НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено
Завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системний аналіз і управління»
спеціальності 124 «Системний аналіз»
на тему: «Моделювання результатів тенісних матчів серед чоловіків методами машинного навчання»

Виконав:
Студент IV курсу, групи КА-92
Шум Кирил Ігорович _____

Керівник:
доц., канд. фіз.-мат. наук Каніовська І.Ю. _____

Консультант з економічного розділу:
доц., канд. економ. наук Рощина Н.В. _____

Консультант з нормоконтролю:
канд. фіз.-мат. наук Статкевич В.М. _____

Рецензент:
доц., канд. фіз.-мат. наук Ільєнко М.К. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ 2023

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
НН Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітня програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Оксана ТИМОЩУК
«__» травня 2023 р.

ЗАВДАННЯ
на дипломну роботу студента
Шума Кирила Ігоровича

1. Тема роботи «Моделювання результатів тенісних матчів серед чоловіків методами машинного навчання», керівник роботи доц., канд. фіз.-мат. наук Каніовська І.Ю. затверджені наказом по університету від «30» травня 2023 р. № 2065-с.

2. Термін подання студентом роботи: 12.06.2023.

3. Вихідні дані до роботи: набір даних, який містить історичні відомості про матчі, які вже відбулися. На основі цих даних буде натреновано прогностичні моделі.

4. Зміст роботи: Дослідження предметної області тенісу та реалізація програмного продукту для моделювання результатів тенісних матчів

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	доц., канд. економ. наук. Рощина Н.В.		

7. Дата видачі завдання: 17.04.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми ДР	17.04.2023-23.04.2023	виконано
2	Ознайомлення зі структурою БДР згідно з Положенням про державну атестацію студентів НТУУ «КПІ ім. І. Сікорського»	17.04.2023-23.04.2023	виконано
3	Ознайомлення з ДСТУ 3008-95 та стандарти ЄСПД	17.04.2023-30.04.2023	виконано
4	Проведення дослідження за темою БДР під керівництвом керівника	17.04.2023-30.04.2023	виконано
5	Завершення роботи над першим варіантом частини БДР	24.04.2023-14.05.2023	виконано
6	Проведення роботи над експериментальною частиною БДР	01.05.2023-14.05.2023	виконано
7	Проведення роботи над програмним продуктом	01.05.2023-14.05.2023	виконано
8	Оформлення БДР та аналіз отриманих результатів	15.05.2023-21.05.2023	виконано

Студент Кирил ШУМ

Керівник Ірина КАНІОВСЬКА

РЕФЕРАТ

Дипломна робота: 166 с., 24 табл., 17 рис., 2 додатки, 29 джерел.

ТЕНІС, ЧОЛОВІЧИЙ ТЕНІС, МАШИННЕ НАВЧАННЯ, МОДЕЛЮВАННЯ РЕЗУЛЬТАТІВ, ПРОГНОЗУВАННЯ, ЛОГІСТИЧНА РЕГРЕСІЯ, НЕЙРОННІ МЕРЕЖІ, МУЛТИШАРОВИЙ ПЕРСЕПТРОН, БІНАРНА КЛАСИФІКАЦІЯ.

Об'єкт дослідження: тенісні матчі серед чоловіків.

Предмет дослідження: методи машинного навчання для моделювання та прогнозування результатів тенісних матчів серед чоловіків.

Мета дослідження: створення програмного продукту, здатного моделювати результати тенісних матчів серед чоловіків на основі аналізу історичних даних та за допомогою застосування методів машинного навчання.

Використані моделі: у роботі було використано модель логістичної регресії та різновид нейронної мережі з прямим поширенням сигналу мультишаровий персептрон.

Отримані результати: розроблено програмний продукт, здатний моделювати результати тенісних матчів серед чоловіків, проведено його випробування на реальному турнірі.

У рамках подальшого дослідження планується оптимізувати та автоматизувати роботу програми, покращити алгоритм прогнозування статистики майбутніх матчів, а також розробити користувацький інтерфейс для зручності використання програми.

ABSTRACT

Thesis: 166 pages, 24 tables, 17 figures, 2 appendices, 29 references.

TENNIS, MEN`S TENNIS, MACHINE LEARNING, OUTCOMES MODELLING, PREDICTION, LOGISTIC REGRESSION, NEURAL NETWORKS, MULTILAYER PERCEPTRON, BINARY CLASSIFICATION.

Object of research: men's tennis matches.

Subject of research: machine learning methods for modelling and predicting results of men's tennis matches.

Purpose of research: to create a software product capable of modelling the results of men's tennis matches based on the analysis of historical data involving machine learning methods.

Models used: the study employed a logistic regression model and a feed forward neural network model known as a multilayer perceptron.

Results achieved: a software product capable of modelling the results of men's tennis matches was developed and tested in a real tournament.

Future research aims to optimize and automate the software, improve the algorithm for predicting future match statistics, and develop a user interface for ease of use.

ЗМІСТ

ВСТУП

Теніс – це не лише захоплюючий спорт, але й глобальний феномен, що об'єднує мільйони любителів по всьому світу. У сучасному спортивному світі теніс викликає неймовірний інтерес та сприймається як справжній витвір мистецтва. Професійні тенісні турніри залучають найкращих гравців з різних країн, а супровідні заходи включають у себе широку аудиторію глядачів, спонсорів та прогнозистів.

У світі, де дані стають все доступнішими, методи машинного навчання відіграють визначну роль у вирішенні складних задач, і прогнозування результатів тенісних матчів не є винятком. Використання цих методів може дати можливість розробити моделі, які враховують різноманітні фактори, такі як рейтинги гравців, їх фізичні дані, попередні результати, властивості навколишнього середовища тощо.

У такому контексті прогнозування результатів тенісних матчів набуває великого значення. По-перше, прогностичні моделі можуть бути використані в спортивній галузі, зокрема тренерами і гравцями, для аналізу та підготовки до змагань. Вони можуть допомогти виявити слабкі та сильні сторони гравців, розробити стратегії гри та визначити оптимальний підхід до підготовки.

