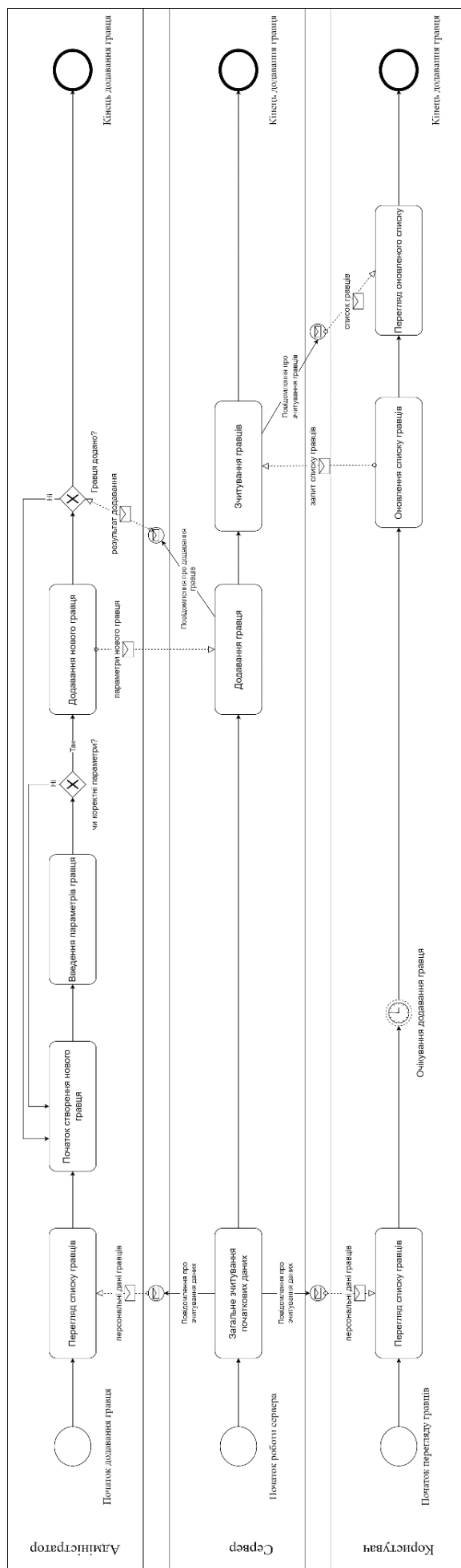


Рисунок 4.2 – Отримання доступу до інформації про нового футболіста, якого додав адміністратор

Однією з головних програмних функцій є редагування персональної інформації гравця, зміна її відповідно до введеної адміністратором інформації (звичайний користувач не має таких повноважень). Алгоритм прийняття та збереження змін щодо гравця зображено на рисунку 4.3.

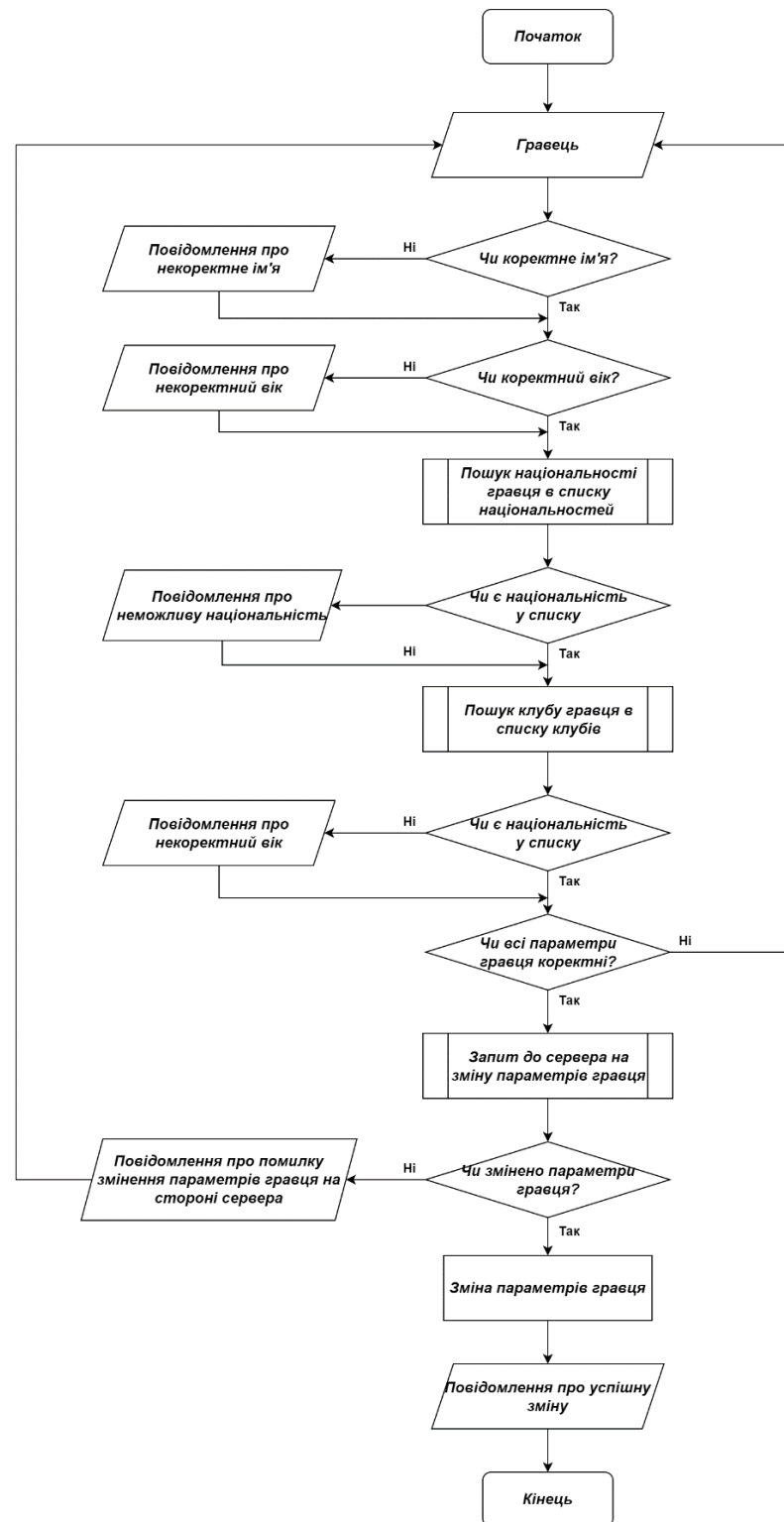


Рисунок 4.3 – Алгоритм зміни персональних параметрів гравця

4.2 Опис модулів реалізованої програми

4.2.1 Опис серверного додатку

Діаграма із ієрархію класів, що забезпечують функціонування серверної частини додатку, та деяких інших визначних структурних елементів системи, зображені на рисунку 4.4

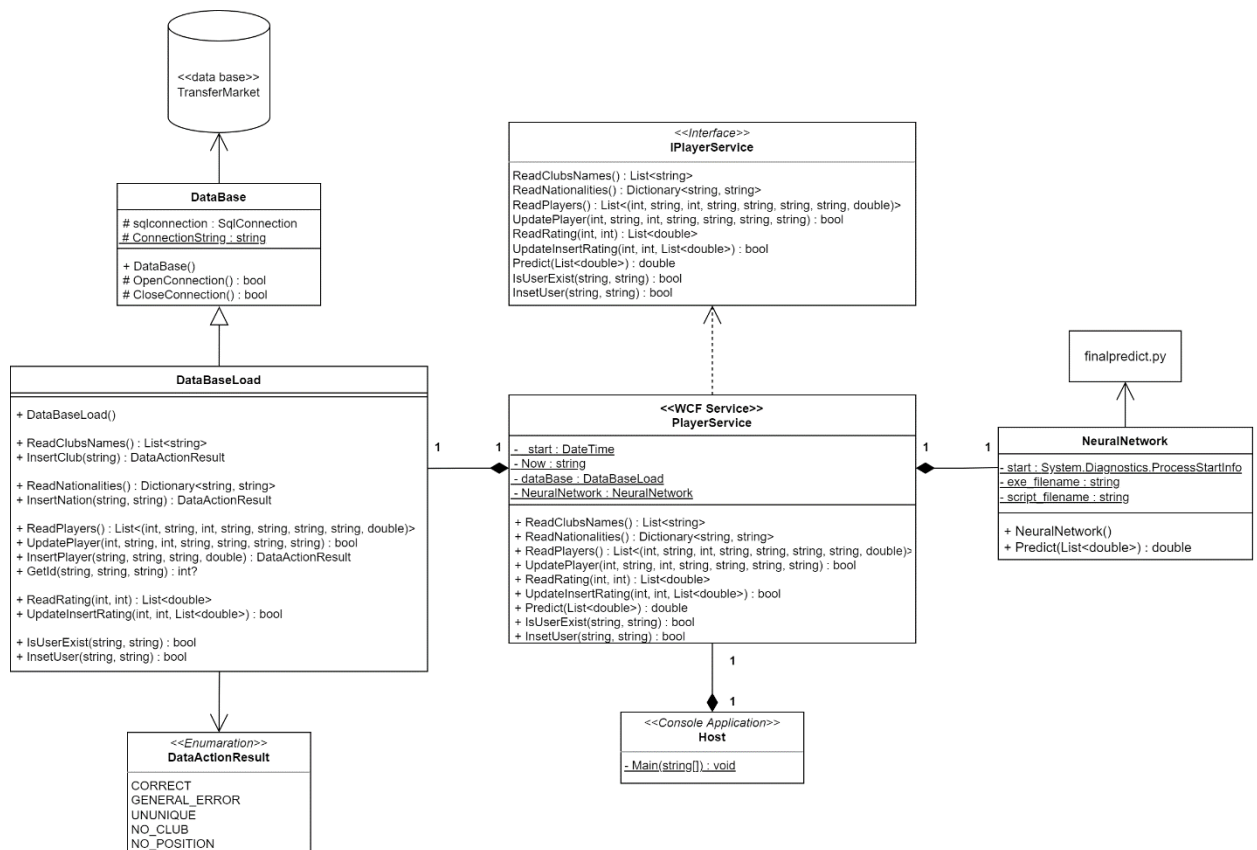


Рисунок 4.4 – Діаграма класів серверної компоненти додатку

Окремим структурним елементом є скрипт з назвою `finalpredict.py` – він відповідає за прогнозування ціни гравця та знаходиться у відповідному віртуальному середовищі `Pyhon`. Такий скрипт працює з моделлю нейронної мережі, створеною за допомогою `Keras` та збереженою з розширенням `.h5`.

Також окремо необхідно виділити БД, що зберігає усі дані про користувачів та дані, які користувачі модифікують. БД створена за допомогою `SQL Server` та зберігається у вигляді файлу з розширенням `.mdf`.

Ще одним важливим елементом структури є звичайний консольний додаток Host. Його задачею є розгортання на визначеному порту WCF сервісу PlayerService. Також, через командний рядок оператор, що має слідкувати за роботою сервера, може отримувати додаткову інформацію про його роботу.

Інші структурні елементи серверного додатку розширені по різних проєктах та представлені у вигляді класів.

4.2.1.1 Клас DataBase

Клас DataBase надає базову функціональність при роботі з БД. Опис полів та методів класу наведено у таблиці 4.1.

Таблиця 4.1 – Опис класу DataBase

Поля та методи класу	Опис
# sqlconnection : SqlConnection	Зберігає об'єкт підключення до БД.
# ConnectionString : string	Зберігає «рядок підключення», з відносним шляхом до БД та деякою сторонньою інформацією.
+ DataBase()	Конструктор, який ініціалізує об'єкт підключення.
# OpenConnection() : bool	Відкриває підключення до БД, змінюючи стан відповідного об'єкта, у разі якщо підключення закрите. Повертає результат відкриття.
# CloseConnection() : bool	Закриває підключення до БД, змінюючи стан відповідного об'єкта, у разі якщо підключення відкрите. Повертає результат закриття.

4.2.1.2 Клас DataBaseLoad

Клас DataBaseLoad успадковує клас DataBase та відповідає за виконання головних запитів до БД: зчитування, створення, видалення та зміна даних. Через те, що клас дуже місткий, було прийнято рішення оголосити його частковим та розподілити його на декілька файлів: PlayerData.cs, UserData.cs, PlayerRatingData.cs, NationalityData.cs, ClubData.cs.

Опис полів та методів класу DataBaseLoad наведено у таблиці 4.2.

Таблиця 4.2 – Опис класу DataBaseLoad

Поля та методи класу	Опис
1	2
+ DataBaseLoad()	Конструктор, який викликає базовий конструктор DataBase().
+ ReadClubsNames() : List<string>	Зчитує назви усіх клубів, занесених у базу даних.
+ InsertClub(string) : DataActionResult	Додає новий, унікальний клуб до БД. Повертає результат додавання.
+ReadNationalities(): Dictionary<string, string>	Зчитує національності гравців, які присутні у БД. Повертає словник, де ключ – це назва національності, а значення – зображення у вигляді прапора.
+InsertNation(string,string): DataActionResult	Додає дані про унікальну національність до БД.
+ ReadPlayers() : List<(int, string, int, string, string, string, string, double)>	Зчитує усіх футболістів, які присутні у БД. Повертає єдиний список, який складається з кортежів. Кожний кортеж містить інформацію про Id,

Продовження таблиці 4.2.

1	2
	ім'я, вік, національність, команду, позицію на полі та трансферний статус, а також вартість на трансферному ринку. Кожний елемент кортежу має відповідний тип даних.
+ UpdatePlayer(int, string, int, string, string, string, string) : bool	Оновлює дані про користувача за Id, повертає визначення чи було оновлення успішним.
+ InsertPlayer(string, string, string, double) : DataActionResult	Додає нового користувача до БД, з вказаними обов'язковими параметрами: ім'ям, позицією та командою. Четвертий параметр – ціна, визначена за замовчуванням як 0.0.
+ GetId(string, string, string) : int?	Повертає Id будь-якого рядка у будь-якій базі БД, якщо певне значення рядка відповідає порівнюваному. Функція приймає назву таблиці БД, назву стовпця таблиці та умову порівняння. У разі, якщо десь у функції відбувається викидання виключення, функція повертає null.
+ ReadRating(int, int) : List<double>	Зчитує оцінку рейтингу певного гравця та повертає список, відповідної структури. Приймає Id користувача та Id гравця, для якого проводиться пошук оцінки. Якщо оцінки ще не має у БД, повертається список із початковими значеннями параметрів.

Продовження таблиці 4.2

1	2
+UpdateInsertRating(int,int, List<double>) : bool	Або оновлює вже надану оцінку гравця користувачем, або додає таку оцінку у БД. Параметри функції відповідають ReadRating(). Повертає результат оновлення/додавання.
+ IsUserExist(string, string) : bool	Визначає, чи існує користувач у БД з заданими логіном та паролем, чи не існує.
+ InsetUser(string, string) : bool	Додає нового, унікального користувача до БД за логіном та паролем.

4.2.1.3 Перелік DataActionResult

Перелік DataActionResult визначає значення, які можуть повертатися у рамках операцій, пов'язаних із БД. Такі значення застосовуються для виведення більш докладної інформації про роботу сервера у хостдодатку.

Опис значень переліку DataActionResult наведено у таблиці 4.3.

Таблиця 4.3 – Опис переліку DataActionResult

Значення	Опис
1	2
CORRECT	Повертається у разі коректного відпрацювання функції.
GENERAL_ERROR	Повертається у випадку, коли функція викинула невизначене виключення.

Продовження таблиці 4.3

1	2
UNUNIQUE	Повертається у разі, якщо була спроба унікальному значенню таблиці присвоїти неунікальне значення.
NO_CLUB	Повертається у разі, якщо вказаний клуб відсутній у БД.
NO_POSITION	Повертається у разі, якщо вказана позиція гравця відсутня у БД.

4.2.1.4 Клас NeuralNetwork

Об'єкт класу NeuralNetwork забезпечує роботу із моделлю нейронної мережі для визначення трансферної вартості гравця, за його параметрами. Об'єкт класу тісно співпрацює зі скриптом finalpredict.py. Для цього програмно створюється ізольований, повноцінний процес (за допомогою простору імен System.Diagnostics), у якому працює Python інтерпретатор та виконує скрипт, з передаванням та поверненням відповідних параметрів. Передача параметрів відбувається на основі головного потоку, як процесу виконання програми на Python, так і додатку C#.

Опис полів та методів класу NeuralNetwork наведено у таблиці 4.4.

Таблиця 4.4 – Опис класу NeuralNetwork

Значення	Опис
1	2
- start : <u>System.Diagnostics.ProcessStartInfo</u>	Поле необхідно для відокремлення налаштування параметрів запуску процесу від самого запуску.

Продовження таблиці 4.4

1	2
- exe_filename : string	Зберігає шлях до інтерпретатору Python.
- script_filename : string	Зберігає шлях до скрипту Python, який необхідно виконати.
+ NeuralNetwork()	Конструктор, у якому ініціалізуються параметри поля start: задається файл до виконання; виконання не буде відбуватися через стандарту функцію Windows ShellExecute(), а через, власне, інтерпретатор Python; для процесу не буде створено нове вікно; результат роботи процесу та інформація про виникненні помилки будуть доступні для зчитування.
+ Predict(List<double>) : double	Обчислює вартість гравця за заданими параметрами. Приймає список ігрових параметрів гравця. Формує єдиний рядок. Передає такий рядок як аргумент для майбутнього процесу. Розгортає процес – запускає через інтерпретатор Python скрипт finalpredict.py, передаючи у головний потік рядок з параметрами гравця. Скрипт зчитує рядок, виконує обчислення та подає на вихід інформацію про результат обчислення та помилки. Рядок результату – ціна гравця, зчитується з потоку та повертається. Також можливе читання потоку помилок.

4.2.1.5 Клас **PlayerService**

Клас **PlayerService** реалізує інтерфейс **IPlayerService**.

Подібна елементарна структура описує сервер з одним розгорнутим сервісом. Для того щоб у майбутньому клієнт зміг проводити обмін даними із сервером, вводиться поняття контракту.

Контракт даних — це офіційна угода між службою та клієнтом, яка абстрактно описує дані, що підлягають обміну. Тобто для зв'язку клієнт і сервіс не повинні спільно використовувати одні й ті самі типи, тільки одні й самі контракти даних [56].

Методи, доступ до яких буде надано користувачу, позначаються атрибутом **OperatonContract**. Для серіалізації об'єктів, програмно визначених класів, використовується атрибут **DataContract**. Через те, що серверні функції повертають або об'єкти базових типів, або об'єкти класів з реалізуємо серіалізацією [57], останній атрибут не було використано.

Сам же клас **PlayerService**, окрім розгортання функцій по роботі з БД на сервісі, проводить обчислення вартості гравця за допомогою методу класу **NeuralNetwork**, та надає доступ клієнту для проведення такого обчислення.

Також клас **PlayerService** утворює оболонку, та дозволяє оператору контролювати роботу сервісу – у консоль виводяться результати роботи функцій з даними, результати обчислення, допоміжна та часова інформація.

Опис полів та методів класу **PlayerService** наведено у таблиці 4.5.

Таблиця 4.5 – Опис класу **PlayerService**

Значення	Опис
1	2
<u>- _start : DateTime</u>	Зберігає дату початку роботи сервісу, починаючи від першого звернення.

Продовження таблиці 4.5

1	2
<u>- Now : string</u>	Зберігає час, відносно значення _start.
<u>- DataBase : DataBaseLoad</u>	Зберігає об'єкт DataBaseLoad для роботи з БД.
<u>- NeuralNetwork : NeuralNetwork</u>	Зберігає об'єкт NeuralNetwork для роботи з НМ.
+ ReadClubsNames() : List<string>	Відповідає однойменній функції класу DataBaseLoad. Виводить у консоль кількість зчитаних команд, час, результат роботи.
+ReadNationalities(): Dictionary<string, string>	Відповідає однойменній функції класу DataBaseLoad. Виводить у консоль кількість національностей гравців, час, результат роботи.
+ ReadPlayers() : List<(int, string, int, string, string, string, string, double)>	Відповідає однойменній функції класу DataBaseLoad. Виводить у консоль кількість зчитаних гравців, час, результат роботи.
+ UpdatePlayer(int, string, int, string, string, string, string) : bool	Відповідає однойменній функції класу DataBaseLoad. Виводить у консоль Id гравця, час, результат роботи.
+ ReadRating(int, int) : List<double>	Відповідає однойменній функції класу DataBaseLoad. Виводить у консоль Id гравця, Id користувача та час.
+ UpdateInsertRating(int, int, List<double>) : bool	Відповідає однойменній функції класу DataBaseLoad. Виводить у

Продовження таблиці 4.5

1	2
	консоль Id гравця, Id користувача, час, результат роботи.
+ Predict(List<double>) : double	Відповідає однойменній функції класу DataBaseLoad. Виводить у консоль обчислене значення ціни гравця, час, результат роботи.
+ IsUserExist(string, string) : bool	Відповідає однойменній функції класу DataBaseLoad. Виводить у консоль логін та пароль гравця, час, результат роботи.
+ InsetUser(string, string) : bool	Відповідає однойменній функції класу DataBaseLoad. Виводить у консоль логін та пароль гравця, час, результат роботи.

4.2.2 Опис ViewModel компоненти клієнтського додатку

Компонента ViewModel забезпечує зв'язок інтерфейсу користувача та моделі та має доволі розгалужену структуру. Ієрархію класів ViewModel представлено у вигляді діаграми класів та зображено на рисунку 4.5.

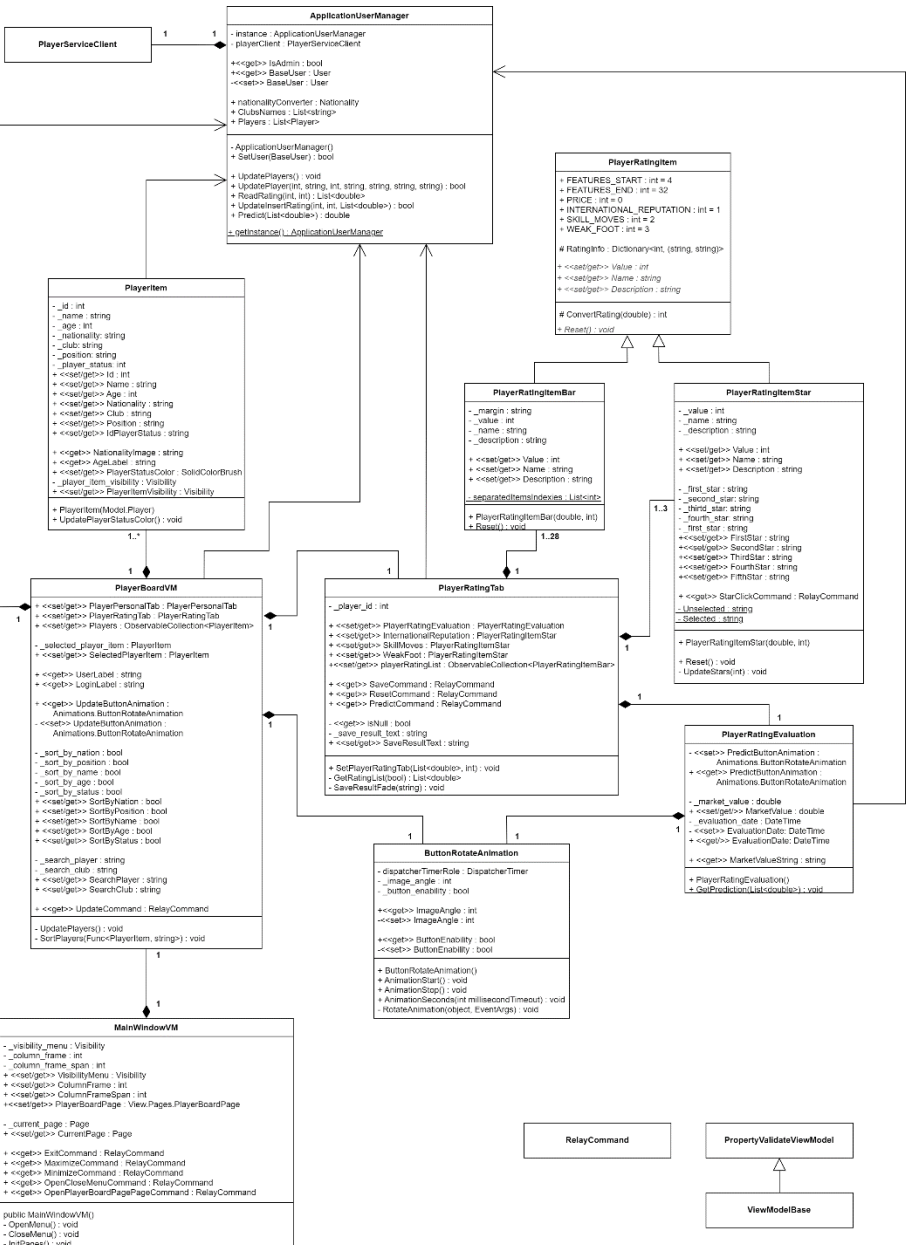


Рисунок 4.5 – Діаграма класів ViewModel компоненти додатку

Додатково необхідно зазначити, що клас `ApplicationUserManager` забезпечує суто роботу з даними та відноситься до `Model` компоненти. Тим не менш, його зображення на загальній діаграмі є доволі виправданим, через його зв'язок з більшістю класів `ViewModel`.

Також до Model відносяться класи, які описують відповідні атрибути, відповідальні за валідацію моделей. Валідація відбувається на базі функціональності з простору імен DataAnnotations.

4.3 Висновки за розділом 4

У розділі 4 було наведено діаграми бізнес процесів для визначених операцій, які можуть виконувати користувачі, та алгоритмів функціонування програми. Окрім, принципів оперування даними було наведено опис структури їх взаємодії, шляхом опису діаграм класів як серверного, так і користувацького додатку.

5 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

В даному розділі описано призначення програми для потреб різних груп користувачів, наведені необхідні умови для коректної роботи програми, побудовано та описано сценарії роботи з додатком. Також було проведено тестування функціональних характеристик програми.

5.1 Призначення програми

В загальному вигляді розроблена програма призначена для

- ведення обліку футболістів;
- виставлення рейтингових оцінок здібностям гравців;
- обчислення ринкової вартості футболістів.

Якщо контролювати інформацію про гравців в системі може лише адміністратор, то до двох інших функціональних можливостей може долучитися будь-який бажаючий.

Оцінювати гру футболістів та, на базі цієї оцінки, обчислювати ціну на трансферному ринку – така функціональність може знадобитися доволі широкому колу осіб. Різними групами користувачів програма може бути застосована для різних цілей:

– для футбольних функціонерів визначити ціну на основі рейтингових показників якостей гравця – головна прерогатива та аргумент при оформленні трансферного договору про купівлю-продаж футболіста. Тут має місце наступний рекомендований алгоритм роботи з програмою: штаб тренера команди аналізує гру футболіста (як об'єктивні і суб'єктивні показники, так і психологічні, соціальні, персональні характеристики), передаючи інформацію головному тренеру. Останній виступає у ролі експерта зі знань (п. 1.2.1) та визначає фінальну оцінку, вносячи її у програму. Отриманий результат обчислення (ціна гравця) має бути використаний на переговорах по трансферу бажаного футболіста;

– для букмекерів та для сфери ставок в цілому, яка в останні роки займає велику долю ринку та навіть впливає на функціонування футбольних клубів, система може розраховувати ціну під кутом створення пропозицій клієнтам для ставки. Якщо букмекер найточнішим чином виставить оцінки гравцям, він зможе на основі такого аналізу пропонувати своїм клієнтам зробити ставку про те, наприклад, дорожче чи дешевше заданої вартості буде оформлена угода про перехід;

– для незалежних футбольних експертів та звичайних користувачів функціональність визначення ціни в грошовому еквіваленті може не бути вирішальним. У такому разі їм буде достатньо вести звичайний облік змін характеристик гравця з часом. Аналітикам – необхідно для порівняння, створення дискусії, вболівальникам – спосіб проведення дозвілля.

5.2 Умови виконання програми

Звичайно, програма вимагає від користувача роботи під операційною системою Windows. Сама розробка проводилась на корпоративній, 64-розрядній операційній системі Windows 10, версії 21H2.

Умови виконання програми можна виділити наступні: додержання файлової структури, наявність у збірці відповідних залежностей, у тому числі посилання на файли проєктів, та деяких додаткових пакетів, додержання мінімальних апаратних вимог. Все це необхідно визначити як для серверного додатка, так і для клієнтського.

5.2.1 Файлова структура

Серверна частина додатку, для коректної роботи, повинна зберігати наступні файли:

– TransferMarket.mdf – БД;

- `dataset.csv` – зберігає таблицю з персональною інформацією по кожному футболісту, включаючи ігрові показники та вихідну ринкову вартість. Дані можуть бути перенесені у БД;

- `nationalities.csv` – зберігає інформацію про національності, тобто їх назву та шлях до відповідного зображення. Дані можуть бути перенесені до БД;

- `Host.exe` – виконуваний файл для початку роботи сервера.

Тут же необхідно зазначити файли, які використовуються для аналізу даних та навчання моделі:

- `dataset.csv` – фінальний датасет на якому навчається НМ;

- `FINAL.h5` – фінальна модель для визначення вартості гравця на трансферному ринку за рейтинговими показниками;

- `source.csv` – початково сформований, необроблений датасет, дані якого завантажені з офіційного сайту гри Fifa;

- `errors.txt` – посилання на профілі футболістів, дані про яких не було зчитано коректно;

- `players.txt` – перелік посилань на профілі гравців гри FIFA, для витягування їх персональних даних.

Дерево ієрархії файлів, відносно розташування яких у файловій структурі проєкту явно визначене, та які використовує сервер, зображено на рисунку 5.1.

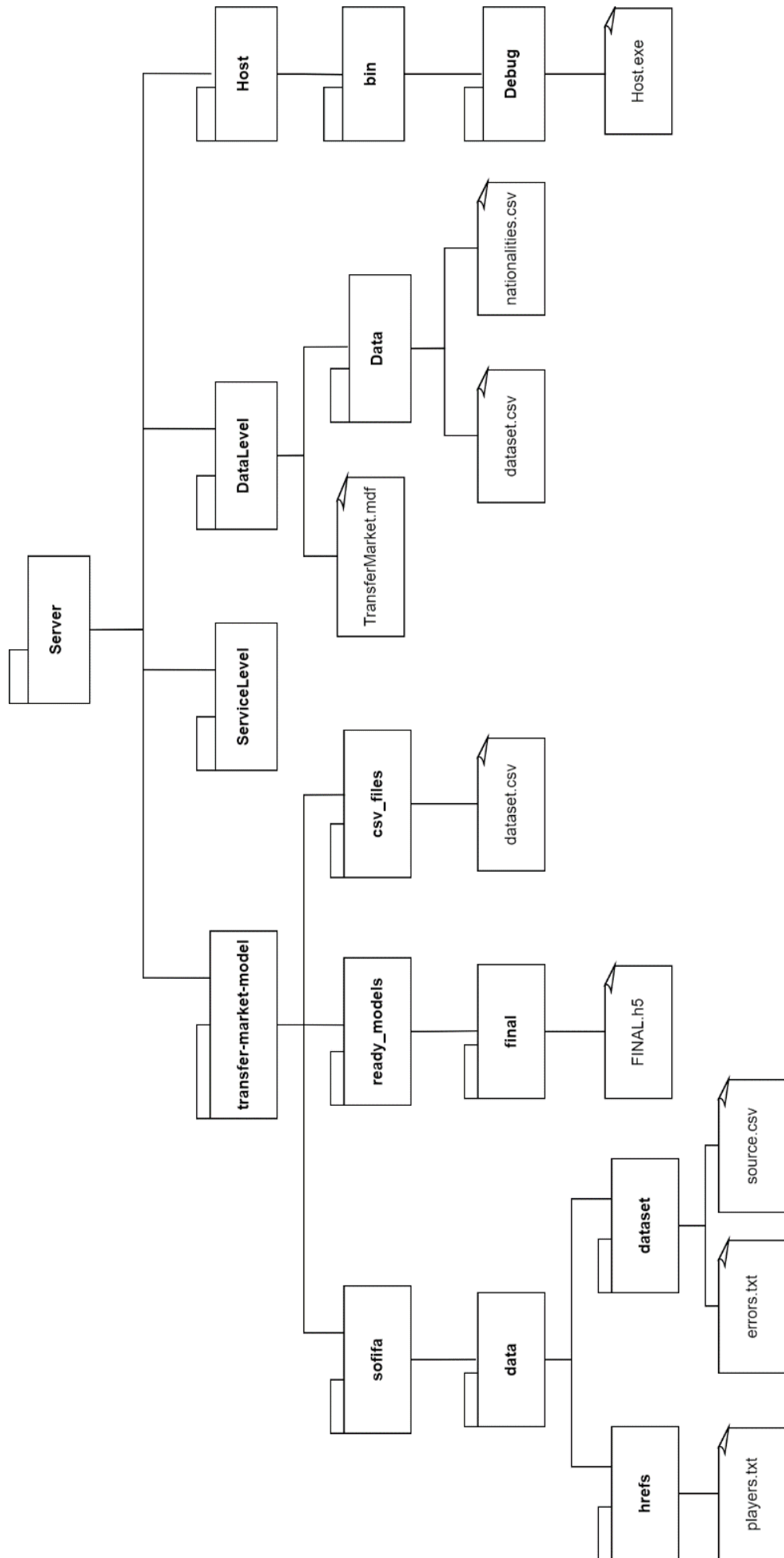


Рисунок 5.1 – Файловая структура серверного застосунку

Клієнтська частина додатку хоч і використовує більше файлів, які можна назвати «зовнішній ресурс», однак це здебільшого зображення чи шрифти, що використовуються на рівні представлення.

5.2.2 Наявність зовнішніх залежностей

Для успішної збірки проєктів та відповідно запуску програми, необхідна відповідна структура посилань та деякі додаткові пакети. Діаграма залежностей та асоціацій програми зображена на рисунку 5.2.

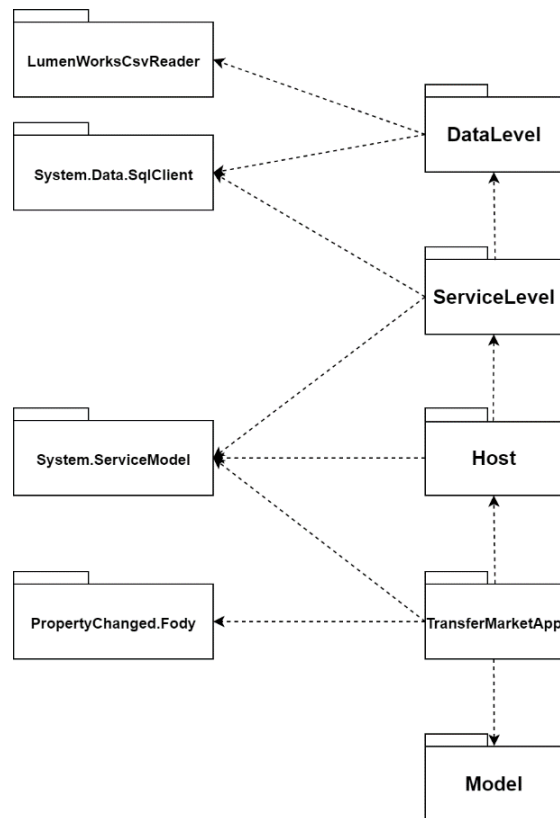


Рисунок 5.2 – Структура залежностей

Область відповідальності зовнішніх пакетів наведено у таблиці 5.1.

Таблиця 5.1 – Опис зовнішніх пакетів

Назва	В.	Посилаються	Область відповідальності
LumenWorksCsvReader	4.0.0	DataLevel	Використовується для роботи з файлами розширення .csv. Зокрема для початкового наповнення БД даним про футболістів.
System.Data.SqlClient	4.8.3	DataLevel, ServiceLevel	Використовується для роботи з БД.
System.ServiceModel	3.4.0	ServiceLevel, Host, TransferMarketApp	Використовується для створення клієнт-серверної архітектури, надаючи можливість описувати сервіси: визначати інтерфейси підключення та проводити серіалізацію даних. Надається як частина пакета WCF [57].
PropertyChanged.Fody	3.4.1	TransferMarketApp	Впроваджує код, який викликає подію PropertyChanged, у властивості класів, які реалізують INotifyPropertyChanged. Може бути автоматично реалізований для певного класу за допомогою атрибута. Використовується для визначення умов валідації властивостей [58].

Окремо слід визначити, що віртуальне середовище Python, яке створює умови для виконання скриптів для визначення ціни гравця, теж повинно зберігати певні бібліотеки, до них треба віднести:

- os, для роботи з файлами;
- sys, для роботи з функціями та змінними, що взаємодіють з Python інтерпретатором;
- numpy для роботи з масивами та матрицями;

– keras для завантаження побудованої моделі та виконання функції прогнозування за вхідним вектором ознак футболіста.

5.2.3 Мінімальні апаратні вимоги

Мінімальні апаратні вимоги для коректної роботи клієнтського застосунку перш за все визначаються ступенем навантаження на графічний процесор. Як вже було сказано у пункті 3.1.4, використання WPF передбачає використання технології DirectX, для відображення візуальних елементів. Тому відеокарта машини, на якій запускається клієнтський додаток, повинна підтримувати технологію DirectX. Загалом же клієнтський додаток повинен задовольняти наступним мінімальним апаратним вимогам:

- процесор: Intel или AMD від 1 гігагерц (ГГц) або швидший;
- відеокарта: з підтримкою DirectX 10;
- оперативна пам'ять: від 2 Гб;
- місце на жорсткому диску під програму: 50 Мб;
- розширення екрану: 576 x 1024.

Серверний додаток не виконує операції з візуальними компонентами, тому графічні вимоги його не стосуються. Натомість, він відповідальний за обробку даних, розгортаючи службу, яка у фоновому режимі прослуховує певний порт на наявність підключень, служби SQL Server, та вікно консолі для перегляду інформації про роботу серверу. Звичайно, більше за все ресурсів споживає інший процес, той, який забезпечує виконання Python скрипта, для обчислення ринкової вартості гравця. З цією ціллю необхідно спочатку запуснути інтерпретатор, який виконує потрібний скрипт. У тому випадку, якщо якась умовно велика кількість користувачів одномоментно спробує обрахувати ціну футболіста, для цього серверу знадобиться багато системних ресурсів. А тому і кількість оперативної пам'яті чітко визначити неможливо, однак можна констатувати, що вона лінійно залежить від кількості користувачів.

Беручи до уваги факт розгортання сервера на звичайному ПК, можна назвати цифру 8 Гб обов'язкової оперативної пам'яті.

5.3 Виконання програми

Виконання програми починається з запуску серверного застосунку, тобто необхідно виконати файл Host.exe від імені адміністратора. Оператор побачить відкритий командний рядок, з інформацією про стан розгорнутого сервісу та номер порту що прослуховується. Початок роботи сервера зображено на рисунку 5.3.

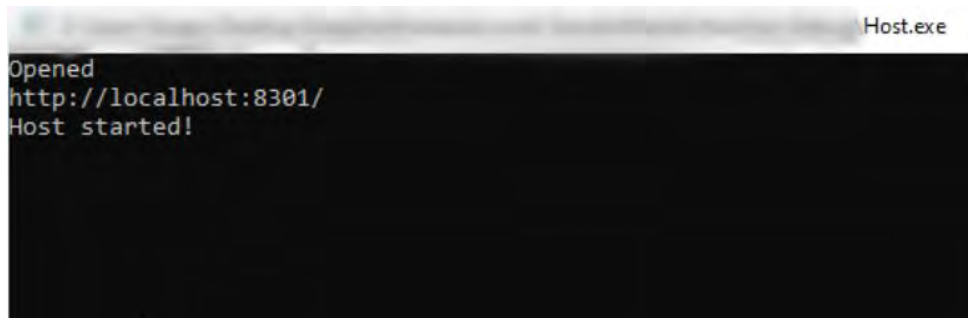


Рисунок 5.3 – Початок роботи сервера

Тепер до сервера можуть під'єднуватися користувачі: або методом реєстрації у системі, або авторизуючись – для цього необхідно натиснути відповідну кнопку початкового вікна. Адміністратор, на відміну від звичайного користувача, не може реєструватися, його дані мають бути занесені до БД напряму. Процес авторизації у системі, як частина користувацького сценарію, зображено на рисунку 5.4.



Рисунок 5.4 – Початкове вікно та вікно авторизації

Після авторизації у системі відкривається головне вікно, із розгорнутою основною сторінкою, або, в контексті програми, дошкою. Тут варто відмітити, що зліва вікна є «навігаційна» панель, вона, окрім вказання на головну сторінку, дає можливість змінити користувачу особисту інформацію – логін та пароль. Процес редагування профілю відбувається у тому ж вікні, що й авторизація, як зображено на рисунку 5.4, тільки відкритому з іншими параметрами.

Головне ж вікно програми, у якому користувач проводить майже весь час при роботі зі застосунком, зображено на рисунку 5.5.



Рисунок 5.5 – Головне вікно програми

Верхня панель вікна містить кнопки відкриття/закриття меню, кнопку згортання вікна до панелі завдань, кнопку розгортання на весь екран та кнопку закриття вікна – повного виходу з програми. Також реалізовано функціональність виходу з повноекранного режиму та автоматичний перехід до віконного, у разі якщо користувач натисне на панель.

Центральне місце головної сторінки, як і всієї програми, займає таблиця з переліком усіх наявних футболістів у системі. Як і належить, кожний рядок таблиці – окремий футболіст, із ім'ям, віком, національністю, командою, за яку виступає, позицією та трансферним статусом.

Сортування таблиці гравців за необхідним критерієм можна здійснювати, натиснувши на відповідний критерій на панелі зверху.