По-друге, прогнозування результатів тенісних матчів може бути цінним інструментом у букмекерській галузі: як для букмекерських контор, так і для людей, що роблять ставки (бетторів). Для букмекерських контор моделі з високою точністю можуть допомогти встановлювати об'єктивні коефіцієнти ставок та забезпечувати більш справедливу гру для ставкодавців, а для бетторів у свою чергу робити ставки «не в сліпу», а враховуючи результати роботи моделей.

Крім того, прогнозування результатів тенісних матчів може застосовуватися в дослідженнях та аналізі спортивних тенденцій. Великі обсяги даних, отримані в результаті моделювання, можуть бути використані

для виявлення нових залежностей, трендів та особливостей в тенісі, що сприятиме подальшому розвитку спорту.

Об'єктом дослідження в даній роботі є процес моделювання результатів тенісних матчів, що базується на методах машинного навчання. В даній дипломній роботі буде проведений детальний аналіз тенісних матчів та факторів, що впливають на результати гри. Будуть використані методи машинного навчання для побудови моделі прогнозування, яка зможе враховувати складні взаємозв'язки між різними змінними.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Проблема прогнозування тенісних матчів є актуальною і викликає значний інтерес у спортивних аналітиків, гральній індустрії та фанатів тенісу. Теніс вважається одним з найбільш непередбачуваних видів спорту, де кожен матч може мати багато нюансів і факторів, які впливають на його результат. У зв'язку з цим, розробка ефективних методів прогнозування тенісних матчів є викликом для багатьох дослідників.

Однією з причин актуальності цієї проблеми є популярність тенісу як спорту. Теніс є одним з найбільш популярних і глобальних спортивних подій, з великою кількістю професійних турнірів. Багато людей цікавляться ставками на тенісні матчі або просто бажають передбачити результати змагань. Тому, розробка надійних моделей прогнозування стає важливою для задоволення потреб спортивних фанатів.

Другою причиною є складність самого тенісу як виду спорту. Теніс включає в себе багато змінних, які можуть впливати на результати матчів. Наприклад, форма гравців, їх фізична підготовка, психологічний стан, а також такі фактори, як покриття корту, погодні умови і стратегія гри. У тенісі також можуть відбуватися несподівані події, такі як травми гравців або стратегічні зміни під час матчу. Всі ці аспекти створюють складну і одночасно цікаву задачу прогнозування, де необхідно аналізувати багато даних та враховувати широкий спектр факторів.

Третя причина актуальності проблеми прогнозування тенісних матчів полягає в постійному розвитку технологій та аналітичних інструментів. Завдяки появі нових методів збору та аналізу даних, таких як машинне навчання, штучний інтелект та статистичні моделі, стало можливим зробити більш точні прогнози в спортивних подіях, включаючи теніс. Використання

цих технологій дозволяє враховувати широкий спектр факторів, таких як статистика, форма гравців, їхні персональні характеристики, показники змагань та інші важливі дані.

Крім того, прогнозування тенісних матчів має практичне значення в багатьох сферах. Наприклад, букмекерські компанії використовують прогнози, щоб встановити коефіцієнти на ставки і забезпечити свою прибутковість. Великі спортивні події, такі як Australian Open, Wimbledon чи US Open також привертають увагу багатьох гравців, які бажають зробити власні прогнози та передбачити результати матчів.

Прогнозування тенісних матчів також має значний науковий і дослідницький потенціал. Розробка ефективних аналітичних моделей та алгоритмів може сприяти виявленню нових залежностей у грі, розумінню факторів, що впливають на результати матчів, і сприяти розвитку спортивної аналітики. Це важливо не лише для тенісу, але і для інших видів спорту, де прогнозування результатів має велике значення.

1.2 Огляд існуючих рішень проблеми

1.2.1 IBM SlamTracker

IBM SlamTracker [1] є аналітичним інструментом, розробленим компанією IBM для збору та візуалізації даних, який використовується в професійному тенісі для надання інформації про матчі у реальному часі. Інтерфейс інструменту представлено на рис. 1.1.