Наступна панель містить текстові поля для пошуку гравців у таблиці за ім'ям та клубом (обоє чинників враховуються при відборі). На випадок, коли необхідно оновити інформацію таблиці (наприклад якщо вона змінена адміністратором), є відповідна кнопка зліва. Далі є можливість додавати користувачів до обліку, або навпаки видаляти. Ці функції доступні тільки адміністратору, для звичайного користувача такі кнопки невидимі.

Наступна панель несе суто інформаційне наповнення – напис із правами авторизованого користувача та під яким логіном він зайшов у систему.

Інформаційна панель праворуч містить вкладку редагування інформації про гравця. Звісно, така функція теж доступна лише адміністратору. Користувацький сценарій зміни інформації про футболіста зображено на рисунку 5.6.

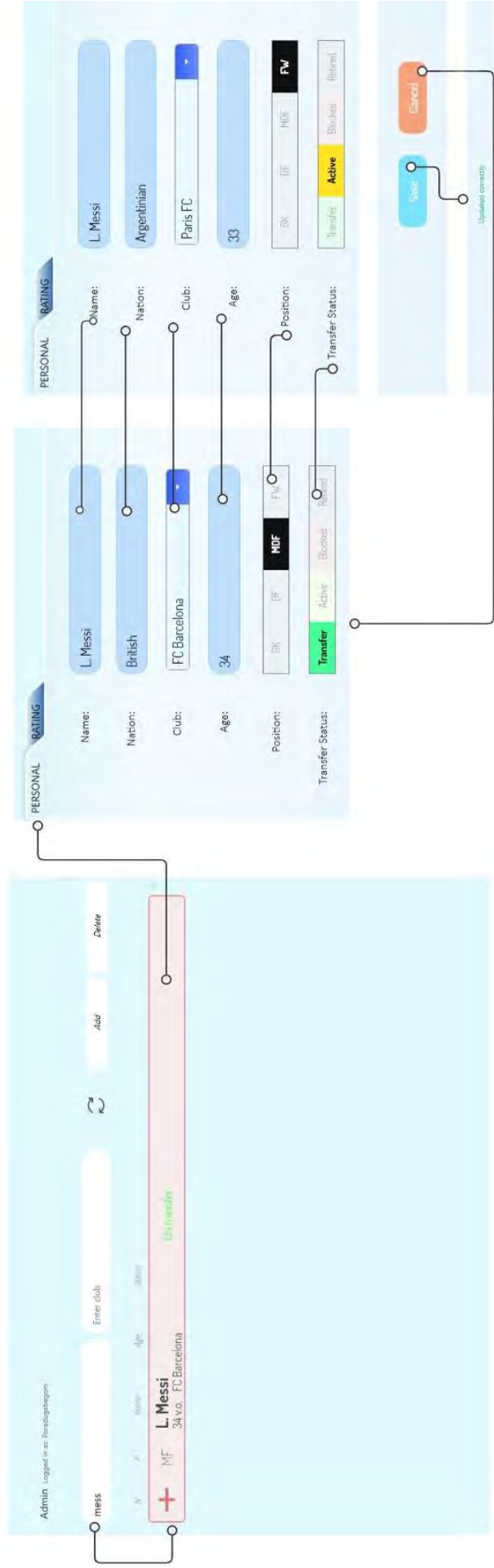


Рисунок 5.6 – Сценарій редагування інформації про гравця

Для зміни інформації про гравця необхідно спочатку вибрати цільового гравця у загальній таблиці, можливо застосовуючи інструменти пришвидшення пошуку. Після натискання на відповідного футболіста, інформаційна панель наповниться його особистими даним. Її контент можна змінювати на власний розсуд: тестові дані за допомогою текстових полів, категоріальні дані, наприклад команду, за допомогою спадаючого списку, або за допомогою перемикача.

Також зміна особистої інформації гравця у системі несе певний предметний сенс, наприклад:

- зміна імені – відбувається не часто та відповідає змінам у паспорті гравця, однак не завжди. Інколи гравці, позбуваючись довгого імені, беруть собі псевдонім;

- зміна національності – теж не часто відбувається, але можливі ситуації, коли футболіст змінює громадянство та виступає за іншу національну збірну;

- зміна команди – типова ситуація після проведення трансферу;

- зміна віку;

- зміна позиції на полі. Позиція – доволі умовний показник, однак існує безліч прикладів, коли гравець виступає буквально на всіх позиціях походу кар'єри, окрім мабуть голкіперів;

- зміна трансферного статусу – ідентифікується футбольними функціонерами: чи може футболіст бути купленим, чи не заблокований його статус, чи є він виставленим на трансфер та чи не закінчив кар'єру.

Будь-які зміни проходять перший етап валідації даних – клієнтський, коли користувач ще при редагуванні полів отримує результат перевірки формату введених даних. Другий етап валідації проходить вже на сервері, при спробі додати дані до БД, у разі успіху чи невдачі користувач отримає відповідне повідомлення.

Тим не менш, усі зміни інформації про гравця мають бути збережені, натисканням кнопки «Save», або навпаки відмінені, натисканням «Cancel».

Головною ж інформаційною панеллю є редагування рейтингу гравця, окремо взятим користувачем. Сценарій зміни оцінок футболіста та підрахунок вартості футболіста на їх основі зображено на рисунку 5.7.

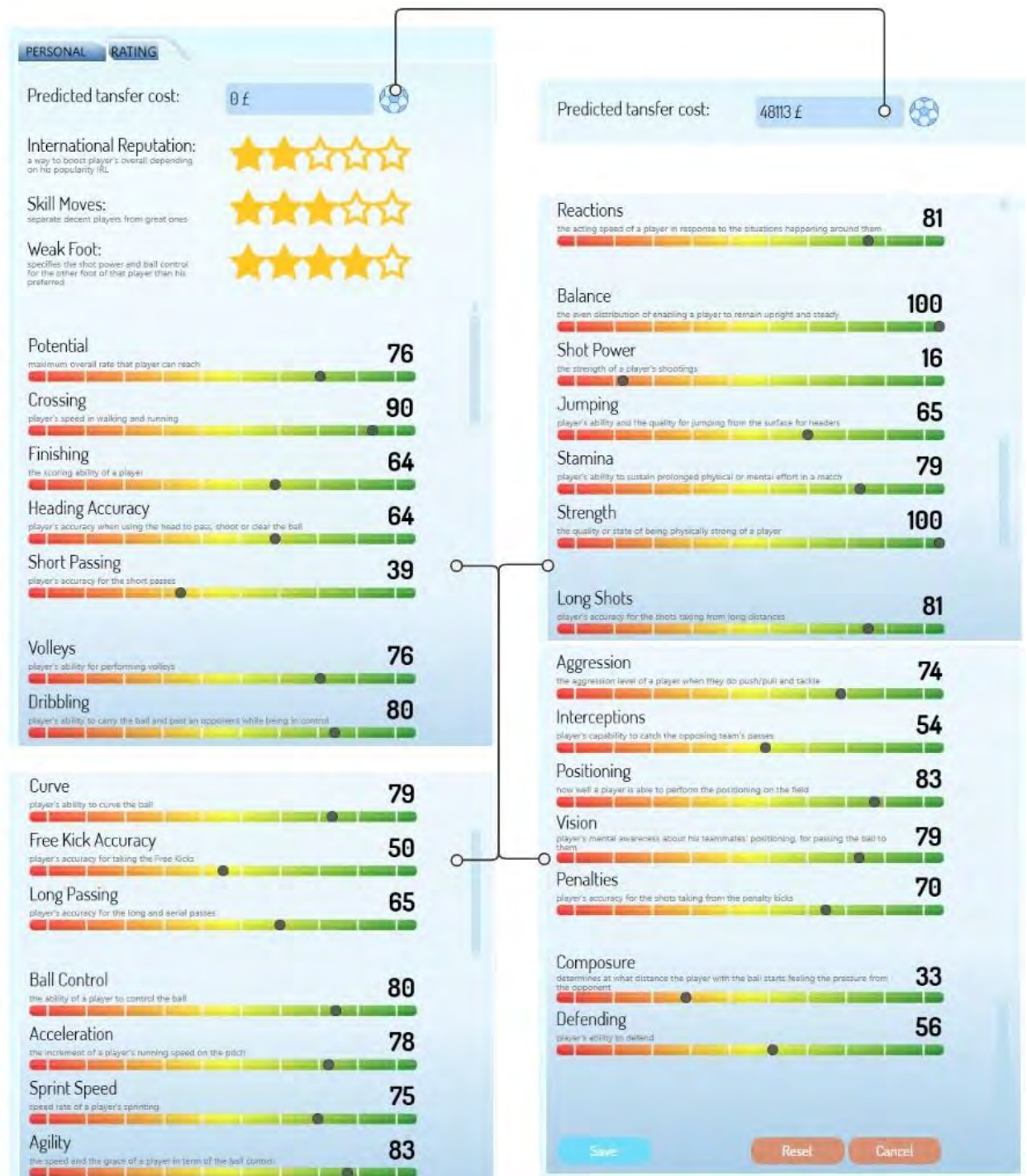


Рисунок 5.7 – Сценарій оцінювання гравця

Оцінка надається за певними, заздалегідь визначеними властивостями, користувачу також надається короткий опис кожної з них; всього двадцять вісім таких властивостей для оцінки від 0 до 100, та три для оцінки від 1 до 5.

Оцінка повинна бути збережена, за допомогою кнопки «Save». Відміна редагованих даних відбувається після натискання кнопки «Cancel» (тільки на стороні клієнта), а «обнуління» характеристик відбувається після натискання «Reset».

Центральним же епізодом сценарію оцінювання є підрахунок ціни гравця. Якщо користувач виставив необхідні оцінки гравцю, він повинен натиснути відповідну кнопку у вигляді футбольного м'яча. На стороні сервера, використовуючи регресійну модель, така вартість буде обчислена та повернена користувачу. Таким чином, на основі проведеного аналізу ринкової вартості футболіста, користувачем може прийматися рішення з приводу трансферної активності.

Кожен користувач, виставивши свої унікальні оцінки параметрам гравця, отримує відповідну ціну на футболіста. Таким чином, функціонер від клубу може визначити параметри гравця, отримати його трансферну вартість та запитувати відповідну ціну на трансферному ринку. Інша сторона потенційного договору, при наявності доступу до розробленого програмного забезпечення, може зробити теж саме, однак отримуючи іншу вартість, через те, що і оцінки інші. Отже, клуби, а точніше їх представники, можуть проводити перемовини на основі не просто суб'єктивних експертних оцінок, а на базі такої оцінки, що вбирає велику кількість попередніх. Та сторона, що має доступ до ПЗ, може мати і деяку перевагу на двосторонніх перемовинах по трансферу.

Усі виставлені користувачами оцінки зберігаються у БД. Кожен користувач може додати до історії рейтингів лише 1 оцінку за день – останню збережену.

Що ж до серверу, то після роботи з клієнтом командний рядок виводить усю супровідну інформації, як зображено на рисунку 5.8.

```

Opened
http://localhost:8301/
Host started!
Nationalities read: 62 | Time: -00:00:00.7985617
Clubs read: 730 | Time: -00:00:00.8516103
Players read: 100 | Time: -00:00:00.9186704
Players read: 100 | Time: -02:43:50.0545047
Read rating UserId = 6 PlayerId = 90069 | Time: -02:43:56.5129180
Update PlayerId = 90069 | Time: -02:44:15.4845506 | Result: Correct
Update/Insert rating UserId = 6 PlayerId = 90069 | Time: -02:44:22.5747259 | Result: Correct
Read rating UserId = 6 PlayerId = 89998 | Time: -02:44:27.7300988
Update/Insert rating UserId = 6 PlayerId = 89998 | Time: -02:44:30.3254812 | Result: Correct
Predicted Market Value = 18112 | Time: -02:44:40.4140397 | Result: Correct
Update/Insert rating UserId = 6 PlayerId = 89998 | Time: -02:44:41.9788679 | Result: Correct
Update/Insert rating UserId = 6 PlayerId = 89998 | Time: -02:44:45.0504457 | Result: Correct
Predicted Market Value = 18535 | Time: -02:44:52.4749624 | Result: Correct

```

Рисунок 5.8 – Інформація про роботу сервера з клієнтом

5.4 Тестування програми

При оцінюванні якості програмного забезпечення необхідно провести тестування на відповідність функціональним вимогам, тим, які було наведено у пункті 3.1.1. Використаємо як негативні, так і позитивні тестові сценарії.

5.4.1 Валідація параметрів

Додавання гравця з певними параметрами та редагування гравця, змінюючи значення цих параметрів, – головні функції адміністратора. Валідація параметрів гравця відбувається на стороні клієнту. Тестування полів гравця наведено у таблиці 5.2.

Таблиця 5.2 – Валідація параметрів гравця

№	Поле	Значення	Очікуване повідомлення	+/-
1	2	3	4	5
1	Name	Порожній рядок	Name is required	+
2	Name	«Ювентус»	First symbol should be alphabetic	+
3	Name	«#d»	First symbol should be alphabetic	+
4	Name	«s»	Name should contain minimum 2 alphabetic	+

Продовження таблиці 5.2

1	2	3	4	5
5	Name	«S#23131»	Name should contain minimum 2 alphabetic	+
6	Name	«Ss123»	Wrong character	+
7	Name	«Messi%»	Wrong character	+
8	Name	«\$Me-ssi»	Wrong character	+
9	Name	«Mess_i»	Wrong character	+
10	Name	«messimessimessime ssimessimessimessim essimessimessimessi messi»	Name character limit exceeded. Max length is 50	-
11	Name	«Jo»	-	-
12	Name	«jo»	-	-
13	Name	«Ronaldo»	-	+
14	Name	«C. Ronaldo»	-	+
15	Name	«t. alexander-arnold»	-	+
17	Name	«Pie'r-Omeric Abumeyang»	-	+
18	Nationality	Порожній рядок	Nationality is required	+
19	Nationality	«Sometext»	Such nationality does not exist	+
20	Nationality	«French»	Such nationality does not exist	+
21	Nationality	«french1»	Such nationality does not exist	+
22	Nationality	«England»	Such nationality does not exist	+
23	Nationality	«British»	-	+
24	Nationality	«French»	-	+
25	Club	«Metalurg Zaporizhzhya»	Such club does not exist	+
26	Club	«FC Krivbass»	Such club does not exist	+
27	Club	«Shalke #04»	Such club does not exist	+
28	Club	Порожній рядок	-	+

Продовження таблиці 5.2

1	2	3	4	5
29	Club	«FC Schalke 04»	-	+
30	Club	«Arsenal»	-	+
31	Age	Порожній рядок	Age should be between 15 and 45	+
32	Age	«Agetext»	Age should be between 15 and 45	+
33	Age	«#54»	Age should be between 15 and 45	+
34	Age	«54»	Age should be between 15 and 45	+
35	Age	«13»	Age should be between 15 and 45	+
36	Age	«15»	-	+
37	Age	«45»	-	+
38	Age	«28»	-	+

Валідація категоріальних даних, таких як позиція футболіста та його трансферний статус, не проводиться.

Тестування параметрів гравця на відповідність шаблону відбувається у головному вікні на вкладці «Personal», після того, як гравця вибрано у таблиці. Панель з полями гравця, у яких відсутні значення, зображено на рисунку 5.9.

PERSONAL RATING

Name: Name is required

Nation: Nationality is required

Club: Such club does not exist

Age: Age should be between 15 and 45

Position:

Transfer Status:

Рисунок 5.9 – Редагування персональної інформації гравця

Після успішного проходження валідації параметрів, у адміністратора з'являється можливість додати гравця до БД, виконавши відповідний запит до сервера.

У разі, якщо правила валідації на рівні клієнту не відповідають обмеженням відповідної таблиці у БД, то сервер поверне помилку операції, а також виведе у консоль попередження. Робота серверу по роботі з даними правильного та неправильного формату зображено на рисунку 5.10.

```

Update PlayerId = 89998 | Time: -00:07:58.5823293 | Result: Error occurred
Read rating UserId = 6 PlayerId = 96839 | Time: -00:14:46.7406796
Read rating UserId = 6 PlayerId = 89998 | Time: -00:14:49.3949240
Read rating UserId = 6 PlayerId = 95400 | Time: -00:14:55.6135018
Read rating UserId = 6 PlayerId = 97058 | Time: -00:15:07.0884914
Read rating UserId = 6 PlayerId = 95400 | Time: -00:15:07.5057103
Update PlayerId = 95400 | Time: -00:15:17.9178344 | Result: Error occurred
Read rating UserId = 6 PlayerId = 89964 | Time: -01:19:28.2496502
Read rating UserId = 6 PlayerId = 96042 | Time: -01:19:28.9846123
Read rating UserId = 6 PlayerId = 93722 | Time: -01:19:30.3545120
Read rating UserId = 6 PlayerId = 95816 | Time: -01:35:48.3277274
Update PlayerId = 95816 | Time: -01:36:02.6421509 | Result: Correct
Update PlayerId = 95816 | Time: -01:36:05.3502697 | Result: Correct
Update PlayerId = 95816 | Time: -01:36:06.8881244 | Result: Correct

```

Рисунок 5.10 – Результат роботи серверу по зміни параметрів гравця

Помилки позначені червоним кольором відповідають перевіркам №10-12, наведеним у таблиці 5.2. Проблема у відсутності максимального ліміту символів у імені футболіста, хоча такий ліміт має бути та складати 50 одиниць, а мінімальний складає 2 символи, замість 3. Проблеми виправлено.

5.4.2 Чеклист функціонального тестування

Тестування програми проводиться на корпоративній, 64-розрядній операційній системі Windows 10, версії 21H2.

Чеклист тестування, згідно функціональних вимог наведено у таблиці 5.3.

Таблиця 5.3 – Чеклист функціонального тестування

Функція	Результат
1	2
Реєстрація нового користувача з коректними параметрами	Пройдено
Реєстрація нового користувача з неунікальними параметрами	Пройдено
Авторизація користувача з коректним параметрами	Пройдено
Авторизація користувача з некоректним параметрами	Пройдено
Редагування параметрів авторизованого користувача	Пройдено
Збереження нових параметрів авторизованого користувача	Пройдено
Відміна параметрів при редагуванні користувача	Пройдено
Завантаження списку гравців	Пройдено
Завантаження списку національностей	Пройдено
Завантаження списку клубів	Пройдено
Вибір гравця у таблиці гравців	Пройдено
Пошук гравця за іменем	Пройдено
Пошук гравця за клубом	Пройдено
Пошук гравця за клубом і іменем	Пройдено
Сортування таблиці гравців за ім'ям	Пройдено
Сортування таблиці гравців за національністю	Пройдено
Сортування таблиці гравців за віком	Пройдено
Сортування таблиці гравців за позицією	Пройдено
Сортування таблиці гравців за статусом	Пройдено
Додавання гравця з коректними, повними параметрами	Пройдено
Додавання гравця з коректними, неповними параметрами	Пройдено
Додавання гравця з некоректними параметрами	Пройдено
Видалення гравця	Пройдено
Оновлення таблиці гравців	Пройдено
Редагування параметрів гравця	Пройдено
Вибір команди гравця	Пройдено

Продовження таблиці 5.3

1	2
Збереження нових параметрів гравця	Пройдено
Відміна параметрів про редагуванні гравця	Пройдено
Зміна рейтингу гравця від 0 до 5 (репутація, слабка нога, майстерність).	Пройдено
Зміна рейтингу від 0 до 100.	Пройдено
Збереження рейтингу гравця	Пройдено
Відміна зміни рейтингових оцінок	Пройдено
Обнуління рейтингових оцінок	Пройдено
Підрахунок вартості гравця	Пройдено

5.4.3 Результати навчання запропонованої моделі

Гіперпараметри моделі, що стосуються її навчання обрано емпірично: кількість епох e , розмір партії (batch) b , крок збіжності λ (швидкість навчання). Для цього було проведено навчання НМ з цими параметрами та аналіз його результатів, стільки раз, доки не було прийнято рішення стосовно значень цих показників.

В контексті предметної області, об'єкт – це футболіст, незалежні змінні – характеристики гри гравця, залежна змінна – вартість гравця на трансферному ринку.

Навчання проводиться на вибірці з 15233 об'єктів, кожен з яких має значення для 31 незалежної ознаки та 1 залежної ознаки.

Значення ознак попередньо нормалізовано за описаними вище принципами (пункт 2.1.3).

Навчальну вибірку перемішано, для виключення можливої упорядкованості за якоюсь із ознак.

Початкову вибірку розділено на навчальну та тестову, у приблизному відношенні 15-20 % до 85-80 %.

Для аналізу процесу навчання моделі корисно будувати графік залежності значення функції помилки від номера ітерації навчання.

Для аналізу ступеня точності отриманої моделі величина квадратичної помилки мало що визначає. Натомість з цією ціллю використано коефіцієнт детермінації R^2 . Він характеризує частку дисперсії ознаки, яка пояснюється регресією, у загальній дисперсії ознаки.

Наприклад, при $e = 5$, $b = 20$, $\lambda = 0.01$, коефіцієнт детермінації R^2 для тестової вибірки дорівнює 0.848. Графічний смисл коефіцієнту детермінації зображений на рисунку 5.11.

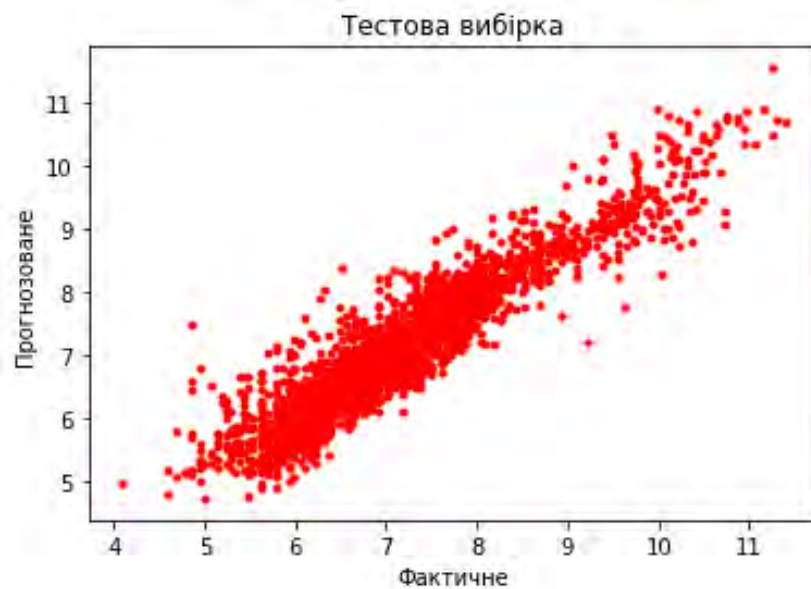


Рисунок 5.11 – Геометричний смисл R^2 для тестової вибірки

Процес навчання за таких же параметрів оптимізатора зображений на рисунку 5.12.

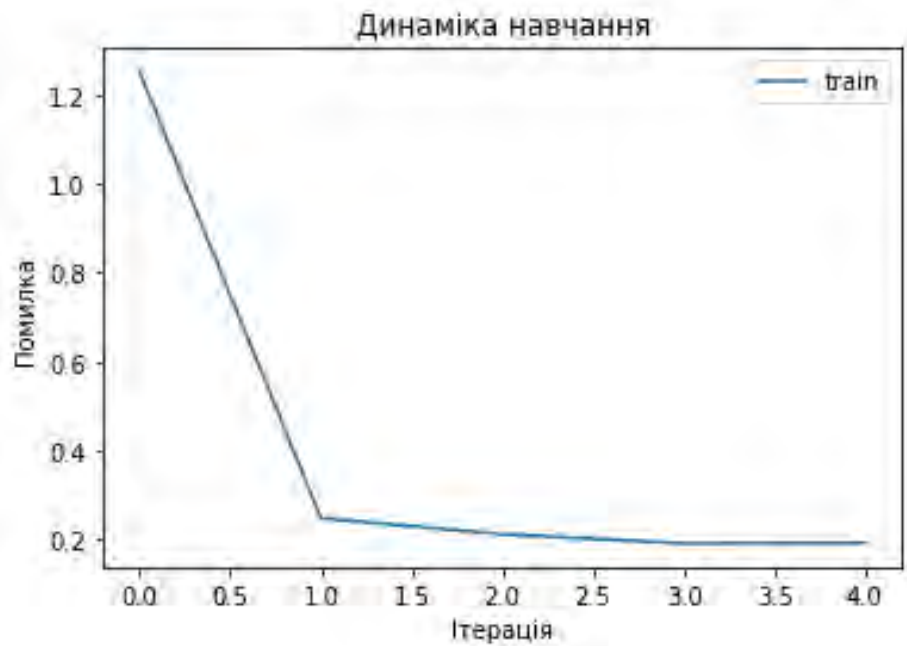


Рисунок 5.12 – Динаміка навчання

Проведено декілька тестів різних комбінацій параметрів оптимізатора. Зокрема можна виділити вибір низького значення кроку навчання рівного 0.1. В такому випадку різницю між фактичною та прогнозованою ціною зображено на рисунку 5.13.

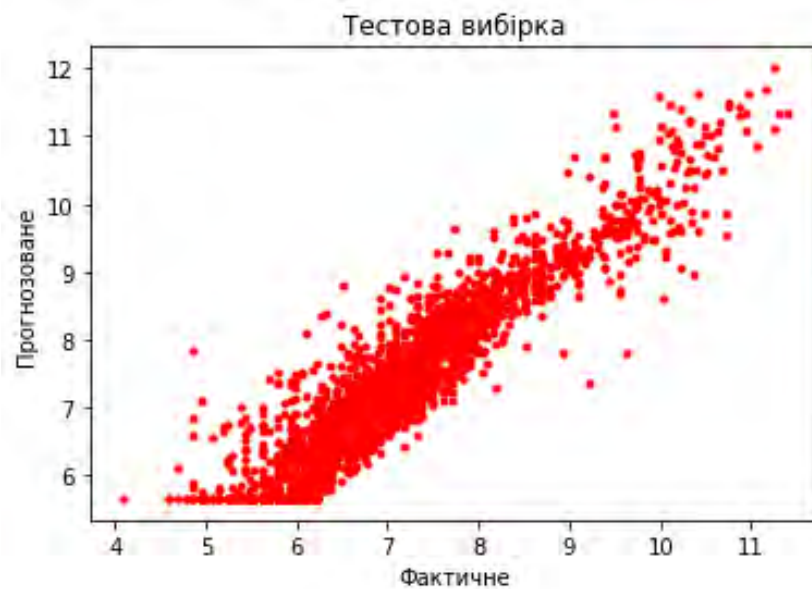


Рисунок 5.13 – Геометричний смисл $R^2=0.785$, при
 $e = 20, b = 20, \lambda = 0.1$

Як видно з рисунка, велика кількість значень ознаки не пояснюється

моделлю, через те, що оптимізатор не може точно дістатися мінімуму функції, не вистачає кроку збіжності.

Також показовим є приклад з параметрами $e = 5000$, $b = 20$, $\lambda = 0.0001$, динаміку навчання моделі за яких зображено на рисунку 5.14.

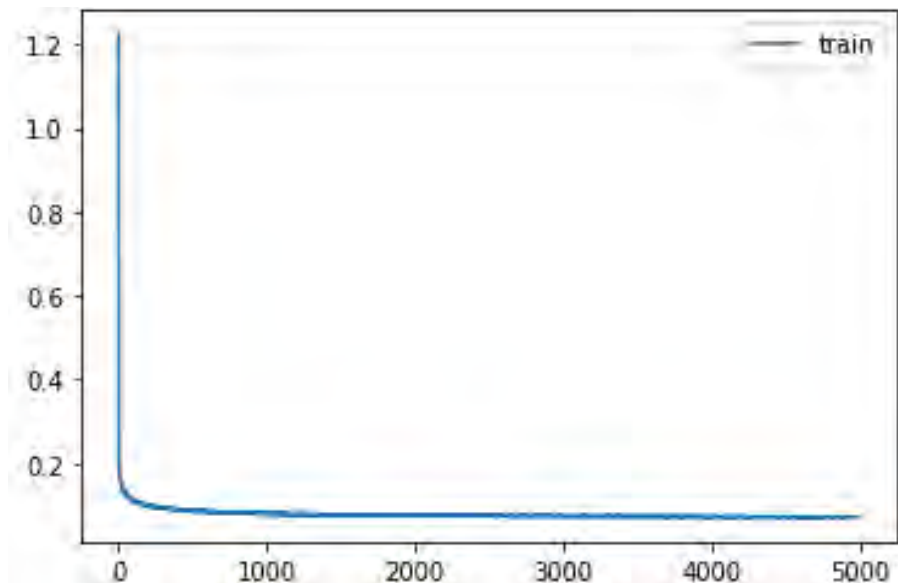


Рисунок 5.14 – Динаміка навчання, при $e = 5000$, $b = 20$, $\lambda = 0.0001$

Результатом навчання такої моделі є доволі велике значення $R^2 = 0.946$, однак лише для навчальної вибірки. Для тестової вибірки R^2 становить 0.898, і є навіть нижчим за звичне.

Після проведення навчання та тестування моделі з різними параметрами можна виділити її головні особливості:

- більшою мірою навчання відбувається у перших двох епохах;
- стабільна точність визначення вихідної ознаки ($R^2 \approx 0.89$) для тестової вибірки в незалежності від гіперпараметрів.

Це пояснюється вдалим вибором навчаючої вибірки: вона є великою за обсягом, репрезентативною, а її дані добре згруповані.

5.4.4 Фінальна конфігурація моделі

В результаті структурного синтезу отримано наступну конфігурацію

НМ для вирішення завдання визначення ціни футболіста на трансферному ринку:

- матриця синаптичних зв'язків = [31 625 250 1];
- функція активація для прихованих шарів ReLU, лінійна функція активація для вихідного шару;
- функція помилки MSE;
- оптимізатор Adam;
- крок збіжності $\lambda = 0.001$;
- епох навчання – 30, пакетів – 20.

В результаті параметричного синтезу визначено вагові коефіцієнти моделі. Коефіцієнт детермінації прогнозованої ціни на гравця становить $R^2=0.900$.

5.4.5 Порівняння моделі з аналогічними рішеннями

Порівнювати отриману модель коректно з роботами, в яких було використано подібні набори даних [36], тобто з класифікаторною нейронною мережею [34] та моделлю машинного навчання методом випадкового лісу [35].

Порівняльну характеристик створеної моделі з аналогічними рішеннями наведено у таблиці 5.4.

Таблиця 5.4 – Порівняння побудованої моделі з аналогічними рішеннями

Критерій	Робота [34]	Робота [35]	Власна
Тип моделі	НМ класифікатор	Випадковий ліс	НМ для прогнозування
Вибір ознак	+	–	+
Точність	–	+	+

У роботі [34] було використано той же самий набір даних, що і при навчанні розробленої моделі. Однак різниця у ознаках все ж таки є: їх більше, присутній показник «загальний рейтинг». Дані взяті за 2017 рік. Тим не менш, вибір ознак дуже схожий. Проблема роботи [35] полягає у визначні типу моделі – це НМ класифікатор. Через це процес навчання моделі доволі складний, з попередженням перенавчання, та довгий. Однак і з точністю проблеми. По-перше модель не оцінює дорогих гравців, на відмінну від запропонованої моделі. По-друге, вводиться спеціальний критерій оцінки ціни, як категоріальної величини. Його значення у кінцевому рахунку складає 87%, однак фінальна прогнозована ціна варується у інтервалі від 1 до 6 мільйонів євро, при середньому значенні трансферної вартості ≈ 1 мільйон євро. Натомість у запропонованій моделі як раз невеликі ціни гравців визначаються добре.

5.5 Висновки за розділом 5

У розділі 5 було описано призначення програми для футбольних функціонерів, букмекерів, незалежних експертів та звичайних користувачів. Також, наведено перелік умов, як програмних, так і апаратних, за яких застосунок коректно працює. Робота всього ПЗ описана у вигляді користувацьких потоків та протестована на відповідність функціональним вимогам. Було проведено експериментальне дослідження запропонованої моделі.

ВИСНОВОК

З метою створення програми для автоматизації процесу визначення ціни футболіста на трансферному ринку, було досліджено два основоположних методи оцінки вартості гравця – метод мудрості натовпу та математичну регресійну модель. Виходячи з їх спільних переваг та цілей розроблюваного продукту, було прийнято рішення об'єднати два методи, генеруючи один новий.

Наукова новизна роботи полягає в запропонованій моделі оцінювання трансферної вартості футболіста, що характеризується використанням нейронної мережі на основі показників футболіста, дозволяючи в результаті отримувати більш гнучкі прогностичні рішення, ніж у моделях, що використовують суто об'єктивні дані.

Спираючись на множину експертних оцінок, як ігрових показників футболіста, так і його вартості на трансферному ринку, було проведено навчання нейронної мережі. Виходячи з отриманого коефіцієнту детермінації, можна стверджувати, що отримана модель є більш точною за аналогічні, а набір ознак більш репрезентативний, ніж у подібних моделях зі схожими використаними даними.

Для отриманої математичної моделі було створено програмну оболонку з графічним користувацьким інтерфейсом. Початкова задача спонукала до розробки клієнт-серверного додатку для можливості одночасної роботи з функціональністю великої кількості користувачів та взаємодії між собою.

Подальший супровід додатку вимагає створення повноцінної електронної системи обміну гравцями, на основі їх вартості, підрахованої моделлю.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кириленко, В. В. Економіка. Навчальний посібник для студентів вищих навчальних закладів [Текст] / В. В. Кириленко. – Тернопіль : Економічна думка, 2002. – 193 с.
2. Кто такой спортивный директор и для чего он нужен футбольному клубу? [Электрон. ресурс]. – Режим доступа : <https://www.sports.ru/tribuna/blogs/thefootballfactory/2857138.html>.
3. Regulations on the status and transfer of players [Electronic resource]. – Access mode : <http://docplayer.net/1570706-Regulations-on-the-status-and-transfer-of-players.html>.
4. Muller, O. Beyond crowd judgments: Data-driven estimation of market value in association football [Text] / O. Muller, A. Simons, M. Weinmann // European journal of operational research. – 2017. – Vol. 263, Issue 2. – P. 611–624.
5. Herm, S. When the crowd evaluates soccer players' market values: Accuracy and evaluation attributes of an online community [Text] / S. Herm, H.-M. Callsen-Bracker, H. Kreis // Sport Management Review. – 2014. – Vol. 17, Issue 4. – P. 484–492.
6. He, M. Football player's performance and market value [Text] / Miao He, R. Cachucho, A. J. Knobbe // European conference on machine learning and principles and practice of knowledge discovery in databases : Thirtieth international conference, 7–11 September 2015 : proceedings. – Porto : MLSA@PKDD/ECML, 2015. – P. 87–95.
7. GitHub repository [Electronic resource]. – Access mode : <https://github.com/ewenme/transfers/>.
8. Poli, R. Financial analysis of the transfer market in the big-5 leagues (2010-2018) [Electronic resource] / R. Poli, L. Ravenel, R. Besson. – Access mode : <https://www.football-observatory.com/IMG/sites/mr/mr37/en/>.
9. Bhatia, A. Do football transfers actually improve performance : thesis ... bachelor / Bhatia Adil. – Cambridge : Harvard College, 2020. – 72 p.

10. Neymar's move to PSG will cost the French soccer giant more than \$500 million [Electronic resource]. – Access mode : <https://www.businessinsider.com/how-much-neymar-salary-transfer-fee-psg-barca-2017-8/>.
11. Poli, R. The real impact of COVID on the football players' transfer market [Electronic resource] / R. Poli, L. Ravenel, R. Besson. – Access mode : <https://football-observatory.com/IMG/sites/mr/mr58/en/>.
12. How good was Serie A in the 2000s [Electronic resource]. – Access mode : <https://thesefootballtimes.co/2017/10/03/how-good-was-serie-a-in-the-2000s/>.
13. Szymanski, S. The financial crisis and English football: the dog that will not bark [Text] / S. Szymanski // International journal of sport finance. – 2010. – Vol. 5, № 1. – P. 28–40.
14. Kehan, C. Specific causes of financial crisis at Barcelona and countermeasures [Text] / C. Kehan // Financial innovation and economic development : Seventh international conference, 14–16 January 2022 : proceedings. – Zhuhai : Atlantis Press, 2022. – P. 187–193.
15. CIES Football Observatory [Electronic resource]. – Access mode : <https://football-observatory.com/>.
16. About CIES football observatory [Electronic resource]. – Access mode : <https://football-observatory.com/-About->.
17. Poli, R. Transfer values and probabilities: the CIES football observatory approach [Electronic resource] / R. Poli, L. Ravenel, R. Besson. – Access mode : <https://www.football-observatory.com/IMG/sites/mr/mr16/en/>.
18. Poli, R. Financial analysis of the transfer market in the big-5 European leagues (2010-2019) [Electronic resource] / R. Poli, L. Ravenel, R. Besson. – Access mode : <https://football-observatory.com/IMG/sites/mr/mr47/en/>.
19. Poli, R. Demographic study of football in Europe [Electronic resource] / R. Poli, L. Ravenel, R. Besson. – Access mode : <https://football-observatory.com/IMG/sites/mr/mr19/en/>.

20. Poli, R. Scientific evaluation of the transfer value of football players [Electronic resource] / R. Poli, L. Ravenel, R. Besson. – Access mode : <https://football-observatory.com/IMG/sites/mr/mr53/en/>.
21. Transfermarkt [Electronic resource]. – Access mode : <https://www.transfermarkt.world/>.
22. How the volunteers of data website Transfermarkt became influential players at European top football clubs [Electronic resource]. – Access mode : <https://www.ftm.eu/articles/transfermarkt-volunteers-european-football>.
23. Transfermarkt market value explained – How is it determined [Electronic resource]. – Access mode : <https://www.transfermarkt.co.in/transfermarkt-market-value-explained-how-is-it-determined-/view/news/385100>.
24. Mannes, A. E. The wisdom of select crowds [Text] / A. E. Mannes, J. B. Scoll, R. P. Larrick // *Journal of Personality and Social Psychology*. – 2014. – Vol. 107, Issue 2. – P. 276–299.
25. Coates, D. The wisdom of crowds and transfer market values [Text] / D. Coates, P. Parshakov // *European Journal of Operational Research*. – 2022. – Vol. 301, Issue 2. – P. 523–534.
26. Субботін, С. О. Еволюційні обчислення та програмування : навчальний посібник [Текст] / А. О. Олійник, С. О. Субботін; за ред. С. О. Субботіна – Запоріжжя : ЗНТУ, 2014. – 219 с.
27. Rodriguez, M. S. Uncovering value drivers of high performance soccer players [Text] / M. S. Rodriguez, A. R. Hassan, A. Coad // *Journal of sports economics*. – 2014. – Vol. 20, Issue 6. – P. 819–849.
28. Poli, R. Econometric approach to assessing the transfer fees and values of professional football players [Text] / R. Poli, R. Besson, L. Ravenel // *Economies*. – 2021. – Vol. 10, Issue 1. – P. 1–14.
29. Ante, L. Determinants of transfers fees: evidence from the five major European football leagues [Electronic resource] / L. Ante. – Access mode : <http://dx.doi.org/10.13140/RG.2.2.18356.91526/1>.

30. WhoScored [Electronic resource]. – Access mode : <https://www.whoscored.com/>.
31. 11 Biggest player rating mistakes in FIFA 21 [Electronic resource]. – Access mode : <https://www.linkedin.com/pulse/11-biggest-player-rating-mistakes-fifa-21-pranav-khatri>.
32. FIFA 20 : 20 player ratings they got wrong [Electronic resource]. – Access mode : <https://www.highpresssoccer.com/fifa-20-20-player-ratings-they-got-wrong/>.
33. FIFA 22 attributes explained [Electronic resource]. – Access mode : <https://www.fifplay.com/fifa-22-player-attributes/>.
34. Dey, S. Pricing football players using neural networks [Electronic resource] / S. Dey. – Access mode : <https://arxiv.org/abs/1711.05865>.
35. Al-Asadi, M. A. – Predict the value of football players using FIFA video game data and machine learning techniques [Text] / M. A. Al-Asadi, S. Tasdemir // IEEE Access. – 2022. – Vol. 10. – P. 22631–22645.
36. Players [Electronic resource]. – Access mode : <https://sofifa.com/players>.
37. Normalization vs standardization [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/normalization-vs-standardization/>.
38. How to transform data in python (Log, Square root, Cube root) [Electronic resource]. – Access mode : <https://www.statology.org/transform-data-in-python/>.
39. Log-normal distribution [Electronic resource]. – Access mode : <https://www.investopedia.com/terms/l/log-normal-distribution.asp>.
40. Hall, M. A. Correlation-based feature selection for machine learning : thesis ... doctor of philisophy / Mark A. Hall. – Hamilton : University of Waikato, 1999. – 178 p.
41. Creating annotated heatmaps [Electronic resource]. – Access mode : https://matplotlib.org/3.5.0/gallery/images_contours_and_fields/image_annotated_heatmap.html.

42. FeatureSelector: отбор признаков для машинного обучения на Python [Электрон. ресурс]. – Режим доступа : <https://proglib.io/p/feature-selector>.
43. Various optimization algorithms for training neural network [Electronic resource]. – Access mode : <https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6>.
44. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. [Текст] / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с.
45. Стандартная библиотека Python [Электрон. Ресурс]. – Режим доступа : https://digitology.tech/docs/python_3/library/index.html.
46. GitHub repository [Electronic resource]. – Access mode : <https://github.com/ChrisKnott/Eel>.
47. Marshal – internal Python object serialization [Electronic resource]. – Access mode : <https://docs.python.org/3/library/marshal.html>.
48. Топ-10 библиотек Python для data science [Электрон. Ресурс]. – Режим доступа : <https://datastart.ru/blog/read/top-10-bibliotek-python-dlya-data-science>.
49. Andrej Karpathy blog – a recipe for training neural networks [Electronic resource]. – Access mode : <http://karpathy.github.io/2019/04/25/recipe/>.
50. Python developers survey 2020 results [Electronic resource]. – Access mode : <https://www.jetbrains.com/lp/python-developers-survey-2020/>.
51. Spyder notebook [Electronic resource]. – Access mode : <https://docs.spyder-ide.org/5/plugins/notebook.html>.
52. Variable explorer [Electronic resource]. – Access mode : <http://docs.spyder-ide.org/current/panes/variableexplorer.html>.
53. Plots [Electronic resource]. – Access mode : <http://docs.spyder-ide.org/current/panes/plots.html>.
54. PyCharm installation requirements [Electronic resource]. – Access mode : <https://www.msvu.ca/campus-life/campus-services/it-services/software-catalogue/pycharm/>.