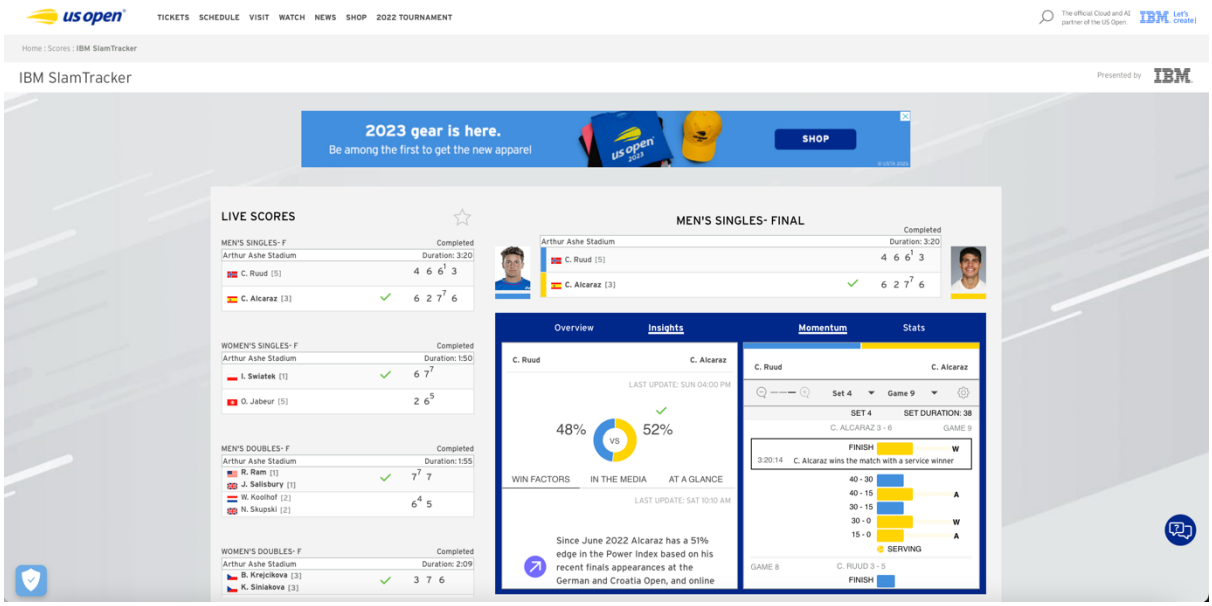


Рис. 1.1 – Сторінка на сайті турніру US Open інструменту IBM SlamTracker

IBM SlamTracker збирає великий обсяг даних під час матчів, таких як розташування кожного удару, швидкість м'яча, кут нахилу ракетки, виграшні та помилкові удари, тривалість розіграшу і багато іншого [2]. Ці дані аналізуються та використовуються для створення різних статистичних показників та візуалізації, які допомагають глядачам, тренерам та коментаторам краще розуміти хід матчу.

Завдяки IBM SlamTracker можна відстежувати розташування гравців на корті, визначати найчастіше використовувані удари, аналізувати ефективність подачі та прийому, прогнозувати результативність гравців у певних ситуаціях і багато іншого. Цей інструмент також може надати статистику з попередніх матчів гравців та порівняти їхні результати.

IBM SlamTracker використовується під час великих тенісних турнірів, таких як Australian Open, French Open, Wimbledon та US Open, для підвищення зацікавленості глядачів і покращення аналітичних можливостей у світі професійного тенісу.

Крім того, у IBM SlamTracker є розділ, де можна побачити прогноз на матч – тобто ймовірність перемоги у матчі для кожного гравця.

1.2.2 OddsTips

OddsTips [3] позиціонує себе як вебсайт, який пропонує користувачам безкоштовні щоденні прогнози на багато видів спорту, включаючи футбол, теніс, крікет, снукер та дартс. Головну сторінку вебсайту показано на рис. 1.2.

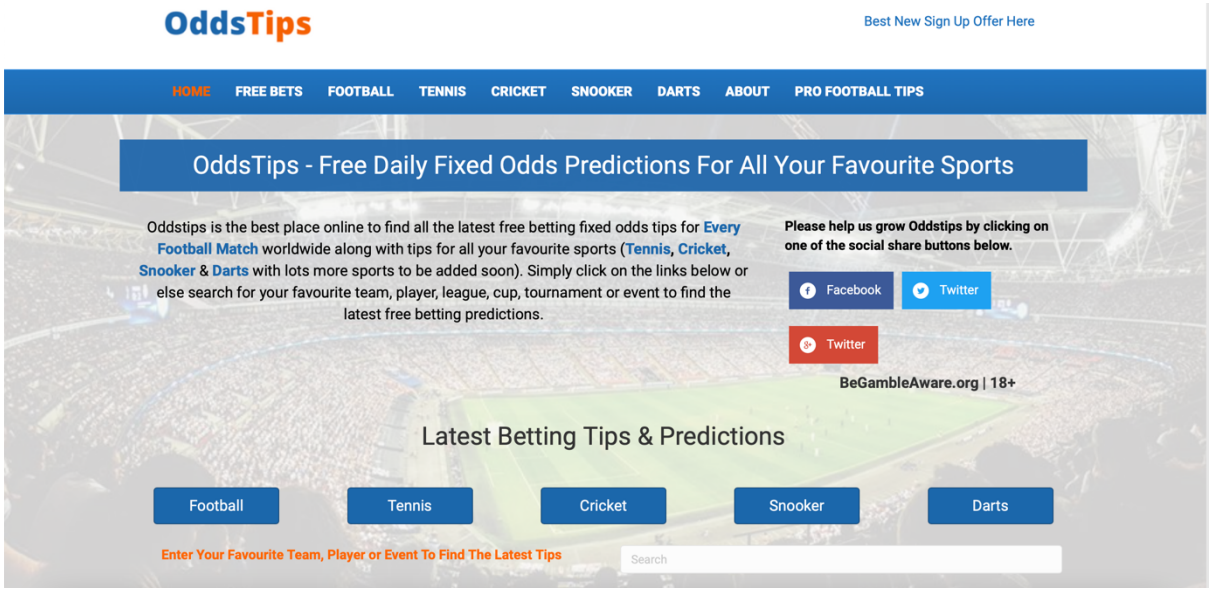


Рис. 1.2 – Головна сторінка вебсайту OddsTips

На сторінці сайту, де розміщуються тенісні прогнози написано, що OddsTips використовує унікальний 38-секційний алгоритм, щоб знайти найкращу ставку на будь-який тенісний матч. Там говориться, що на відміну від багатьох інших сайтів, які використовують нескладні алгоритми, засновані лише на 4 або 5 фрагментах даних (зазвичай попередні результати прямого порівняння), саме у них можна знайти чудові ставки, які базуються на набагато глибшому аналізі статистики та інформації.

Також зазначається, що алгоритм OddsTips враховує такі фактори, як прогнозована вологість, прогнозована температура на майданчику, скільки днів було надано гравцям відпочинку, час початку матчу, покриття, кількість матчів, зіграних кожним гравцем у поточному сезоні, тенденції форми і багато-багато іншого.

1.2.3 toptennistips.com

toptennistips.com [4] – це сайт, який за платну підписку надає рекомендації з тенісних ставок, підкріплених статистичним аналізом та штучним інтелектом, на більшість матчів. Домашню сторінку вебсайту продемонстровано на рис. 1.3.