55. Штейников, Р.С. Сравнительный анализ графических систем WPF и WinForms [Текст] / Р. С. Штейников // Студенческий научный форум : VIII Международная студенческая научная конференция, Волгоград, 15–20 февраля 2016 г. : тезисы докладов. – Волгоград: Российской Академией Естествознания, 2016. – 3 с.

56. Using data contracts [Electronic resource]. – Access mode : <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/using-data-contracts>.

57. Types supported by the data contract serializer [Electronic resource]. – Access mode : <https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/types-supported-by-the-data-contract-serializer>.

58. Windows Communication Foundation client libraries [Electronic resource]. – Access mode : <https://github.com/dotnet/wcf>.

59. PropertyChanged.Fody [Electronic resource]. – Access mode : <https://github.com/Fody/PropertyChanged>.

ДОДАТОК А
Технічне завдання

A.1 Вступ

A.1.1 Назва програми

Найменування – «Програмне забезпечення підтримки прийняття рішень про футбольну трансферну активність». Найменування англійською – «Football Transfer Decision Support Software».

A.1.2 Призначення та область застосування

Програма має використовуватися у трансферному секторі бізнес політики футбольних клубів. Тому цільовими користувачами можуть бути усі функціонери клубу, що приймають участь у проведенні такої політики, від спортивного тренера до робітника селекційного відділу.

Розроблювана програма призначена для автоматизації процесу оцінювання гравця, його ігрових та неігрових характеристик. Головним таким показником футболісту є вартість гравця на трансферному ринку – інструмент проведення трансферної політики футбольного клубу.

A.2 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Програмне забезпечення підтримки прийняття рішень про футбольну трансферну активність» у відповідності з наказом № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка».

Основою для проведення розробки програмного забезпечення є завдання на дипломний проєкт Ліпіхіна Сергія Олександровича, видане 4 квітня 2022 року та затверджене Національним Університетом «Запорізька Політехніка».

Умовне позначення теми розробки (шифр теми) – «РинкТрансфВартість», та англійською – «TransferMarketValue».

A.3 Призначення розробки

A.3.1 Функціональне призначення

Функціональним призначенням програми є надання можливості користувачу:

- збирати, зберігати та розповсюджувати інформацію про загальновідомі персональні дані про футболістів: їх ім'я, вік, громадянство, тощо;
- оцінювати футболістів за певними, заздалегідь визначеними критеріями: потенціал, самовладання, бачення поля, сила удару, тощо;
- для кожного футболіста, з огляду на виставлену йому користувачем оцінку, обчислювати орієнтовану вартість на трансферному ринку, у грошовому еквіваленті;
- розподіляти зобов'язання та права користувачів в системі, розгортаючи клієнт-серверну архітектуру додатка.

A.3.2 Експлуатаційне призначення

Програма має експлуатуватися як у особистих інтересах окремо взятого користувача, так і у інтересах певної спільноти людей, які мають загальні інтереси, наприклад, працівники та функціонери футбольного клубу чи соратники по агентській діяльності.

Цілями експлуатації програмного продукту користувачами можуть бути:

- збирання даних про футболістів, з ціллю впорядкованої їх обробки у подальшій діяльності, вільного до них доступу та зручної обробки. Більш детально – введення статистики за тими футболістами, які є в короткому списку клубу для їх придбання, оформивши трансферну угоду;

- оцінювання футболістів тренером команди, з зазначенням здібностей та схильностей гравців, для створення більш гнучкої схеми (тактики) гри, для підвищення рівня «боездатності» спортивного колективу;

- визначення ціни на футболістів агентами, функціонерами селекційного відділу, спортивними директорами, менеджерами та власниками клубів, для проведення ефективної трансферної політики та укладання найбільш вигідних угод;

- оцінювання виступів гравців (як їх рейтингу, так і вартості) одразу багатьма користувачами, для порівняння та зіставлення декількох точок зору на гру одного і того ж самого футболіста. Можливо організувати навчання правильному перегляду футбольних поєдинків та отримання з них експертних знань, перевіряючи їх за допомогою порівняння оцінок, наданих у програмі.

- оцінювання виступів гравців на постійній основі – складання щомісячних звітів про гру футболістів, чи оцінка їх діяльності по кожному матчу, зберігаючи дані у системі за датою;

- створення імпровізованих команд «мрії», спираючись на оцінку гравців та комбінуючи їх між собою, – створювати футбольні колективи.

А.4 Вимоги до програми

А.4.1 Вимоги до функціональних характеристик

А.4.1.1 Вимоги до складу виконуваних функцій

Функціональність програми повинна бути розділена на дві частини:

- перша частина повинна забезпечувати конструювання та розгортання певної автоматизованої моделі підрахунку вартості гравця;

- друга частина – створювати графічну оболонку по забезпеченню додаткових функцій та надавання користувачу можливості взаємодіяти зі створеною математичною моделлю.

Програма має задовольняти можливість виконання зазначених нижче функцій, пов'язаних з розробкою та роботою моделі системи:

- виконувати під'єднання до інтернет мережі, відправляти та обробляти запити з приводу вивантаження необхідних даних у вільному доступі;

- зчитувати та обробляти HTML сторінки, виконуючи парсинг рядків та, як наслідок, витягуючи необхідну інформацію;

- працювати із .txt, .csv та іншими файлами, необхідними для збереження даних для моделі;

- реструктуризовувати початкову інформацію, для приведення її у більш зрозумілі вигляди як для людини, так і для машини;

- обробляти та статистично аналізувати дані;

- будувати візуальні представлення результатів аналізу даних, такі як: графіки, гістограми, діаграми, тощо;

- будувати нову та зберігати отриману математичну модель визначення вартості гравця на трансферному ринку, для подальшого її використання.

Програма має виконувати наступні функції, пов'язані з реалізацією можливостей моделі, через користувацький графічний інтерфейс:

- працювати із .txt, .csv та іншими файлами, необхідними для роботи з даними;

- розгортати БД, забезпечуючи функціональність для роботи з нею: читання, видалення, зміна та додавання записів;

- виконувати пошук даних, їх сортування та ітерацію;

- використовувати функціональність розробленої математичної моделі, підтримуючи з нею взаємодію програмно;

- розгортати сервер, що повинен надавати клієнтам доступ до даних та обчислювальних можливостей;

- виконувати під'єднання клієнтів до серверу, забезпечуючи їх усією доступною функціональністю по взаємодії з сервером;

- забезпечувати можливість реєстрації нових користувачів та авторизації зареєстрованих;
- забезпечувати можливістю користувачів взаємодіяти із графічним інтерфейсом, з його базовими елементами;
- реалізовувати ергономічний інтерфейс, відповідний предметній області, без навантаження якою-небудь графікою;
- представляти багатосторінковий додаток для розміщення у одному вікні широкої функціональності по взаємодії з системою;
- створювати власні графічні, користувацькі елементи контролю;
- перевіряти дані на дотримання визначеного формату. Виконувати валідацію як на рівні представлення, так і на рівні БД;
- виконувати обробку користувацьких сигналів від клавіатури та миші;
- обробляти весь спектр можливих помилок та виключень, які можуть виникнути в ході роботи програми;
- надавати можливість змінювати інформацію про користувача системи;
- працювати з масивом футболістів: створювати, видаляти, редагувати, згідно до запитів користувача;
- надавати можливість користувачу змінювати рейтингові характеристики футболістів, чи інші оцінювальні показники;
- рахувати трансферну вартість гравця згідно з його здібностями;
- зберігати інформацію з зазначенням часу її створення та редагування для майбутнього доступу до неї;
- забезпечувати розділення функціональності та етапів створення програми на частини, згідно з провідними архітектурними шаблонами проектування;
- забезпечувати користувача інформацією про роботу сервера, можливо у вигляді власного невеликого графічного додатка або у вигляді командного рядка.

А.4.1.2 Вимоги до організації вхідних даних

Вхідні дані, що будуть використовуватися моделлю, мають відповідати наступним вимогам:

- відповідність предметній області;
- здатність до формалізації;
- великий обсяг;
- репрезентативність;
- узагальнення;
- представлення вартості гравця як неперервного числа, поданого у грошовому еквіваленті.

Вхідні дані, у контексті клієнтського додатку, – це ті дані, які отримані у результаті взаємодії користувача з графічним інтерфейсом програми. Вхідні дані мають бути представлені:

- сигналами, подані від користувача системі. Такі дані програма має обробляти на рівні подання та згідно до них вибирати шляхи рішення проблемних ситуацій;
- введеною користувачем з клавіатури текстовою інформацією. Такі дані, разом з результатами обробки сигналів, програма має перетворювати у зрозумілі їй та БД структури даних, які пізніше можуть бути застосовані для внутрішніх обчислень, а у кінцевому рахунку, оформлені як вихідні дані.

Вимоги до сигналів мають відповідати загальному стилю оформлення представлення програми.

Вимоги до структур даних мають відповідати інформаційним структурам та структурі таблиць БД.

А.4.1.3 Вимоги до організації вихідних даних

Головним результатом роботи моделі програми є підрахунок вартості гравця на трансферному ринку – це дійсне число.

Вихідні дані – це дані, які отримані шляхом перетворення вхідних даних програмою. Вихідні дані подаються користувачу у вигляді інформації, за допомогою графічного інтерфейсу програми.

Вихідні дані мають бути представлені у тій же формі, у якій задавалися вхідні – це має бути забезпечене успішним проектуванням інтерфейсу програми. Більш того, вихідні дані повинні сприйматися користувачем однозначно, що має підвищену важливість у розумінні введених футбольних понять та термінів. Усі неточні чи неповні відомості, або дані, які можуть трактуватися по різному, мають бути підкріплені та підкреслені пояснювальними повідомленнями.

Наприклад:

- кожна характеристика гравця повинна мати пояснення – яку здатність футболіста ця характеристика висловлює;
- кожна сутність предметної області повинна бути підкреслена кольором чи графічним вираженням.

А.4.2 Вимоги до надійності

А.4.2.1 Вимоги до забезпечення надійного функціонування програми

Надійне функціонування програми повинно бути забезпечене користувачем у сукупності організаційно-технічних пунктів:

- безперервне постачання живлення технічних пристроїв на яких буде встановлена дана програма;
- використання ліцензованого програмного забезпечення;
- регулярним виконання вимог захисту інформації. Виконання перевірок на наявність вірусів.

Програма не повинна використовувати функцію збереження стану, тому при некоректному закритті програми, у випадку збійної ситуації, система не «розгортає» попередній стан при повторному запуску програми.

A.4.2.2 Збійні ситуації

Наступні збійні ситуації мають оброблятися програмою:

- відсутній зв'язок з сервісом;
- відсутній зв'язок з БД;
- занесення до БД даних, що не відповідають обмеженням;
- занесення до БД даних, що мають бути унікальними, хоча такими не є;
- спроба звертання до неіснуючої таблиці БД
- некоректне порівняння значення у комірці БД з рядком, що зберігає символ лапок (');
- неможливо виконати запит за зазначеними початковими даними;
- неможливість читання даних за посиланням у інтернет мережі;
- помилка індексу ітерації;
- неможливість коректного зчитування даних з БД;
- некоректна валідація даних на рівні представлення;
- неправильно прив'язані дані;
- відсутність користувача у системі;
- некоректно вказані дані користувачів при авторизації;
- некоректно вказана дата;
- невказаний обов'язковий параметр;
- некоректне введення даних для побудови графіку;
- некоректна робота із моделлю системи;

A.4.3 Вимоги до технічних та програмних засобів

Програма має бути створена як десктоп додаток, для персонального комп'ютеру під керівництвом операційної системи Windows 10.

Початкові мінімальні вимоги до технічних засобів, які можуть бути змінені, згідно до використовуваних технологій при розробці програми, наступні:

- центральний процесор з частотою 1 ГГц або швидший;
- оперативна пам'ять: 2 ГБ для 64-розрядної системи;
- місце на жорсткому диску: 32 ГБ для 64-бітної і 32-бітної ОС;
- графічний процесор з мінімальною підтримкою технології роботи з графічними даними DirectX, щонайменше версії 9;
- розширення екрану: 1024 x 768 пікселів. Може бути і більшим.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати:

- розроблене технічне завдання відповідно до теоретичних відомостей, предметної області, мети та завдань, що ставляться перед розробкою та розробником;
- пояснювальна записка з виконаної роботи, що включає: опис проблемної області, зокрема футбольного трансферного ринку як об'єкта вивчення; опис проведеного дослідження та класифікації типу вирішуваної задачі, конструювання та налагодження математичної моделі, опис проєктування системи, опис програми (структурна схема програмного забезпечення, опис модулів з визначенням основних функцій та описом їх реалізації), опис тестування розробленої програми та порядку використання створеного додатку;
- перелік джерел інформації, та супровідні записи про використану інформацію з таких конкретних джерел.

A.6 Стадії та етапи розробки

Розробка складається з наступних етапів:

- постановка завдання роботи (1 тиждень) – завершується складанням технічного завдання із переліченими головними відомостями про майбутній проєкт та вимогами до розроблюваної програми;

- аналіз предметної області (2 тиждень). В ході цього етапу необхідно провести аналіз проблемної області. Він має включати розгляд теоретичної точки зору на проблеми, наведення термінів та їх пояснення тощо;

- дослідження моделей для розв’язання проблеми (3 тиждень). Використовується огляд аналогічних програмних засобів та аналізується їх запропонована методика досягнення цілі – визначення ціни футболісту. Виконується вибір структури моделі та її реалізація;

- проєктування програми (4 тиждень). Виділяються основні об’єкти, основні компоненти майбутньої системи; аналізуються функції, які виконують виділені об’єкти предметної області. В результаті проведеного проєктування має бути отриманий робочий варіант архітектури системи, що включає: пакети, модулі, класи, бібліотеки, тощо, з описом призначення та функцій кожного компонента;

- розробка програми (5 тиждень). Потрібно реалізувати, проаналізувати та протестувати базові функції системи. Цей етап можна рахувати завершеним, у тому разі, якщо більша частина основної бізнес-логіки предметної області була перенесена в інформаційні технології та пов’язана з математичною моделлю;

- тестування та експериментальне дослідження програми (6 тиждень). Необхідно описати головні залежності для коректної роботи програми, провести її тестування та навести опис циклу взаємодії користувача із додатком;

- оформлення програмної документації (7 тиждень). Склад та зміст програмної документації, по факту завершення роботи, описано у розділі А.5.

ДОДАТОК Б
Текст програми

Б.1 Текст файла hrefs.py

```
# hrefs.py script for reading players hrefs from sofifa.com for future data

import os
import requests
import numpy as np
from bs4 import BeautifulSoup

# number of pages to read
_numberOfpages = 300
# hrefs list directory
filename
os.path.dirname(os.path.realpath(__file__))+'/data/hrefs/players.txt'

def _get_hrefs(numberOfpages):

    url = 'https://sofifa.com/players?r=220001&set=true&offset='

    hrefs = []
    # iterate throw pages by changing offset
    for i in range(numberOfpages):
        # get offset
        offset = i*60
        # request html page
        page = requests.get(url + str(offset))
        # get text from page
        soup = BeautifulSoup(page.content, features='html.parser')
        html = list(soup.children)[2]
        body = list(html.children)[3]

        for a in body.select('td.col-name a', href=True):
            if '/player/' in str(a):
                hrefs.append(a['href'])
        # to view progress
        print("Iteration #{0} with offset = {1}".format(i, offset))

    return hrefs

def _write_hrefs_in_file(hrefs):
    with open(filename, 'w') as f:
        [f.write(href + '\n') for href in hrefs]

def read_hrefs_from_file():
    with open(filename, 'rt') as f:
        hrefs = f.readlines()
        # delete space character
        hrefs = [href.strip() for href in hrefs]
        # delete ununique players
        hrefs = np.unique(hrefs)
        return hrefs

# # get from internet
# _write_hrefs_in_file(_get_hrefs(_numberOfpages))
# hrefs = read_hrefs_from_file()
```

Б.2 Текст файла player.py

player.py module for reading all one-player-data from sofifa.com by reference

```
import os
import requests
import sampleinfo as si
from bs4 import BeautifulSoup
```

```
_namek = si.HEADER_22[0]
_clubk = si.HEADER_22[1]
_positionk = si.HEADER_22[2]
_weakfootk = si.HEADER_22[3]
_skillmovesk = si.HEADER_22[4]
_intrepk = si.HEADER_22[5]
_pricek = si.HEADER_22[6]
_featuresk = si.HEADER_22[7:]
```

```
def _print_soup_children(soup):
    print([type(item) for item in list(soup.children)])
```

```
def _get_price(price_string):
    price = '' # all PRICES IN EUROx1000
    for char in price_string.strip()[1:]: # price line
        if char == 'M': # million
            price = float(price) * 1000
            break
        elif char == 'K': # thousand
            price = float(price)
            break
        price += char
    return price
```

```
def sofifa_get_player(url = 'https://sofifa.com/player/188545/robert-lewandowski/220054/')
```

```
    root = os.path.dirname(os.path.realpath(__file__))+'/html_code/test'
```

```
    page = requests.get(url)
    # get page content
    soup = BeautifulSoup(page.content, features='html.parser')
```

```
    # get only tag items that keep features100
    features100 = soup.select('li span.bp3-tag.p')
    # convert to simple list
    features100 = [l.get_text() for l in features100]
    # convert to dict
    features100 = dict([[_featuresk[i], features100[i]]
                        for i in range(len(_featuresk))])
```

```
    player = dict()
```

```
    name = soup.select('h1', class_='ellipsis')[0].get_text()
    player[_namek] = name
```

```
    club = soup.select('h5 a')[0].get_text().lstrip()
```

```

player[_clubk] = club

position = soup.select('li span.pos')[1].get_text()
player[_positionk] = position

weakFoot = \
    soup.select('div.block-quarter
li.ellipsis')[1].get_text()[0]
player[_weakfootk] = weakFoot

skillMoves =\
    soup.select('div.block-quarter
li.ellipsis')[2].get_text()[0]
player[_skillmovesk] = skillMoves

internationalReputation =\
    soup.select('div.block-quarter
li.ellipsis')[3].get_text()[0]
player[_intrepk] = internationalReputation

price = _get_price(soup.select('section
div')[4].get_text())
player[_pricek] = price

player.update(features100)

return player

```

Б.3 Текст файлу htмлdata.py

```

# htмлdata.py script for reading all plyers data from sofifa.com iterating
# throw hrefs/players.txt

import sys
sys.path.append('sofifa')

import csv
import numpy as np

import hrefs
import player
import sampleinfo as si

_root = './sofifa/data/dataset/'

def _write_players_in_file(players):
    with open(_root + 'source.csv', 'w', encoding='UTF8', newline='') as f:
        writer = csv.writer(f)
        writer.writerow(si.HEADER_22)
        writer.writerows(players)

def _write_errors_in_file(errors):
    with open(_root + 'errors.csv', 'w', encoding='UTF8', newline='') as f:
        [f.write(error + '\n') for error in error_players]

```

```

def get_players(players_number):
    """Iterate throw hrefs/players.txt file and request fifa website to get
    all necessary info about players"""

    url = 'https://sofifa.com'
    playerref = hrefs.read_hrefs_from_file()

    if players_number < 0 or players_number > len(playerref):
        players_number = len(playerref)

    players = []
    error_players = []
    for i in range(0, players_number):
        print('Reading player #{0} on {1}'.format(i, playerref[i]))

        try:
            players.append(list(player.sofifa_get_player(url
playerref[i]).values()))
        except:
            error_players.append(playerref[i])
            pass

    return np.array(players), np.array(error_players)

# players, error_players = get_players(17979)
# _write_players_in_file(players)
# _write_errors_in_file(error_players)

```

Б.4 Текст файлы csvdata.py

```

# csvdata.py script with reforming and restructuring data from sofifa.com
# to appropriate format for machine learning processes

import os
import csv
import numpy as np
import sampleinfo as si
from itertools import tee

def _read_source_csv():
    """ Read and template data from /data/dataset/source.csv after multiple
    html-requests for each player from website sofifa.com"""
    with
open(os.path.dirname(os.path.realpath(__file__))+'/data/dataset/source.csv'
    , encoding="utf8") as f:
        # read
        csv_reader = csv.reader(f, delimiter=',')
        # not include header
        next(csv_reader, None)
        # csv-list comprehension method of packing data.
        # create two iterators
        i1, i2, i3, i4, i5 = tee(csv_reader, 5)
        # personal info: name, club, pos
        names = np.array([[data[0]] for data in i1])
        clubs = np.array([[data[1]] for data in i2])
        positions = np.array([[data[2]] for data in i3])
        # features 1-5 0-100