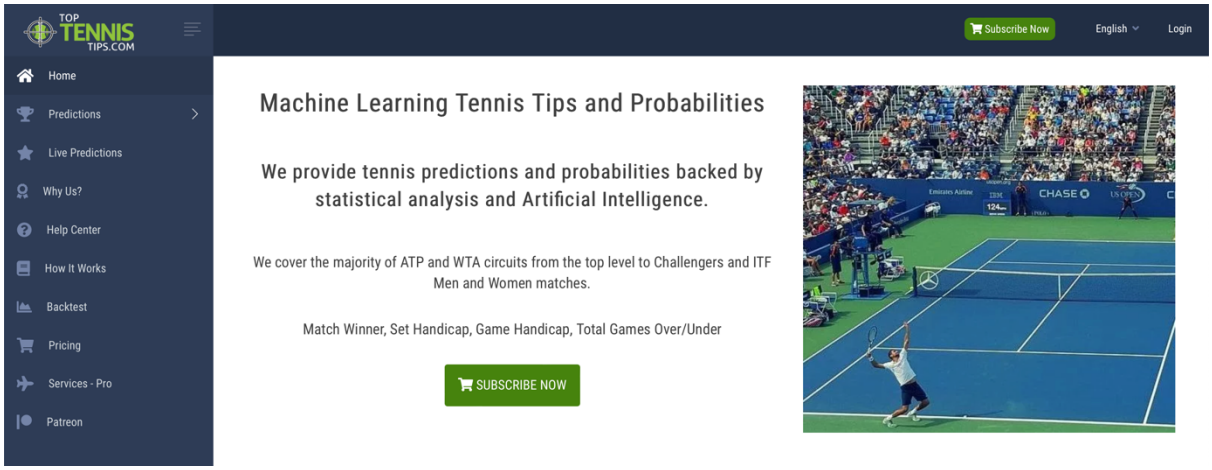


Рис 1.3 – Домашня сторінка вебсайту toptennistips.com

На сайті зазначається, що було ретельно проаналізовано 400 000 зіграних тенісних матчів, використовуючи всі види статистики, починаючи від покриття корту до особистих зустрічей. Потім були застосовані алгоритми машинного навчання та після багатьох коректувань отримано модель, що визначає шанси на перемогу одного гравця порівняно з іншим. Результатом роботи моделі є прогноз для кожного майбутнього матчу та ймовірність цього прогнозу.

1.3 Теніс. Загальні відомості. Параметри гри

Теніс – це спортивна гра, у якій приймають участь два або чотири гравці. Вона має проходити на спеціальному корті, розділеному навпіл за допомогою сітки. Для гри у теніс необхідно мати м’яч та ракетку. Головна мета гри – відправити м’яч на сторону суперника таким чином, щоб він не зміг його відбити. Гравці почергово виконують подачі та набирають очки, які потім формують гейми, що у свою чергу формують сети. Гра триває, доки один з гравців не виграє необхідну кількість сетів.

1.3.1 Керівні організації

Теніс поділяють на чоловічий та жіночий. Основним керівним органом чоловічого професійного тенісу є ATP (Association of Tennis Professionals) [5]. Ця асоціація організовує професійні міжнародні турніри серед чоловіків. Аналогом такої організації у жінок є WTA (Women’s Tennis Association) [6]. Існує також і третя організація, яка співпрацює із ATP та WTA, та є керівним органом світового тенісу. Ця організація має назву ITF (International Tennis Federation) [7]. Головними обов’язками ITF є підтримка та дотримання правил тенісу, регулювання міжнародних командних змагань, просування гри та збереження цілісності спорту за допомогою антидопінгових і антикорупційних програм. ITF також проводить ряд власних змагань.

1.3.2 Тенісні тури

У чоловічому тенісі існує три рівні турів [8]. Так туром найвищого рівня є Тур ATP (ATP Tour), який організовується Асоціацією тенісистів-професіоналів (ATP). Друге місце займає Тур ATP Челленджер (ATP Challenger Tour), який так само проводиться Асоціацією тенісистів-професіоналів. І найнижчий за рівнем існуючий тур це Чоловічі змагання ITF (ITF Men's Circuit), що проходять під егідою ITF.

У жіночому тенісі також існує три тури [9], аналогічні чоловічим: Тур WTA (WTA Tour), який проводиться Жіночою тенісною асоціацією (WTA), серія WTA 125K (WTA 125k series, також організовується WTA) та Жіночі змагання ITF (ITF Women's Circuit), що керуються ITF.

1.3.3 Категорії турнірів

У тенісі, як чоловічому так жіночому, існує поділ турнірів на категорії. Це зумовлено престижністю змагання: чим більш високу категорію воно має, тим більше призових та очків до рейтингу можна заробити. Нижче буде представлено таблиці 1.1 та 1.2 із переліком існуючих категорій турнірів, їх кількість у турі, а також організації, під чиєю егідою ці змагання проводяться, кількість очок, які можуть бути зароблені переможцем, та призовий фонд турніру.

Всі дані наведено станом на 2023 рік або найсвіжіші дані, якщо за поточний рік вони відсутні. Основну інформацію взято з [8-11].

Таблиця 1.1 – Інформація про чоловічі турніри

Категорія турніру	Кількість	Очки	Призовий фонд (у доларах)	Керівний орган
Grand Slam	4	2000	40 350 000 – 76 500 000	ITF
ATP Finals	1	1100 – 1500	14 750 000	ATP
ATP Tour Masters 1000	9	1000	5 779 335 – 8 800 000	ATP
ATP Tour 500	13	500	1 831 515 – 3 633 875	ATP
ATP Tour 250	38	250	562 815 – 1 377 025	ATP
ATP Challenger Tour	178	50 – 175	40 000 – 220 000	ATP
ITF Men’s Circuit	~550	15 – 20	15 000 – 25 000	ITF

Таблиця 1.2 – Інформація про жіночі турніри

Категорія турніру	Кількість	Очки	Призовий фонд (у доларах)	Керівний орган
Grand Slam	4	2000	40 350 000 – 76 500 000	ITF
WTA Finals	1	1500	5 000 000	WTA
WTA Premier Mandatory	4	1000	3 572 618 – 8 800 000	WTA
WTA Premier 5	5	900	2 527 250 – 2 788 468	WTA
WTA 500	12	470	703 590 – 899 500	WTA
WTA 250	~30	280	~259 000	WTA
WTA 125k series	24	160	125 000 – 162 480	WTA

Кінець таблиці 1.2

Категорія турніру	Кількість	Очки	Призовий фонд (у доларах)	Керівний орган
ITF Women’s Circuit	~500	10 – 140	15 000 – 100 000	ITF

1.3.4 Розмір турнірної сітки

У тенісі існує декілька варіантів розміру турнірної сітки змагання. Чим більш престижний турнір, тим більше гравців приймають у ньому участь. Так, наприклад, на кожному з турнірів Великого шолому (Grand Slams) у основній сітці змагань одиночного розряду беруть участь 128 тенісистів, у той час, як на турнірах серії ATP 250 розмір сітки може сягати 32.