```

```

        features = np.array([[float(d) for d in data[3:6] + data[7:]] for data
in i4])
    # prices
    prices = []
    [prices.extend([float(d) if '0Value' not in str(d) else float(0)
                    for d in data[6:7]]) for data in i5]
    prices = np.asarray(prices)
    return names, clubs, positions, features, prices

def _generalize_position(positions):
    """ Too many positions need to be replaced by only 'GK' 'MF' 'FW' 'GK'"""

    keygk = ['GK']
    keydef = ['CB', 'LB', 'RB', 'RWB', 'LWB', 'LCB', 'RCB']
    keymid = ['CM', 'CDM', 'CAM', 'RM', 'LM', 'LCM', 'RCM', 'LDM', 'RDM']
    keyfor = ['ST', 'CF', 'RW', 'LW', 'RF', 'LF', 'LS', 'RS']
    left = ['RES', 'SUB']

    for i in range(len(positions)):
        if positions[i] in keygk:
            positions[i] = 'GK'
        elif positions[i] in keydef:
            positions[i] = 'DF'
        elif positions[i] in keymid:
            positions[i] = 'MF'
        elif positions[i] in keyfor:
            positions[i] = 'FW'
        else:
            positions[i] = 'MF'

    return positions

def _formatate_table(table):
    """ Some manipulations with data"""

    # unique rows handler
    repeats = dict()
    final = []
    for row in table:
        # remove player with 0 transfer value
        if float(row[3]) == 0:
            continue
        # remove GK
        if str(row[2]) == 'GK':
            continue
        # remove repeated names
        if row[0] not in repeats:
            final.append(row)
            repeats[row[0]] = True

    return np.array(final)

# read data frm source file
names, clubs, positions, features, prices = _read_source_csv()

# generalize players positions
positions = _generalize_position(positions)

# transform 1-row array into 1-column array

```

```

names = names.reshape(len(names), 1)
clubs = clubs.reshape(len(clubs), 1)
positions = positions.reshape(len(positions), 1)
prices = prices.reshape(len(prices), 1)

# concatenate arrays and got table
table = np.append(names, clubs, axis=1)
table = np.append(table, positions, axis=1)
table = np.append(table, prices, axis=1)
table = np.append(table, features, axis=1)

# remove all redundant table info
table = _formatate_table(table)

# write final dataset into file
with open('./csv_files/dataset.csv', 'w', encoding='UTF8', newline='') as f:
    writer = csv.writer(f)
    writer.writerow(si.HEADER_22)
    writer.writerows(table)

```

Б.5 Текст файлу csv_reader.py

```

# csv_reader.py script for dataset reading from .csv file

import csv
import os
import numpy as np
from itertools import tee

def read_dataset(filename = 'dataset.csv'):
    with open(os.path.dirname(os.path.realpath(__file__))+'./csv_files/' +
filename
, encoding="utf8") as f:
        # read
        csv_reader = csv.reader(f, delimiter=',')
        # not include header
        next(csv_reader, None)
        # csv-list comprehension method of packing data.
        # create two iterators
        i1, i2, i3, i4, i5 = tee(csv_reader, 5)
        # personal info: name, club, pos
        names = np.array([[data[0]] for data in i1])
        clubs = np.array([[data[1]] for data in i2])
        positions = np.array([[data[2]] for data in i3])
        # features 1-5 0-100
        features = np.array([[float(d) for d in data[4:7] + data[7:]] for data
in i4])
        # prices
        prices = []
        [prices.extend([float(d) if '0Value' not in str(d) else float(0)
for d in data[3:4]]) for data in i5]
        prices = np.asarray(prices)
        return names, clubs, positions, features, prices

```

Б.6 Текст файла datatools.py

```
# datatools.py module for data preprocessing before data analysis and NN
# learning

import numpy as np
import sampleinfo as si

def normalize(features):
    """Normalization or Min-Max Scaling"""
    x_min = np.min(features, axis=0)
    x_max = np.max(features, axis=0)
    # get values from 0 to 1
    return (features - x_min) / (x_max - x_min)

def normalize_z(features):
    """Standardization or Z-Score Normalization"""
    # simple mean value of each feature
    mu = np.mean(features, axis=0)
    # standard deviation (with N in denominator, not N-1)
    sigma = np.std(features, axis=0)
    return (features - mu) / sigma

def shuffle_data(features, prices):
    """ Shuffle features """
    np.random.seed(0) # To maintain consistency across runs
    perm = np.random.permutation(si.NUM_TOTAL)
    temp_features = np.zeros_like(features)
    temp_prices = np.zeros_like(prices)
    for i in range(len(perm)):
        temp_features[i, :] = features[perm[i], :]
        temp_prices[i] = prices[perm[i]]
    return temp_features, temp_prices

def split_data(features, prices):
    """ Separate into training and test """
    xtr = features[:si.NUM_TRAIN, :]
    ytr = prices[:si.NUM_TRAIN]
    xte = features[si.NUM_TRAIN:, :]
    yte = prices[si.NUM_TRAIN:]
    return xtr, ytr, xte, yte
```

Б.7 Текст файла drawgraphs.py

```
# drawgraphs.py module for some graphs

import matplotlib.pyplot as plt
import sampleinfo as si

def draw_price_number_graph(prices):
    plt.figure(figsize=(9, 10))
```

```

plt.hist(prices, edgecolor='black')
plt.title('Ціновий розподіл', fontsize=24)
plt.xlabel('€', fontsize=19)
plt.ylabel('N', fontsize=19)
plt.show()

def draw_pirson_graph(pirson):
    plt.figure(figsize=(10, 7))
    plt.barh(si.FEATURES_AND_PRICE_UKR, pirson, color='slateblue')
    plt.title('Коефіцієнт кореляції Пірсона', fontsize=24)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=8)
    plt.show()

def draw_features_correlation(x, y, title = 'Correlation'):
    plt.plot(x, y, 'r.')
    plt.title(title)
    plt.xlabel('Independent ')
    plt.ylabel('Dependent ')
    plt.show()
    plt.close()

```

Б.8 Текст файлу model.py

```

# model.py script for model trainging and testing

import csv_reader
import datatools as dt
import numpy as np

import os
import matplotlib.pyplot as plt
from keras.models import load_model

from sklearn.metrics import r2_score

names, clubs, positions, features, prices =
csv_reader.read_dataset('/dataset_Overall_Defending.csv')

features = dt.normalize(features)
# natural logarithm
prices = np.log(prices)

features, prices = dt.shuffle_data(features, prices)
xtr, ytr, xte, yte = dt.split_data(features, prices)

# no Reaction, Ball Control, Potential
# features = np.concatenate((features[:,3], features[:,4:13],
#                             features[:,14:16], features[:,17:]), axis=1)

import rendermodel as rm

# LOOP FOR HIDDEN LAYER SIZE
# result = []
# for i in range(1,50):

```

```

#         hidden_size = i*25

#         model = rm.model(xtr, ytr, num_epoch=30, batch_size=20, hidden_size=50)
#         pte = model.predict(xte)
#         r2 = r2_score(yte,pte)
#         result.append([r2, hidden_size])

model = rm.model(xtr, ytr, num_epoch=30, batch_size=20, hidden_size=625)

#                                     model
load_model(os.path.dirname(os.path.realpath(__file__))+'/ready_models/Test_Model.h5')

ptr = model.predict(xtr)
pte = model.predict(xte)

plt.plot(yte,pte,'r.')
plt.title('Тестова вибірка')
plt.xlabel('Фактичне')
plt.ylabel('Прогнозоване')

r21 = r2_score(ytr,ptr)
r2 = r2_score(yte,pte)

```

Б.9 Текст файлу ApplicationUserManager.cs

```

using System.Collections.Generic;
using System.Linq;
using System.Windows;

using PlayerService;
using Model;

namespace TransferMarketApp.Service
{
    public class ApplicationUserManager
    {
        private static ApplicationUserManager instance;

        private static PlayerServiceClient playerClient;

        private ApplicationUserManager()
        {
            nationalityConverter = new Nationality(new Dictionary<string,
string>());
            ClubsNames = new List<string>();
            Players = new List<Player>();
        }
        public static ApplicationUserManager getInstance()
        {
            if (instance == null)
                instance = new ApplicationUserManager();
            return instance;
        }

        public bool SetUser(BaseUser user)
        {
            if (User != null)
                return false;
        }
    }
}

```

```

        User = user;

        try
        {
            playerClient = new PlayerServiceClient();
            // read nation
            nationalityConverter = new
Nationality(playerClient.ReadNationalitiesAsync().Result);
            // read clubs
            ClubsNames =
playerClient.ReadClubsNamesAsync().Result.ToList();
            // read players
            playerClient.ReadPlayersAsync().Result.ToList().ForEach(el =>
Players.Add(new Player(el)));
            playerClient.Abort();
        }
        catch
        {
            MessageBox.Show("Error server!");
        }

        return true;
    }

    public bool IsAdmin =>
        User.GetType() == typeof(Admin);
    public BaseUser User { private set; get;}

    public Nationality nationalityConverter;
    public List<string> ClubsNames;
    public List<Player> Players;

    public void UpdatePlayers()
    {
        try
        {
            playerClient = new PlayerServiceClient();
            Players.Clear();
            playerClient.ReadPlayersAsync().Result.ToList().ForEach(el =>
Players.Add(new Player(el)));
            playerClient.Abort();
        }
        catch { }
    }

    public bool UpdatePlayer(int id, string name, int age, string
nationality, string club, string position, string status)
    {
        try
        {
            playerClient = new PlayerServiceClient();
            bool ans = playerClient.UpdatePlayerAsync(id, name, age,
nationality, club, position, status).Result;
            playerClient.Abort();
            return ans;
        }
        catch
        {
            return false;
        }
    }

    public List<double> ReadRating(int userId, int playerId)

```

```

    {
        try
        {
            playerClient = new PlayerServiceClient();
            List<double> ans = playerClient.ReadRatingAsync(userId,
playerId).Result.ToList();
            playerClient.Abort();
            return ans;
        }
        catch
        {
            return new List<double>();
        }
    }
    public bool UpdateInsertRating(int userId, int playerId, List<double>
rating)
    {
        try
        {
            playerClient = new PlayerServiceClient();
            bool ans = playerClient.UpdateInsertRatingAsync(userId,
playerId, rating.ToArray()).Result;
            playerClient.Abort();
            return ans;
        }
        catch
        {
            return false;
        }
    }

    public double Predict(List<double> features)
    {
        try
        {
            playerClient = new PlayerServiceClient();
            double ans =
playerClient.PredictAsync(features.ToArray()).Result;
            playerClient.Abort();
            return ans;
        }
        catch
        {
            return 0.0;
        }
    }
}

```

Б.10 Текст файла MainWindowVM.cs

```

using System.Windows;
using System.Windows.Controls;

namespace TransferMarketApp.ViewModels.WindowsVM
{
    public class MainWindowVM : ViewModelBase
    {
        public MainWindowVM()
        {

```

```

        OpenMenu();
        InitPages();

        CurrentPage = PlayerBoardPage;
    }

    private Visibility _visibility_menu;
    private int _column_frame;
    private int _column_frame_span;

    public Visibility VisibilityMenu
    {
        get { return _visibility_menu; }
        set
        {
            _visibility_menu = value;
            OnPropertyChanged(nameof(VisibilityMenu));
        }
    }
    public int ColumnFrame
    {
        get { return _column_frame; }
        set
        {
            _column_frame = value;
            OnPropertyChanged(nameof(ColumnFrame));
        }
    }
    public int ColumnFrameSpan
    {
        get { return _column_frame_span; }
        set
        {
            _column_frame_span = value;
            OnPropertyChanged(nameof(ColumnFrameSpan));
        }
    }
    public View.Pages.PlayerBoardPage PlayerBoardPage { set; get; }

    private Page _current_page;
    public Page CurrentPage
    {
        get { return _current_page; }
        set
        {
            _current_page = value;
            OnPropertyChanged(nameof(CurrentPage));
        }
    }

    public RelayCommand ExitCommand
    {
        get { return new RelayCommand(obj => {
Application.Current.Shutdown(); }); }
    }
    public RelayCommand MaximizeCommand
    {
        get
        {
            return new RelayCommand(obj =>
            {
                Window this_window = (Window)obj;

```

```

        this_window.WindowState = this_window.WindowState ==
WindowState.Normal ? WindowState.Maximized : WindowState.Normal;
    });
    }
}
public RelayCommand MinimizeCommand
{
    get
    {
        return new RelayCommand(obj =>
        {
            Window this_window = (Window)obj;
            this_window.WindowState = WindowState.Minimized;
        });
    }
}
public RelayCommand OpenCloseMenuCommand
{
    get
    {
        return new RelayCommand(obj =>
        {
            if (VisibilityMenu == Visibility.Visible) CloseMenu();
            else OpenMenu();
        });
    }
}
public RelayCommand OpenPlayerBoardPagePageCommand
{
    get { return new RelayCommand(obj => { CurrentPage =
PlayerBoardPage; }); }
}

private void OpenMenu()
{
    VisibilityMenu = Visibility.Visible;
    ColumnFrame = 1;
    ColumnFrameSpan = 0;
}
private void CloseMenu()
{
    VisibilityMenu = Visibility.Hidden;
    ColumnFrame = 0;
    ColumnFrameSpan = 2;
}
private void InitPages()
{
    PlayerBoardPage = new View.Pages.PlayerBoardPage();
}
}
}

```

Б.11 Текст файла ButtonRotateAnimation.cs

```

using System;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Threading;

```

```

namespace TransferMarketApp.ViewModels.Animations
{
    public class ButtonRotateAnimation : ViewModelBase
    {
        public ButtonRotateAnimation()
        {
            dispatcherTimerRole = new DispatcherTimer() { Interval = new
TimeSpan(0, 0, 0, 0, 10) };
            dispatcherTimerRole.Tick += new EventHandler(RotateAnimation);
            ButtonEnability = true;
        }

        private DispatcherTimer dispatcherTimerRole;

        private int _image_angle;
        private bool _button_enability;

        public void AnimationStart()
        {
            dispatcherTimerRole.Start();
            ButtonEnability = false;
        }
        public void AnimationStop()
        {
            dispatcherTimerRole.Stop();
            ButtonEnability = true;
        }
        public async void AnimationSeconds(int millisecondTimeout)
        {
            await Task.Run(() =>
            {
                AnimationStart();
                Thread.Sleep(millisecondTimeout);
                AnimationStop();
            });
        }

        public int ImageAngle
        {
            get { return _image_angle; }
            private set
            {
                _image_angle = value;
                OnPropertyChanged(nameof(ImageAngle));
            }
        }
        public bool ButtonEnability
        {
            get { return _button_enability; }
            private set
            {
                _button_enability = value;
                OnPropertyChanged(nameof(ButtonEnability));
            }
        }

        private void RotateAnimation(object sender, EventArgs e) => ImageAngle
+= 10;
        }
    }
}

```

Б.12 Текст файла PlayerBoardVM.cs

```

using System;
using System.Collections.ObjectModel;
using System.Windows;
using TransferMarketApp.ViewModels.PagesVM.PlayerBoardVM.PlayerRatingTypes;
using TransferMarketApp.Service;
using System.Linq;
using System.Threading.Tasks;

namespace TransferMarketApp.ViewModels.PagesVM.PlayerBoardVM
{
    public class PlayerBoardVM : ViewModelBase
    {
        public PlayerPersonalTab PlayerPersonalTab { get; set; }
        public PlayerRatingTab PlayerRatingTab { get; set; }

        public ObservableCollection<PlayerItem> Players { get; set; }

        private PlayerItem _selected_player_item;
        public PlayerItem SelectedPlayerItem
        {
            get { return _selected_player_item; }
            set
            {
                _selected_player_item = value;
                PlayerPersonalTab.Update(_selected_player_item);

                PlayerRatingTab.SetPlayerRatingTab(ApplicationUserManager.GetInstance().ReadRating(ApplicationUserManager.GetInstance().User.Id,
                    _selected_player_item.Id),
                    _selected_player_item.Id);

                OnPropertyChanged(nameof(SelectedPlayerItem));
            }
        }

        public PlayerBoardVM()
        {
            PlayerPersonalTab = new PlayerPersonalTab();
            PlayerRatingTab = new PlayerRatingTab();

            Players = new ObservableCollection<PlayerItem>();

            ApplicationUserManager.GetInstance().Players.ForEach(player =>
                Players.Add(new PlayerItem(player)));

            _search_player = "";
            _search_club = "";
            UpdateButtonAnimation = new Animations.ButtonRotateAnimation();
        }

        public string UserLabel => string.Format("{0}",
            ApplicationUserManager.GetInstance().IsAdmin ? "Admin" : "User");
        public string LoginLabel => string.Format("Logged in as: {0}",
            ApplicationUserManager.GetInstance().User.Login);
        public Animations.ButtonRotateAnimation UpdateButtonAnimation {
            private set; get; }
    }
}

```

```

private bool _sort_by_nation;
private bool _sort_by_position;
private bool _sort_by_name;
private bool _sort_by_age;
private bool _sort_by_status;

public bool SortByNation
{
    get { return _sort_by_nation; }
    set
    {
        _sort_by_nation = value;
        if (value) SortPlayers(value => value.Nationality);
        OnPropertyChanged(nameof(SortByNation));
    }
}
public bool SortByPosition
{
    get { return _sort_by_position; }
    set
    {
        _sort_by_position = value;
        if (value) SortPlayers(value => value.Position);
        OnPropertyChanged(nameof(SortByPosition));
    }
}
public bool SortByName
{
    get { return _sort_by_name; }
    set
    {
        _sort_by_name = value;
        if (value) SortPlayers(value => value.Name);
        OnPropertyChanged(nameof(SortByName));
    }
}
public bool SortByAge
{
    get { return _sort_by_age; }
    set
    {
        _sort_by_age = value;
        if (value) SortPlayers(value => value.Age.ToString());
        OnPropertyChanged(nameof(SortByAge));
    }
}
public bool SortByStatus
{
    get { return _sort_by_status; }
    set
    {
        _sort_by_status = value;
        if (value) SortPlayers(value => value.PlayerStatus);
        OnPropertyChanged(nameof(SortByStatus));
    }
}

private string _search_player;
public string SearchPlayer
{
    get { return _search_player; }
    set

```

```

    {
        _search_player = value;
        UpdatePlayers();
        OnPropertyChanged(nameof(SearchPlayer));
    }
}
private string _search_club;
public string SearchClub
{
    get { return _search_club; }
    set
    {
        _search_club = value;
        UpdatePlayers();
        OnPropertyChanged(nameof(SearchClub));
    }
}
private async void UpdatePlayers()
{
    await Task.Run(() =>
    {
        Players.ToList().ForEach(player =>
        {
            if
                (player.Name.ToUpper().Contains(SearchPlayer.ToUpper())           &&
                 player.Club.ToUpper().Contains(SearchClub.ToUpper()))
            {
                player.PlayerItemVisibility = Visibility.Visible;
            }
            else
            {
                player.PlayerItemVisibility = Visibility.Collapsed;
            }
        });
    });
}

private void SortPlayers(Func<PlayerItem, string> selector)
{
    ObservableCollection<PlayerItem> tempPlayers = new
ObservableCollection<PlayerItem>(Players);
    Players.Clear();
    tempPlayers.OrderBy(selector).ToList().ForEach(Players.Add);
}

public RelayCommand UpdateCommand => new RelayCommand(obj =>
{
    UpdateButtonAnimation.AnimationSeconds(500);

    Players.Clear();

    SearchClub = "";
    SearchPlayer = "";

    SortByNation = false;
    SortByPosition = false;
    SortByName = false;
    SortByAge = false;
    SortByStatus = false;

    ApplicationUserManager.GetInstance().UpdatePlayers();
}

```

```

        ApplicationUserManager.GetInstance().Players.ForEach(player =>
Players.Add(new PlayerItem(player)));
    });
}

```

Б.13 Текст файлу PlayerPersonalTab.cs

```

using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using TransferMarketApp.ViewModels.Validation;

namespace TransferMarketApp.ViewModels.PagesVM.PlayerBoardVM
{
    public class PlayerPersonalTab : ViewModelBase
    {
        public PlayerPersonalTab()
        {
            ClubsNames
Service.ApplicationUserManager.GetInstance().ClubsNames;
        }

        public List<string> ClubsNames { get; }

        #region controls enability
        public bool IsReadOnly =>
            !Service.ApplicationUserManager.GetInstance().IsAdmin;
        public bool IsEnable =>
            Service.ApplicationUserManager.GetInstance().IsAdmin;
        #endregion

        private string _name;
        private int _age;
        private string _nationality;

        [PlayerNameValidation]
        public string Name
        {
            get { return _name; }
            set
            {
                _name = value;
                UpdateResultText = "";
                OnPropertyChanged(nameof(Name));
            }
        }
        [PlayerNationalityValidation]
        public string Nationality
        {
            get { return _nationality; }
            set
            {
                _nationality = value;
                UpdateResultText = "";
                OnPropertyChanged(nameof(Nationality));
            }
        }
        [PlayerAgeValidation]