1.3.5 Рейтинг та очки

У залежності від того, як далеко проходить тенісист по сітці у змаганнях, йому нараховується відповідна кількість очок, попередньо оговорена та затверджена керуючим органом змагань. У кожній асоціації (як ATP, так і WTA) існує одразу два рейтинги тенісистів: загальний рейтинг 52 тижнів (52-week rolling ranking) та поточний рейтинг за рік до дати (year to date ranking) [5].

Рейтинг за рік до дати скидається до нуля із початком нового сезону. До нього додаються очки, зароблені спортсменом у ході його виступів впродовж сезону. Таким чином формується рейтинг, за результатами якого наприкінці

ігрового сезону відбираються 8 найкращих тенісистів для участі у підсумковому турнірі асоціації (ATP Finals або WTA Finals).

Рейтинг 52 тижнів це основний рейтинг, який показує силу гравця. Очки у цьому рейтингу накопичуються впродовж року (тобто від дати закінчення окремого турніру до дати його початку або закінчення наступного року). Наприклад, якщо у 2022 році гравець набрав умовні 500 очок на турнірі Великого шолому Australian Open, то він зберігатиме ці очки до початку або завершення Australian Open 2023 року. Після цього попередні 500 очок за 2022 рік «згорять» до нуля, та гравцю буде нараховано очки, набрані на турнірі 2023 року. Рейтинг 52 тижнів не просто демонструє силу гравця, а також є важливим чинником у формуванні турнірних сіток змагань. Так гравці із високим рейтингом отримують кращі номери для посіву під час жеребкування основної сітки змагань. Це означає, що їм не потрібно проходити кваліфікацію для потрапляння в основні змагання, а також як правило, шлях для фіналу для вищих сіяних є більш простим (щонайменше «на листочку»).

1.3.6 Розряд

У тенісі існує два види розрядів: одиночний та парний. Одиночний розряд це гра 1 на 1, а парний 2 на 2. Окрім парного розряду серед чоловіків та жінок існує ще й так званий мікст – парний розряд, де пару формують чоловік та жінка.

1.3.7 Структура матчу

Гра у теніс складається із сетів, які у свою чергу складаються із геймів, які у свою чергу складаються з очок.

Гейм – це період часу, протягом якого подає тільки один гравець [12]. Кожен гейм починається з рахунку 0-0. Далі за кожний виграний м’яч нараховуються очки за схемою «15-30-40». При рахунках «40-0», «40-15», «40-30» наступний виграний розіграш першим гравцем означатиме перемогу у геймі. При рахунку «40-40» (deuce) розігрується так звана перевага (advantage). Справа у тому, що у геймі перевага одного з гравців має бути мінімум у два м’ячі. Отже, якщо рахунок на табло показує «Adv-40» і наступний розіграш виграє перший гравець, то він виграє і гейм. Якщо виграє другий гравець, то рахунок повертається до вигляду «40-40».

Боротьба у сеті триває, допоки один з гравців не виграє 6 геймів. Але аналогічно, як і з очками у геймі, перевага одного з гравців у сеті має становити мінімум два гейми. Це означає, що при рахунку «6-5» на користь одного з гравців розігрується ще один гейм. Якщо рахунок стає «7-5», то сет завершується перемогою гравця, у якого на рахунку 7 геймів. Якщо рахунок набуває вигляду «6-6», то розігрується так званий тайбрейк (tiebreak).

На тайбрейку гравці почергово виконують подачі та набирають очки. Тайбрейк триває, як правило, до 7 очок (деякі турніри видозмінюють регламент та грають до 10). Незмінною умовою виграшу тайбрейку є різниця у два очки між гравцями. Він буде продовжуватись, доки не буде набрано мінімальної кількості очок та одночасно не досягнуто мінімальної різниці. Рахунок після завершення тайбрейку матиме вигляд «7-6» по геймах на користь одного з гравців.

Варто зазначити, що не на всіх турнірах передбачено тайбрейк при рахунку «6-6». Так, донедавна на турнірі Великого шолому US Open у

фінальному сеті тайбрейк не грали, а грали доти, доки один з гравців не отримає перевагу у два гейми.

Зазвичай тенісний матч відбувається допоки один з гравців не виграє два сети. Тобто максимально може бути зіграно три сети у матчі («2-0», «2-1»). Однак, на турнірах Великого шолому у матчах одиночного та парного розрядів серед чоловіків гра триває до 3 сетів, тобто максимально може бути зіграно 5 сетів.

1.3.8 Покриття

У тенісі існує 4 види покриття кортів: хард, ґрунт, трава та килим. Кожен тип покриття має свої особливості, а тому гравці показують кращі або гірші результати в залежності від того, на якому корті вони виступають.

1.4 Конкретизація задачі

1.4.1 Чоловічий теніс

У даній роботі розглядатиметься лише чоловічий теніс, оскільки чоловіки є більш надійними для прогнозування, ніж жінки.

Одна з основних причин полягає в тому, що чоловіки зазвичай краще тримають власну подачу. Вони мають сильнішу фізичну конституцію та здатність до більш швидкого та потужного удару, що робить їх подачі складнішими для суперників. Це призводить до меншої кількості пропущених геймів на своїй подачі, що робить їх результати більш передбачуваними.

У порівнянні з цим, в жіночому тенісі подача не завжди є настільки сильною, що створює можливості для прогнозування. Жінки, хоч і технічно висококваліфіковані, зазвичай мають меншу фізичну потужність, і це може призводити до більшої кількості брейків та втрати геймів на своїй подачі. Це створює більшу невизначеність в результатах матчів і робить жіночий теніс менш надійним для прогнозування.

Для підтвердження вищесказаного скористаємось даними, взятими у [13], і наведемо статистику за останні 10 років, яка показує, скільки відсотків геймів вигравали чоловіки та жінки на власних подачах (рис. 1.4):

Рис 1.4 – Відсоток геймів виграних на своїй подачі представниками кожної статі

Як бачимо, чоловіки краще тримають свою подачу на цілих 12%.

1.4.2 Тури ATP та ATP Challenger

При виборі турів для прогнозування матчів було прийнято рішення обмежитися двома турами – ATP та ATP Challenger, оскільки ITF Men’s Circuit є туром низького рівня.

ITF Men’s Circuit є туром, де беруть участь гравці низького рівня, включаючи новачків, які ще тільки починають свій професійний шлях у тенісі. Цей тур є важливим перехідним етапом для гравців, де вони набувають необхідного досвіду та підвищують свої навички перед переходом до вищого рівня змагань. Однак, через низький рівень гри та меншу конкуренцію, прогнозування результатів матчів в ITF Men’s Circuit може бути менш надійним.

Крім того, в ITF Men’s Circuit зазвичай відсутні статистичні дані про матчі. У порівнянні з турами ATP та ATP Challenger, де звіти, статистика та інформація про гравці є у вільному доступі, ITF Men’s Circuit не отримує такого розгалуженого покриття. Відсутність вичерпних статистичних даних ускладнює аналіз та прогнозування матчів.

1.4.3 Матчі з трьох сетів

Щоб збільшити точність та надійність прогнозів, було прийнято рішення спрямувати увагу на матчі з трьох сетів. Це дозволить сконцентруватися на стандартному форматі гри та розробити модель прогнозування, яка краще враховує особливості трисетових поєдинків. П’ятисетові поєдинки мають свою специфіку гри і не можуть прогнозуватися одночасно з трисетовими матчами.

1.4.4 Одиночний розряд

Було прийнято рішення зосередитись на прогнозуванні матчів одиночного розряду.

Справа в тому, що у парному розряді важливо враховувати форму кожного гравця, його навички, фізичну підготовку та стиль гри. Кожен гравець може мати свої переваги та слабкі сторони, які впливають на командну гру. Окрім того, успішність пари залежить від їхньої взаємодії, координації та спільного розуміння на корті.

Прогнозування матчів в парному розряді також ускладнюється відсутністю статистичних даних для кожного гравця окремо. Зазвичай, доступні дані про матчі в парному розряді обмежені, а статистика для кожного гравця окремо не є легкодоступною. Це ускладнює аналіз форми гравців та прогнозування їхньої продуктивності у парному розряді.

1.5 Головна проблема моделювання

У даній роботі для моделювання результатів тенісних матчів будуть використовуватися прогностичні моделі, які для отримання найбільш точних прогнозів використовують багато різних факторів, у тому числі і статистику гравців у матчі. Проте, головною проблемою, яка виникає на даному етапі, є відсутність можливості передбачити статистику конкретного матчу заздалегідь. Для ефективного прогнозу результатів майбутніх матчів необхідно здійснити передбачення статистики матчу, який ще не відбувся.

На щастя, існує рішення цієї проблеми. Одним з таких підходів є накладання вагів на попередні матчі гравців в залежності від часового інтервалу між ними. Цей метод описаний у роботі Міхаля Сіпко [14]. Ідея полягає в тому, що більша вага надається недавнім матчам, оскільки вони відображають більш актуальну форму гравців. Чим ближче в часі попередній матч до майбутнього, тим більшу роль він відіграє в прогнозуванні. Наприклад, матчі, які 35-річний гравець зіграв у минулому році, ймовірно, дадуть кращі оцінки його продуктивності, ніж матчі, зіграні у віці 20 років.

Описаний метод називається часовим дисконтуванням (time discounting). Ваги будуть накладатися за допомогою експоненціальної функції:

$$W(t) = \min(f^t,\ f)\ ,$$

де t – це час у місяцях, що пройшли від часу запланованого матчу, а f – дисконтний коефіцієнт. Він може набувати значень від 0 до 1, і визначає величину ефекту дисконтування часу. Якщо обрати f досить малим, то це означатиме, що більш старі матчі матимуть меншу вагу.

Наведемо приклад функції ваг для $f = 0.7$ (рис 1.5):