```

```

public int Age
{
    get { return _age; }
    set
    {
        _age = value;
        UpdateResultText = "";
        OnPropertyChanged(nameof(Age));
    }
}
// Club presentation in View is special. So there is no DataAnnotations.
private bool ClubValidate => ClubsNames.Any(club => club == _club);

private string _club;
public string Club
{
    get { return _club; }
    set
    {
        _club = value;
        if (!ClubValidate) ClubErrorText = "Such club does not exist";
        else ClubErrorText = "";
        UpdateResultText = "";
        OnPropertyChanged(nameof(Club));
    }
}
private string _club_error_text;
public string ClubErrorText
{
    get { return _club_error_text; }
    set
    {
        _club_error_text = value;
        OnPropertyChanged(nameof(ClubErrorText));
    }
}
private string _update_result_text;
public string UpdateResultText
{
    get { return _update_result_text; }
    set
    {
        _update_result_text = value;
        OnPropertyChanged(nameof(UpdateResultText));
    }
}
// Button Commands
public RelayCommand SavePlayerCommand
{
    get
    {
        return new RelayCommand(obj =>
        {
            if (selectedPlayerItem != null)
            {
                SetOriginFromTab();
            }
        });
    }
}
public RelayCommand CancelPlayerCommand
{

```

```

get
{
    return new RelayCommand(obj =>
    {
        if (selectedPlayerItem != null)
        {
            SetTabFromOrigin();
        }
    });
}

}

public PlayerItem selectedPlayerItem { private set; get; }

// update PlayerPersonalTab by new player
public void Update(PlayerItem playerItem)
{
    selectedPlayerItem = playerItem;
    SetTabFromOrigin();
}

private void SetTabFromOrigin()
{
    Name = selectedPlayerItem.Name;
    Age = selectedPlayerItem.Age;
    Nationality = selectedPlayerItem.Nationality;
    Club = selectedPlayerItem.Club;
    SetPlayerPosition(selectedPlayerItem.Position);
    SetPlayerStatus(selectedPlayerItem.PlayerStatus);
}

private void SetOriginFromTab()
{
    if (!AllValidations(this))
    {
        UpdateResultText = "Update failure";
        return;
    }

    try
    {
        if (Service.ApplicationUserManager.GetInstance().
            UpdatePlayer(selectedPlayerItem.Id,      Name,      Age,
Nationality, Club, Position, PlayerStatus))
        {
            selectedPlayerItem.Name = Name;
            selectedPlayerItem.Age = Age;
            selectedPlayerItem.Nationality = Nationality;
            selectedPlayerItem.Club = Club;
            selectedPlayerItem.Position = Position;
            selectedPlayerItem.PlayerStatus = PlayerStatus;

            UpdateResultText = "Updated correctly";
            return;
        }
        UpdateResultText = "Update failure";
    }
    catch
    {
        UpdateResultText = "Server failure";
        return;
    }
}

```

```

// check if all parameters is correct
private static bool AllValidations(PlayerPersonalTab infoTabVM)
{
    return Validator.TryValidateObject(infoTabVM, new
ValidationContext(infoTabVM),
    new List<ValidationResult>(), true) && infoTabVM.ClubValidate;
}

#region Player status picker
public string PlayerStatus => IsOnTransfer ? "On transfer" : IsActive
? "Active" : IsBlocked ? "Blocked" : "Retired";
private void SetPlayerStatus(string playerStatus)
{
    switch (playerStatus)
    {
        case "On transfer": { IsOnTransfer = true; break; }
        case "Active": { IsActive = true; break; }
        case "Blocked": { IsBlocked = true; break; }
        default: { IsRetired = true; break; }
    }
}

private bool _is_on_transfer;
private bool _is_active;
private bool _is_blocked;
private bool _is_retired;
public bool IsOnTransfer
{
    get { return _is_on_transfer; }
    set
    {
        _is_on_transfer = value;
        OnPropertyChanged(nameof(IsOnTransfer));
        OnPropertyChanged(nameof(PlayerStatus));
    }
}
public bool IsActive
{
    get { return _is_active; }
    set
    {
        _is_active = value;
        OnPropertyChanged(nameof(IsActive));
        OnPropertyChanged(nameof(PlayerStatus));
    }
}
public bool IsBlocked
{
    get { return _is_blocked; }
    set
    {
        _is_blocked = value;
        OnPropertyChanged(nameof(IsBlocked));
        OnPropertyChanged(nameof(PlayerStatus));
    }
}
public bool IsRetired
{
    get { return _is_retired; }
    set
    {
        _is_retired = value;

```

```

        OnPropertyChanged(nameof(IsRetired));
        OnPropertyChanged(nameof(PlayerStatus));
    }
}
#endregion

#region Player position picker
public string Position => IsGk ? "GK" : IsDef ? "DF" : IsMdf ? "MF" :
"FW";

private void SetPlayerPosition(string position)
{
    switch (position)
    {
        case "GK": { IsGk = true; break; }
        case "DF": { IsDef = true; break; }
        case "MF": { IsMdf = true; break; }
        default: { IsFw = true; break; }
    }
}

private bool _is_gk;
private bool _is_def;
private bool _is_mdf;
private bool _is_fw;
public bool IsGk
{
    get { return _is_gk; }
    set
    {
        _is_gk = value;
        OnPropertyChanged(nameof(IsGk));
        OnPropertyChanged(nameof(Position));
    }
}
public bool IsDef
{
    get { return _is_def; }
    set
    {
        _is_def = value;
        OnPropertyChanged(nameof(IsDef));
        OnPropertyChanged(nameof(Position));
    }
}
public bool IsMdf
{
    get { return _is_mdf; }
    set
    {
        _is_mdf = value;
        OnPropertyChanged(nameof(IsMdf));
        OnPropertyChanged(nameof(Position));
    }
}
public bool IsFw
{
    get { return _is_fw; }
    set
    {
        _is_fw = value;
        OnPropertyChanged(nameof(IsFw));
    }
}

```

```

        OnPropertyChanged(nameof(Position));
    }
}
#endregion
}
}

```

Б.14 Текст файла PlayerValidation.cs

```

using System.ComponentModel.DataAnnotations;
using System.Text.RegularExpressions;
using TransferMarketApp.Service;
using System.Linq;

namespace TransferMarketApp.ViewModels.Validation
{
    public class PlayerNameValidationAttribute : ValidationAttribute
    {
        public override bool IsValid(object value)
        {
            if (value is string playerName)
            {
                if (string.IsNullOrEmpty(playerName))
                {
                    ErrorMessage = "Name is required"; return false;
                }
                if (!Regex.IsMatch(playerName, @"^[a-zA-Z]"))
                {
                    ErrorMessage = "First symbol should be alphabetic"; return
false;
                }
                if (playerName.Where(char.IsLetter).ToList().Count < 2)
                {
                    ErrorMessage = "Name should contain minimum 2 alphabetic";
return false;
                }
                if (!Regex.IsMatch(playerName, @"^[a-zA-Z .'-]+$"))
                {
                    ErrorMessage = "Wrong character"; return false;
                }
                return true;
            }
            return false;
        }
    }

    public class PlayerNationalityValidationAttribute : ValidationAttribute
    {
        public override bool IsValid(object value)
        {
            if (value is string playerNationality)
            {
                if (string.IsNullOrEmpty(playerNationality))
                {
                    ErrorMessage = "Nationality is required"; return false;
                }
                if
(!ApplicationUserManager.GetInstance().nationalityConverter.Contains(playerNa
tionality))
                {

```

```

        ErrorMessage = "Such nationality does not exist"; return
false;
    }
    return true;
}
return false;
}
}
public class PlayerAgeValidationAttribute : ValidationAttribute
{
    public override bool IsValid(object value)
    {
        if (value is int playerAge)
        {
            if (playerAge > 45 || playerAge < 15)
            {
                ErrorMessage = "Age should be between 15 and 45"; return
false;
            }
            return true;
        };
        return false;
    }
}
}

```

Б.15 Текст файла NeuralNetwork.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace Model
{
    public class NeuralNetwork
    {
        private static System.Diagnostics.ProcessStartInfo start = new
System.Diagnostics.ProcessStartInfo();

        private static string exe_filename =
@"C:\Users\Sergey\AppData\Local\Programs\Python\Python39\python.exe";
        // приходится итерироваться с помощью ..\ в зависимости откуда
запускается программа
        private static string script_filename = @"..\..\..\..\transfer-market-
model\finalpredict.py";

        public NeuralNetwork()
        {
            //python interpreter location
            start.FileName = exe_filename;
            start.UseShellExecute = false; // Do not use OS shell
            start.CreateNoWindow = true; // We don't need new window
            start.RedirectStandardOutput = true; // Any output, generated by
application will be redirected back
            start.RedirectStandardError = true; // Any error in standard output
will be redirected back (for example exceptions)
            start.LoadUserProfile = true;

```

```

    }

    public double Predict(List<double> x_predict)
    {
        string x_predict_string = string.Join(" ", x_predict).Replace(" ",
        ".").Replace(" ", ",");

        // pass arguments to.py
        start.Arguments = string.Format("{0} {1}", script_filename,
        x_predict_string);

        using (System.Diagnostics.Process process =
        System.Diagnostics.Process.Start(start))
        {
            using (StreamReader reader = process.StandardOutput)
            {
                string stderr = process.StandardError.ReadToEnd(); // Here
are the exceptions from Python script
                string result = reader.ReadToEnd(); // Here is the result
of StdOut

                double r = 0.0;
                try
                {
                    r = double.Parse(result,
                    System.Globalization.CultureInfo.InvariantCulture);
                }
                catch { }

                return r;
            }
        }
    }
}

```

Б.16 Текст файлу CsvDataReader.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.IO;
using LumenWorks.Framework.IO.Csv;
using System.Linq;
using System.Globalization;

namespace DataLevel
{
    public static class CsvDataReader
    {
        private const string dataSet =
        @"..\..\..\DataLevel\Data\dataset.csv";
        private const string nationalities =
        @"..\..\..\DataLevel\Data\nationalities.csv";
        public static void GetUniqueSymbols(List<string> source_string)
        {
            List<char> all_symbols = new List<char>();
            source_string.ForEach(el => el.ToCharArray().ToList().ForEach(sy
=> all_symbols.Add(sy)));
        }
    }
}

```

```

        all_symbols.Distinct().Where(el =>
!Char.IsLetterOrDigit(el)).ToList().ForEach(el => Console.WriteLine(el));
    }

    private static List<string> ReadColumn(string filename, string
columnName)
    {
        List<string> data = new List<string>();

        var csvTable = new DataTable();
        using (var csvReader = new CsvReader(new
StreamReader(System.IO.File.OpenRead(filename)), true))
        {
            csvTable.Load(csvReader);

            foreach (DataRow row in csvTable.Rows)
            {
                data.Add(row[columnName].ToString());
            }
        }
        return data;
    }

    public static List<string> ReadClubs(string filename = dataSet) =>
        ReadColumn(filename, "Club").Distinct().ToList();
    public static List<string> ReadPositions(string filename = dataSet) =>
        ReadColumn(filename, "Position").Distinct().ToList();
    public static List<(string, string)> ReadNationalities(string filename
= nationalities)
    {
        var names = ReadColumn(filename,
"Nationality").Distinct().ToList();
        var images = ReadColumn(filename, "Image").Distinct().ToList();

        List<(string, string)> nationalities = new List<(string,
string)>();
        for(int i = 0; i < names.Count; i++)
        {
            nationalities.Add((names[i], images[i]));
        }
        return nationalities;
    }

    public static List<(string, string, string, double)>
ReadPlayers(string filename = dataSet)
    {
        List<(string, string, string, double)> player =
        new List<(string, string, string, double)>();

        var csvTable = new DataTable();
        using (var csvReader = new CsvReader(new
StreamReader(System.IO.File.OpenRead(filename)), true))
        {
            csvTable.Load(csvReader);

            foreach (DataRow row in csvTable.Rows)
            {
                player.Add((
                    row["Name"].ToString(),
                    row["Club"].ToString(),
                    row["Position"].ToString(),

```

```

        Convert.ToDouble(row["Price"],
CultureInfo.InvariantCulture)
    ));
    }
}

return player;
}
}
}

```

Б.17 Текст файла PlayerRatingItem.cs

```

using System;
using System.Collections.Generic;

namespace
TransferMarketApp.ViewModels.PagesVM.PlayerBoardVM.PlayerRatingTypes
{
    public abstract class PlayerRatingItem : ViewModelBase
    {
        public const int FEATURES_START = 4;
        public const int FEATURES_END = 32;

        public const int PRICE = 0;

        public const int INTERNATIONAL_REPUTATION = 1;
        public const int SKILL_MOVES = 2;
        public const int WEAK_FOOT = 3;

        public abstract int Value { set; get; }
        public abstract string Name { set; get; }
        public abstract string Description { set; get; }

        public abstract void Reset();

        protected Dictionary<int, (string, string)> RatingInfo = new
Dictionary<int, (string, string)>()
        {
            [1] = ("International reputation", "a way to boost player's overall
depending on his popularity IRL"),
            [2] = ("Skill moves", "separate decent players from great ones"),
            [3] = ("Weak foot", "specifies the shot power and ball control for
the other foot of that player than his preferred"),

            [4] = ("Potential", "maximum overall rate that player can reach"),

            [5] = ("Crossing", "player's speed in walking and running"),
            [6] = ("Finishing", "the scoring ability of a player"),
            [7] = ("Heading Accuracy", "player's accuracy when using the head
to pass, shoot or clear the ball"),
            [8] = ("Short Passing", "player's accuracy for the short passes"),
            [9] = ("Volleys", "player's ability for performing volleys"),

            [10] = ("Dribbling", "player's ability to carry the ball and past
an opponent while being in control"),
            [11] = ("Curve", "player's ability to curve the ball"),

```

```

        [12] = ("Free Kick Accuracy", "player's accuracy for taking the
Free Kicks"),
        [13] = ("Long Passing", "player's accuracy for the long and aerial
passes"),
        [14] = ("Ball Control", "the ability of a player to control the
ball"),

        [15] = ("Acceleration", "the increment of a player's running speed
on the pitch"),
        [16] = ("Sprint Speed", "speed rate of a player's sprinting"),
        [17] = ("Agility", "the speed and the grace of a player in term of
the ball control"),
        [18] = ("Reactions", "the acting speed of a player in response to
the situations happening around them"),
        [19] = ("Balance", "the even distribution of enabling a player to
remain upright and steady"),

        [20] = ("Shot Power", "the strength of a player's shootings"),
        [21] = ("Jumping", "player's ability and the quality for jumping
from the surface for headers"),
        [22] = ("Stamina", "player's ability to sustain prolonged physical
or mental effort in a match"),
        [23] = ("Strength", "the quality or state of being physically
strong of a player"),
        [24] = ("Long Shots", "player's accuracy for the shots taking from
long distances"),

        [25] = ("Aggression", "the aggression level of a player when they
do push/pull and tackle"),
        [26] = ("Interceptions", "player's capability to catch the opposing
team's passes"),
        [27] = ("Positioning", "how well a player is able to perform the
positioning on the field"),
        [28] = ("Vision", "player's mental awareness about his teammates'
positioning, for passing the ball to them"),
        [29] = ("Penalties", "player's accuracy for the shots taking from
the penalty kicks"),
        [30] = ("Composure", "determines at what distance the player with
the ball starts feeling the pressure from the opponent"),

        [31] = ("Defending", "player's ability to defend")
    };

    protected int ConvertRating(double rating)
    {
        try
        {
            return Convert.ToInt32(rating);
        }
        catch
        {
            return 100;
        }
    }
}

```

Б.18 Текст файла PlayerRatingItemBar.cs

```

using System;
using System.Collections.Generic;
using System.Text;

namespace
TransferMarketApp.ViewModels.PagesVM.PlayerBoardVM.PlayerRatingTypes
{
    public class PlayerRatingItemBar : PlayerRatingItem
    {
        private string _margin;
        public string Margin
        {
            get { return _margin; }
            set
            {
                _margin = value;
                OnPropertyChanged(nameof(Margin));
            }
        }

        public PlayerRatingItemBar(double rating, int index)
        {
            Value = ConvertRating(rating);
            Name = RatingInfo[index].Item1;
            Description = RatingInfo[index].Item2;

            Margin = separatedItemsIndexies.Contains(index) ? "0,35,0,0" :
"0";
        }

        private int _value;
        private string _name;
        private string _description;
        public override int Value
        {
            get { return _value; }
            set
            {
                _value = value;
                OnPropertyChanged(nameof(Value));
            }
        }
        public override string Name
        {
            get { return _name; }
            set
            {
                _name = value;
                OnPropertyChanged(nameof(Name));
            }
        }
        public override string Description
        {
            get { return _description; }
            set
            {
                _description = value;
                OnPropertyChanged(nameof(Description));
            }
        }
    }
}

```

```

    }
}
public override void Reset() => Value = 0;

private static List<int> separatedItemsIndexies = new List<int>()
{
    4,9,14,19,24,30
};
}
}

```

Б.19 Текст файла PlayerRatingEvaluation.cs

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace
TransferMarketApp.ViewModels.PagesVM.PlayerBoardVM.PlayerRatingTypes
{
    public class PlayerRatingEvaluation : ViewModelBase
    {
        public PlayerRatingEvaluation()
        {
            PredictButtonAnimation = new Animations.ButtonRotateAnimation();
        }

        // Button animation
        public Animations.ButtonRotateAnimation PredictButtonAnimation {
private set; get; }

        public async void GetPrediction(List<double> features)
        {
            PredictButtonAnimation.AnimationStart();

            double answer = 0.0;
            await Task.Factory.StartNew(() =>
            {
                answer
Service.ApplicationUserManager.GetInstance().Predict(features);
            });

            MarketValue = answer;
            EvaluationDate = DateTime.Now;

            PredictButtonAnimation.AnimationStop();
        }

        private double _market_value;
        public double MarketValue
        {
            get { return _market_value; }
            set
            {
                _market_value = value;
                OnPropertyChanged(nameof(MarketValue));
                OnPropertyChanged(nameof(MarketValueString));
            }
        }
    }
}

```

```

    }

    private DateTime _evaluation_date;
    public DateTime EvaluationDate
    {
        get { return _evaluation_date; }
        private set
        {
            _evaluation_date = value;
            OnPropertyChanged(nameof(EvaluationDate));
            OnPropertyChanged(nameof(EvaluationDateString));
        }
    }

    public string MarketValueString => string.Format("{0} \u00A3",
MarketValue);
    public string EvaluationDateString => EvaluationDate ==
DateTime.MinValue ? "" :
    "Evaluated for " + EvaluationDate.ToString("d");
}
}

```

Б.20 Текст файла PlayerRatingData.cs

```

using System;
using System.Collections.Generic;
using System.Data.SqlClient;

namespace DataLevel
{
    public partial class DataBaseLoad : DataBase
    {
        public List<double> ReadRating(int userId, int playerId)
        {
            List<double> rating = new List<double>() { 0.0, 1.0, 1.0, 1.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0 };

            try
            {
                OpenConnection();
                string query = "SELECT * FROM Rating WHERE UserId=" + userId +
" AND PlayerId=" + playerId + ";";

                SqlCommand command = new SqlCommand(query, sqlconnection);
                SqlDataReader result = command.ExecuteReader();

                if (result.HasRows)
                {
                    while (result.Read())
                    {
                        try
                        {
                            rating.Clear();
                            #region Initialization features

                            rating.Add(Convert.ToDouble(result["Price"]));

```

```

reputation"]));
rating.Add(Convert.ToDouble(result["International
moves"]));
rating.Add(Convert.ToDouble(result["Skill
foot"]));

rating.Add(Convert.ToDouble(result["Potential"]));

rating.Add(Convert.ToDouble(result["Crossing"]));

rating.Add(Convert.ToDouble(result["Finishing"]));
rating.Add(Convert.ToDouble(result["Heading
Accuracy"]));
rating.Add(Convert.ToDouble(result["Short
Passing"]));
rating.Add(Convert.ToDouble(result["Volleys"]));

rating.Add(Convert.ToDouble(result["Dribbling"]));
rating.Add(Convert.ToDouble(result["Curve"]));
rating.Add(Convert.ToDouble(result["Free Kick
Accuracy"]));
rating.Add(Convert.ToDouble(result["Long
Passing"]));
rating.Add(Convert.ToDouble(result["Ball
Control"]));

rating.Add(Convert.ToDouble(result["Acceleration"]));
rating.Add(Convert.ToDouble(result["Sprint
Speed"]));
rating.Add(Convert.ToDouble(result["Agility"]));

rating.Add(Convert.ToDouble(result["Reactions"]));
rating.Add(Convert.ToDouble(result["Balance"]));

rating.Add(Convert.ToDouble(result["Shot
Power"]));
rating.Add(Convert.ToDouble(result["Jumping"]));
rating.Add(Convert.ToDouble(result["Stamina"]));
rating.Add(Convert.ToDouble(result["Strength"]));
rating.Add(Convert.ToDouble(result["Long
Shots"]));

rating.Add(Convert.ToDouble(result["Aggression"]));

rating.Add(Convert.ToDouble(result["Interceptions"]));

rating.Add(Convert.ToDouble(result["Positioning"]));
rating.Add(Convert.ToDouble(result["Vision"]));

rating.Add(Convert.ToDouble(result["Penalties"]));

rating.Add(Convert.ToDouble(result["Composure"]));

rating.Add(Convert.ToDouble(result["Defending"]));