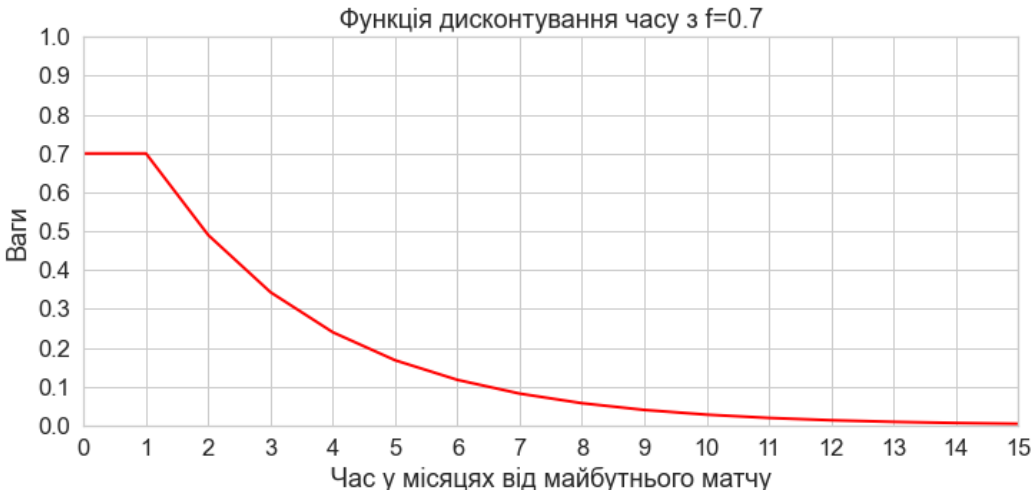


Рис 1.5 – Зображення функції дисконтування часу для $f = 0.7$

1.6 Висновки до розділу 1

У цьому розділі було з’ясовано актуальність прогнозування тенісних матчів, розглянуто існуючі рішення цієї проблеми, наведено загальні відомості про теніс та ідентифіковано основну проблему, яка стає на шляху моделювання результатів.

Прогнози на спортивні події завжди цікавили як професіоналів, так і любителів. Відтак, моделювання результатів тенісних матчів стає актуальною

та перспективною галуззю. Дана робота пишеться з метою зробити вагомий внесок у цю сферу, використовуючи потужні алгоритми машинного навчання.

Огляд існуючих рішень дав нам можливість побачити, що вже існують вебсайти, що надають прогнози на тенісні події. Однак, зрозуміло, що велика частина цих рішень може бути обмежена у своїх методах та точності прогнозування. Тому метою цієї роботи є створення предиктивних моделей та покращення точності прогнозування результатів.

Опис тенісу та загальних відомостей про нього був важливим етапом даного дослідження, оскільки це дозволило підготуватись до розуміння контексту та особливостей, які впливають на результати тенісних матчів. Було з’ясовано безпосередню структуру матчу, класифіковано існуючі тенісні тури та турніри, розглянуто багато інших факторів, які можуть впливати на гру.

У даному досліджені основну увагу буде сконцентровано на прогнозуванні результатів чоловічого одиночного розряду двох найпрестижніших тенісних турів: ATP та ATP Challenger. Також до уваги не братимуться турніри, на яких за регламентом грається 5 сетів. Це 4 турніри серії Grand Slam.

Як вже було з’ясовано раніше, головною проблемою, яка виникає при моделюванні результатів тенісних матчів, є відсутність статистики для майбутніх матчів, тобто матчів, які ще не відбулися. Для вирішення цією проблеми було запропоновано рішення, яке полягає в накладанні вагів на попередні матчі та використанні їх статистики для прогнозування результатів. Використовуючи вагові коефіцієнти, можна присвоїти більшу вагу останнім матчам. Цей підхід дозволяє враховувати тенденції та форму гравців перед початком гри.

Варто зазначити, що побудовані у роботі моделі також будуть враховувати і інші фактори, що можуть впливати на результат гри. Наприклад, покриття корту, фізичні характеристики гравців, рівень турніру, рейтинг тощо. Це дозволить створити більш комплексну та точну модель прогнозування.

2 ОПИС МЕТОДІВ МАШИННОГО НАВЧАННЯ ДЛЯ МОДЕЛЮВАННЯ РЕЗУЛЬТАТІВ ТЕНІСНИХ МАТЧІВ

2.1 Переваги машинного навчання перед статистичними моделями

Машинне навчання в останні роки стало невід'ємною складовою багатьох сфер нашого життя. Ця галузь інформатики використовує комп'ютерні алгоритми для навчання системи робити прогнози, засновані на великій кількості даних. У контексті моделювання результатів тенісних матчів машинне навчання має кілька переваг порівняно з традиційними статистичними моделями.

По-перше, машинне навчання здатне виявляти складні залежності в даних. У тенісі є багато факторів, які можуть впливати на результат матчу, такі як форма гравців, рейтинг, тип корту, антропометрія гравців тощо. Машинне навчання може аналізувати великі обсяги даних і виявляти складні взаємозв'язки між цими факторами, що допомагає покращити точність прогнозів.

По-друге, машинне навчання здатне адаптуватися до умов, що змінюються. У тенісному світі стилі гри гравців можуть змінюватися з часом. Традиційні статистичні моделі можуть бути обмеженими в адаптації до таких змін. За допомогою машинного навчання можна будувати моделі, які можуть вчитися на нових даних і враховувати зміни в тенісній індустрії.

По-третє, машинне навчання дозволяє автоматизувати процес аналізу даних. Замість ручного визначення статистичних правил і залежностей, машинні алгоритми можуть самостійно виявляти патерни та структури в даних. Це дозволяє значно збільшити ефективність та швидкість прогнозування, зокрема і результатів тенісних матчів.

Ще однією перевагою є те, що машинне навчання може бути гнучким і готовим до масштабування. Завдяки широкому спектру алгоритмів і підходів, можна використовувати різні методи машинного навчання, такі як нейронні мережі, дерева рішень, методи опорних векторів та інші, щоб досягти оптимального результату. Крім того, збільшення обсягу даних також не є проблемою, оскільки моделі можуть ефективно працювати з великими наборами даних.

Важливо також враховувати широкий доступ до інформації і ресурсів, який є сьогодні. Велика кількість тенісних статистичних даних доступна онлайн, включаючи історичні результати матчів, статистику гравців, рейтинги та інші параметри. Використання машинного навчання дозволяє ефективно використовувати ці дані для прогнозування результатів тенісних змагань.

Отже, обрання машинного навчання для моделювання результатів тенісних матчів є обґрунтованим рішенням. Його переваги, такі як здатність до виявлення складних залежностей, адаптація до змінюючихся умов, автоматизація процесу аналізу даних та гнучкість, роблять машинне навчання ефективним інструментом для прогнозування результатів тенісних змагань.