#endregion

```

```

        }
        catch (IndexOutOfRangeException)
        {
            result.Close();
            return rating;
        }
    }

    result.Close();
    return rating;
}
catch
{
    return rating;
}
finally
{
    CloseConnection();
}
}
public bool UpdateInsertRating(int userId, int playerId, List<double>
rating)
{
    try
    {
        OpenConnection();

        string condition = "IF EXISTS (SELECT Rating.Id FROM Rating
WHERE Rating.UserId = " + userId + " AND Rating.PlayerId = " + playerId + ")";
        string action1 = "\nBEGIN\n" +
            "UPDATE Rating\n" +
            "SET[Price] = " + rating[0] + ", [International
reputation]= " + rating[1] + ", [Skill moves]= " + rating[2] + ", [Weak foot]=
" + rating[3] + "," +
            " [Potential]= " + rating[4] + ", [Crossing]= " + rating[5]
+ ", [Finishing]= " + rating[6] + "," +
            " [Heading Accuracy]= " + rating[7] + ", [Short Passing]=
" + rating[8] + ", [Volleys]= " + rating[9] + ", [Dribbling]= " + rating[10] +
", [Curve]= " + rating[11] + "," +
            " [Free Kick Accuracy]= " + rating[12] + ", [Long
Passing]= " + rating[13] + ", [Ball Control]= " + rating[14] + ", [Acceleration]=
" + rating[15] + ", [Sprint Speed]= " + rating[16] + "," +
            " [Agility]= " + rating[17] + ", [Reactions]= " +
rating[18] + ", [Balance]= " + rating[19] + ", [Shot Power]= " + rating[20] +
", " +
            " [Jumping]= " + rating[21] + ", [Stamina]= " + rating[22]
+ ", [Strength]= " + rating[23] + ", [Long Shots]= " + rating[24] + ",
[Aggression]= " + rating[25] + ", " +
            " [Interceptions]= " + rating[26] + ", [Positioning]= " +
rating[27] + ", [Vision]= " + rating[28] + ", [Penalties]= " + rating[29] +
", " +
            " [Composure]= " + rating[30] + ", [Defending]= " +
rating[31] + "\n" +
            "WHERE Rating.UserId = " + userId + " AND Rating.PlayerId
= " + playerId + ";\n" +
            "END\n";

        string action2 = "ELSE\n" +
            "BEGIN\n" +

```

```

        "INSERT INTO Rating (UserId, PlayerId, [Price],
[International reputation], [Skill moves], [Weak foot], [Potential],
[Crossing], [Finishing]," +
        " [Heading Accuracy], [Short Passing], [Volleys],
[Drizzling], [Curve], [Free Kick Accuracy], [Long Passing], [Ball Control],
[Acceleration]," +
        " [Sprint Speed], [Agility], [Reactions], [Balance], [Shot
Power], [Jumping], [Stamina], [Strength], [Long Shots], [Aggression],
[Interceptions]," +
        " [Positioning], [Vision], [Penalties], [Composure],
[Defending])\n" +
        "VALUES(" + userId + ", " + playerId + ", " + rating[0] +
", " + rating[1] + ", " + rating[2] + ", " + rating[3] + ", " + rating[4] + ",
" + rating[5] + "," +
        " " + rating[6] + ", " + rating[7] + ", " + rating[8] + ",
" + rating[9] + ", " + rating[10] + ", " + rating[11] + ", " + rating[12] + ",
" + rating[13] + "," +
        " " + rating[14] + ", " + rating[15] + ", " + rating[16] +
", " + rating[17] + ", " + rating[18] + ", " + rating[19] + ", " + rating[20]
+ ", " + rating[21] + "," +
        " " + rating[22] + ", " + rating[23] + ", " + rating[24] +
", " + rating[25] + ", " + rating[26] + ", " + rating[27] + ", " + rating[28]
+ ", " + rating[29] + "," +
        " " + rating[30] + ", " + rating[31] + ");\n" +
        "END";

        string query = condition + action1 + action2;

        SqlCommand command = new SqlCommand(query, sqlconnection);

        if (command.ExecuteNonQuery() != 1) return false;

        return true;
    }
    catch (SqlException ex)
    {
        return false;
    }
    catch
    {
        return false;
    }
    finally
    {
        CloseConnection();
    }
}
}
}

```

Б.21 Текст файлу PlayerData.cs

```

using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using ActionResult;
using System.Linq;

namespace DataLevel

```

```

{
    public partial class DataBaseLoad : DataBase
    {
        public List<(int, string, int, string, string, string, string, double)>
ReadPlayers()
        {
            List<(int, string, int, string, string, string, string, double)>
players = new List<(int, string, int, string, string, string, string,
double)>();

            try
            {
                OpenConnection();
                string query = "SELECT Players.Id AS Id, Players.Name AS Name,
Players.Age AS Age, Nationalities.Name AS Nationality, Clubs.Name AS Club,
Positions.Name AS Position, PlayerStatus.Status AS Status, Players.Price AS
Price " +
                            "FROM PlayerStatus INNER JOIN(Positions INNER JOIN (Clubs
INNER JOIN (Players LEFT JOIN Nationalities ON Players.Nationality =
Nationalities.Id) ON Clubs.Id = Players.Club) " +
                            "ON Positions.Id = Players.Position) ON PlayerStatus.Id =
Players.Status";

                SqlCommand command = new SqlCommand(query, sqlconnection);
                SqlDataReader result = command.ExecuteReader();

                if (result.HasRows)
                {
                    while (result.Read())
                    {
                        try
                        {
                            players.Add((
                                Convert.ToInt32(result["Id"]),
                                Convert.ToString(result["Name"]),
                                Convert.ToInt32(result["Age"]),
                                Convert.ToString(result["Nationality"]),
                                Convert.ToString(result["Club"]),
                                Convert.ToString(result["Position"]),
                                Convert.ToString(result["Status"]),
                                Convert.ToDouble(result["Price"])
                            ));
                        }
                        catch (IndexOutOfRangeException)
                        {
                            result.Close();
                            return players;
                        }
                    }
                }

                result.Close();
                return players.OrderByDescending(player =>
player.Item8).Take(100).ToList();
            }
            catch
            {
                return players;
            }
            finally
            {
                CloseConnection();
            }
        }
    }
}

```

```

    }
}
public bool UpdatePlayer(int id, string name, int age, string
nationality, string club, string position, string status)
{
    int? nationalityId = GetId("Nationalities", "Name", nationality);
    if (nationalityId == null) return false;

    int? clubId = GetId("Clubs", "Name", club);
    if (clubId == null) return false;

    int? positionId = GetId("Positions", "Name", position);
    if (positionId == null) return false;

    int? statusId = GetId("PlayerStatus", "Status", status);
    if (statusId == null) return false;

    try
    {
        OpenConnection();

        string query = "UPDATE Players SET Name='" + name + "', Age='"
+ age + "', Nationality='"
        + nationalityId + "', Club='" + clubId + "', Position='" +
positionId + "', Status='" + statusId + "' WHERE Id LIKE '" + id + "' ";

        SqlCommand command = new SqlCommand(query, sqlconnection);

        if (command.ExecuteNonQuery() != 1) return false;

        return true;
    }
    catch (SqlException ex)
    {
        return false;
    }
    finally
    {
        CloseConnection();
    }
}

public DataActionResult InsertPlayer(string name, string club, string
position, double price = 0.0)
{
    int? clubId = GetId("Clubs", "Name", club);
    if (clubId == null) return DataActionResult.NO_CLUB;

    int? positionId = GetId("Positions", "Name", position);
    if (positionId == null) return DataActionResult.NO_POSITION;

    try
    {
        OpenConnection();

        string query = "INSERT INTO Players (Name, Club, Position,
Price) output INSERTED.Id VALUES (@name,@club,@position,@price)";
        SqlCommand command = new SqlCommand(query, sqlconnection);

        command.Parameters.AddWithValue("@name", name);
        command.Parameters.AddWithValue("@club", clubId);
        command.Parameters.AddWithValue("@position", positionId);
        command.Parameters.AddWithValue("@price", price);
    }
}

```

```

        command.ExecuteScalar();

        return DataActionResult.CORRECT;
    }
    catch (SqlException ex) when (ex.Number == 2627)
    {
        return DataActionResult.UNUNIQUE;
    }
    catch
    {
        return DataActionResult.GENERAL_ERROR;
    }
    finally
    {
        CloseConnection();
    }
}

public int? GetId(string tableName, string columnName, string
conditionName)
{
    try
    {
        int? id = null;
        OpenConnection();

        if (conditionName.Contains('\')) conditionName =
conditionName.Insert(conditionName.IndexOf('\'), "\\");

        string query = "SELECT Id FROM " + tableName + " WHERE " +
columnName + " LIKE '" + conditionName + "'";

        SqlCommand command = new SqlCommand(query, sqlconnection);
        SqlDataReader result = command.ExecuteReader();

        if (result.HasRows)
        {
            if (result.Read())
            {
                try
                {
                    id = Convert.ToInt32(result["Id"]);
                }
                catch
                {
                    result.Close();
                }
            }
        }
        result.Close();

        return id;
    }
    catch
    {
        return null;
    }
    finally
    {
        CloseConnection();
    }
}

```

```

    }
}
}

```

Б.22 Текст файлу Nationality.cs

```

using System.Collections.Generic;

namespace Model
{
    public class Nationality
    {
        private Dictionary<string, string> NationalityDictionary;
        private string defaultImage =
"/View/Images/CountryIcons/icons8_image_96px_1.png";

        public Nationality(Dictionary<string, string> nationality_dictionary)
        {
            NationalityDictionary = new Dictionary<string,
string>(nationality_dictionary);
        }

        public string GetImage(string nationality)
        {
            return NationalityDictionary.ContainsKey(nationality) ?
NationalityDictionary[nationality] : defaultImage;
        }
        public bool Contains(string nationality)
        {
            return NationalityDictionary.ContainsKey(nationality);
        }
        public Dictionary<string, string>.KeyCollection GetNames()
        {
            return NationalityDictionary.Keys;
        }
        public Dictionary<string, string>.ValueCollection GetImages()
        {
            return NationalityDictionary.Values;
        }
    }
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
кафедра програмних засобів

Дипломна кваліфікаційна робота

Програмне забезпечення підтримки прийняття рішень
про футбольну трансферну активність

Football Transfer Decision Support Software

Виконав
студент групи КНТ – 138

С. О. Ліпихін

Керівник
к.т.н., доцент

В. М. Льовкін

2022

Рисунок В.1 – Слайд №1

Вступ

Об'єкт дослідження – процес оцінювання трансферної вартості футболістів

Предмет дослідження – програмне забезпечення підтримки прийняття рішень про футбольну трансферну активність.

Мета роботи – створення програмного забезпечення підтримки прийняття рішень про футбольну трансферну активність щодо визначення ціни гравця на трансферному ринку для автоматизації цього процесу згідно з оцінкою його ігрових показників футбольними експертами.

Завдання:

- аналіз предметної області;
- дослідження методів і моделей оцінювання трансферної вартості футболіста;
- проектування програмного забезпечення;
- розробка програмного забезпечення.

Рисунок В.2 – Слайд №2

Актуальність розробки

Футбольна трансферна активність – це діяльність визначених акторів на футбольному трансферному ринку.

За останні 25 років:

- кількість трансферів збільшилася у 4 рази;
- обсяг трансферних витрат збільшився у 110 разів.

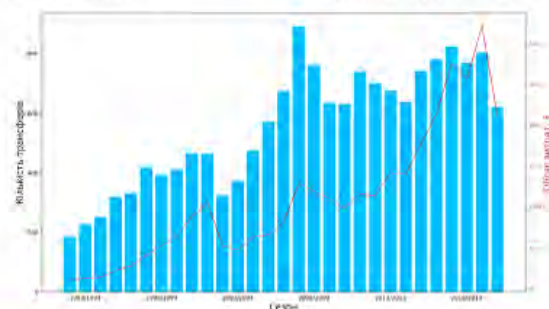


Рисунок В.3 – Слайд №3

Порівняння методів



Рисунок В.4 – Слайд №4

Пошук даних



	Назва	Опис	Шкала
X N = 34	Міжнародна репутація	Показник гравця в залежності від його популярності	1-5
	Дриблінг	Здатність гравця, контролюючи м'яч, провести його повз суперника	0-100
	Вачення поля	Обізнаність гравця про позиційну гру	0-100
	Самовладання	Відчуття спокою та контролю над своїми емоціями при поразках	0-100
Y	Ціна	Трансферна вартість	0-5 * 10 ⁸

Рисунок В.5 – Слайд №5

Нормалізація даних

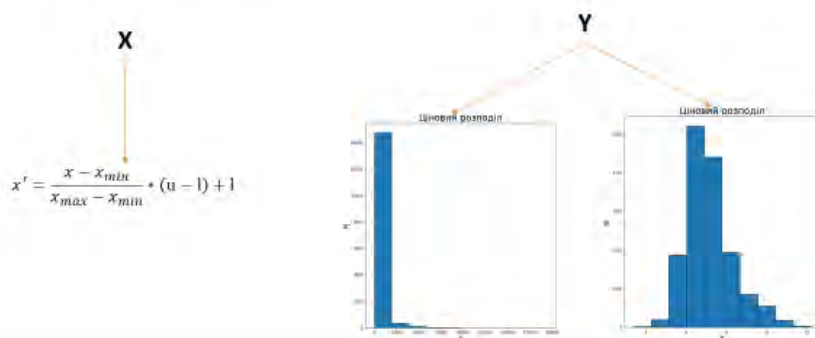


Рисунок В.6 – Слайд №6

Аналіз ознак

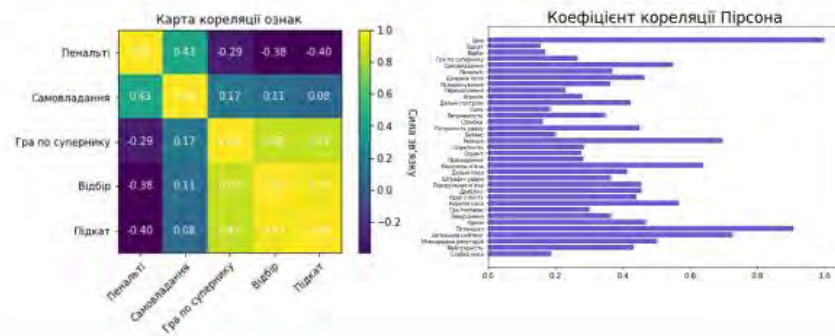


Рисунок В.7 – Слайд №7

Функціональність

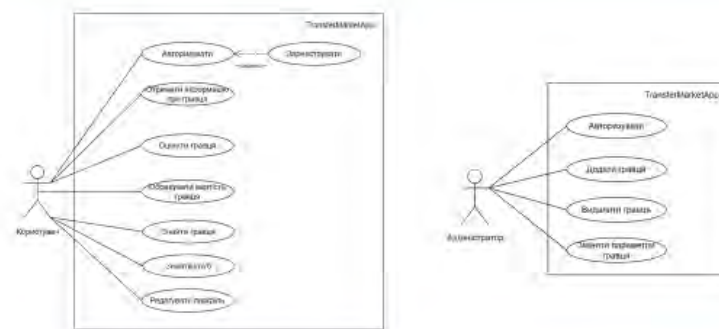


Рисунок В.8 – Слайд №8

Вибір мови програмування та IDE

Критерій	Python	C#	Критерій	Scala	Ruby
Вбудована функціональність	+	+	Дисперсія	+	±
Робота з даними	++	+	Кристалічність	+	+
Підвищення продуктивності	-	+	Інтерактивність	-	±
Створення графічного інтерфейсу	±	+	Висвітленість	+	±
Створення розмітаних інтерфейсів	±	+	Відслідковуваність	+	±
Створення та інтегрування бібліотек	-	+	Важкостворюваність	+	-
Підтримка клієнт-сервєрної архітектури	+	++	Інтерфейс	+	-
Робота під Windows	+	++	Доступність коду	+	-
Кросплатформність	++	-	Розумність коду	±	+
Машинне навчання	+	±	Aquifer Notebook	±	-
Візуалізація	++	-	Доступність інформації	+	+
Швидкість виконання	-	++	Паралельна робота	+	-
Простота та швидкість розробки	++	+	Паралельна графічна робота	+	-
Спільнота та документація	+	+	Паралельна робота	-	±
			Висвітленість	-	±
			Скористатися кодом	+	+

Рисунок В.9 – Слайд №9

Структура додатка

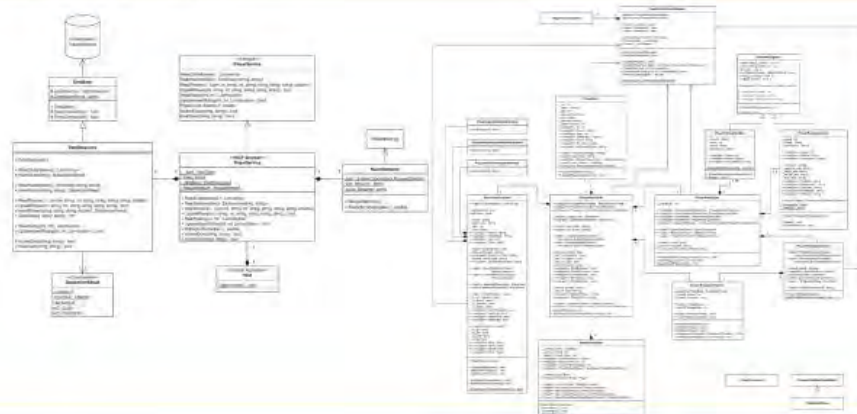


Рисунок В.13 – Слайд №13

Умови виконання програми

Файлова цілісність



Наявність залежностей

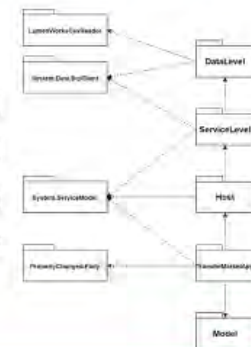


Рисунок В.14 – Слайд №14

Головне вікно

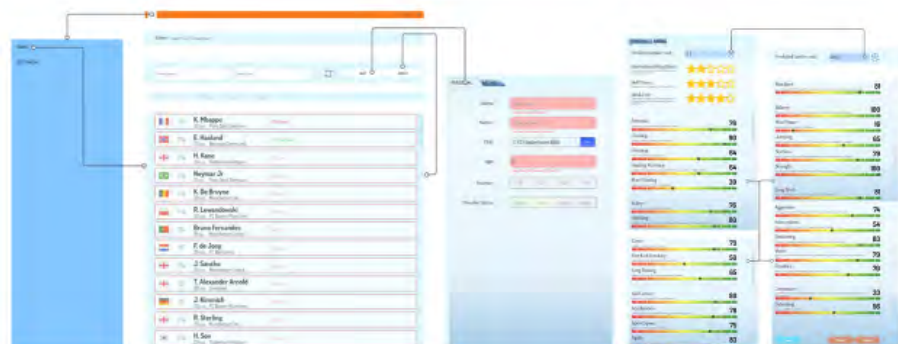


Рисунок В.15 – Слайд №15

Порівняння моделі

Критерій	Робота S. Deu	Робота M. Al-Asadi	Власна
Тип моделі	НМ класифікатор	Випадковий ліс	НМ для прогнозування
Вибір ознак	+	–	+
Існість	–	+	+

Рисунок В.19 – Слайд №19

Висновки

З метою створення програми для автоматизації процесу визначення ціни футболісту на трансферному ринку, було досліджено два основоположних методи оцінки вартості гравця – метод мудрості наповнута імовірлісна регресійна модель. Виходячи з їх спільних переваг, та цілей розроблюваного продукту, було прийнято рішення об'єднати два методи, генеруючи один новий.

В результаті розроблена модель характеризується:

- використанням нейронної мережі на основі показників футболіста;
- більшою гнучістю прогностичного рішення, ніж у моделях, що використовують суто об'єктивні дані;
- більшою точністю за аналогічн, виходячи з отриманого коефіцієнту детермінації;
- більш репрезентативним набором ознак, ніж у подібних моделях.

Для отриманої математичної моделі було створено програмну оболонку з графічним користувацьким інтерфейсом та застосовуючи клієнт-серверну архітектуру додатку.

Подальший супровід додатку вимагає створення повноцінної електронної системи обміну гравцями, на основі вартості, підрахованої моделлю.

Рисунок В.20 – Слайд №20