2.2 Логістична регресія

2.2.1 Визначення

Логістична регресія [15] є алгоритмом машинного навчання, який використовується для моделювання ймовірності належності об’єкта до певного класу. Вона є одним з ключових методів для бінарної та багатокласової класифікації і знаходить широке застосування у багатьох галузях, включаючи медицину, фінанси, маркетинг та багато інших.

Основна ідея логістичної регресії полягає у моделюванні залежності між залежною змінною (вихідними даними) і набором незалежних змінних (вхідними даними) за допомогою логістичної функції. Зазвичай, залежна змінна представляє собою бінарний результат, такий як “так” або “ні”, “1” або “0”. Однак, логістична регресія також може бути застосована до багатокласових задач класифікації, використовуючи розширені моделі.

У логістичній регресії використовується логістична функція, яка перетворює лінійну комбінацію вхідних змінних в ймовірність виникнення певного результату (тобто вихідне значення знаходиться в інтервалі [0, 1]). Логістична функція, також відома як сигмоїдна функція, має форму:

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

де $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$ є лінійною комбінацією незалежних змінних $x_i, i = \overline{1,n}$ та їх коефіцієнтів $\beta_i, i = \overline{1,n}$, а $n -$ кількість вхідних змінних (предикторів). Сигмоїдну функцію зображено на рис. 2.1.

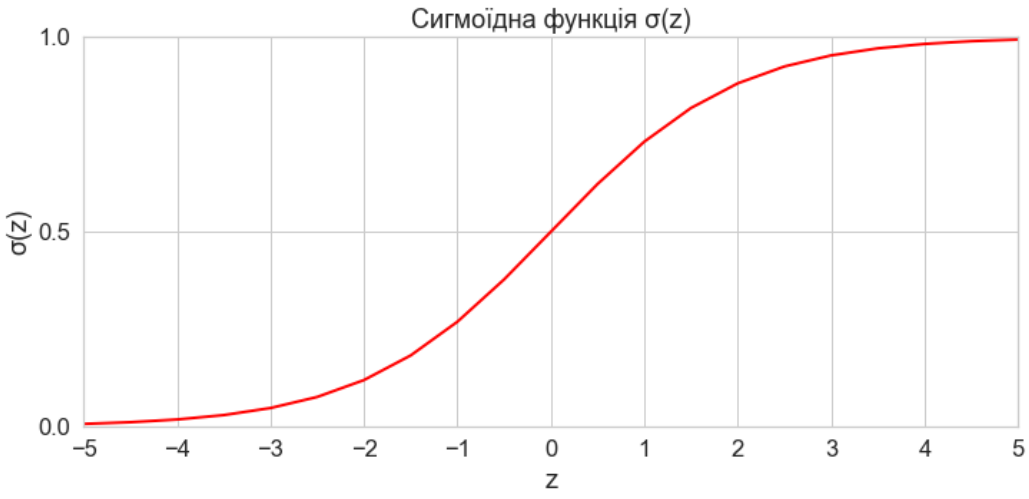


Рис. 2.1 – Зображення сигмоїдної функції на відрізку [-5;5]

2.2.2 Функція логістичних втрат та методи її мінімізації

Навчання моделі логістичної регресії полягає в оптимізації параметрів β, що відповідають за ваги змінних моделі. Цей процес має на меті знайти оптимальні значення ваг, які мінімізують функцію логістичних втрат [16] і забезпечують кращу прогнозну здатність моделі. Функція логістичних втрат, у свою чергу, має наступний вигляд:

$$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i)) \quad ,$$

де $N -$ кількість вхідних прикладів (samples), $y_i -$ реальне значення i-того прикладу, а $p_i -$ прогнозована ймовірність для i-того прикладу.

Існує декілька методів мінімізації функції логістичних втрат у логістичній регресії [17-18], деякі з них приведені нижче:

1. Градієнтний спуск – це ітеративний алгоритм, що шукає локальний мінімум функції шляхом крокування в напрямку найшвидшого зменшення функції. У логістичній регресії, градієнтний спуск використовується для оновлення параметрів моделі на кожному кроці.
2. Стохастичний градієнтний спуск – це метод градієнтного спуску, який обчислює градієнт на кожному зразку тренувальної вибірки і оновлює параметри моделі відповідно. Він ефективний для великих даних, оскільки використовує лише один зразок в кожному кроці, тому що він знижує обчислювальну складність.
3. Метод Ньютона. Цей метод використовує другу похідну функції втрат, відому як матриця Гессе, для знаходження оптимальних значень параметрів. Він здатний швидше збігатися до мінімуму, оскільки враховує інформацію про кривизну функції.

4. L-BFGS – це ітеративний алгоритм, який поєднує в собі ідеї методів градієнтного спуску і методу Ньютона. Він шукає оптимальне значення функції шляхом обчислення градієнту на кожному кроці, але використовує збереження історії кроків для оновлення параметрів.

2.2.3 Перенавчання моделі та способи запобігання

Перенавчання моделі (overfitting) є явищем у машинному навчанні, коли модель надмірно адаптується до тренувальних даних і стає неефективною в узагальненні до нових, невідомих даних [19]. Таке явище виникає, коли модель занадто складна або має забагато параметрів, і вона “запам’ятовує” шум чи випадкові закономірності у тренувальних даних.

У випадку логістичної регресії, перенавчання може стати проблемою, якщо модель стає дуже чутливою до тренувальних даних і недостатньо узагальнюється до нових прикладів. Це може призводити до невірних прогнозів на нових даних.

Для запобігання перенавчанню моделі логістичної регресії, можна використовувати наступні способи [19]:

1. Регуляризація. Додавання штрафної функції до функції втрат, яка штрафує великі значення параметрів моделі. Це допомагає контролювати розмір параметрів і уникнути перенавчання. Дві типові форми регуляризації – L1 (LASSO) та L2 (регуляризація Ridge). L1 регуляризація використовує штрафну функцію, яка базується на сумі абсолютних значень параметрів моделі. Цей метод має властивість робити деякі параметри нульовими, що дозволяє використовувати його для відбору ознак. Тобто, L1 регуляризація використовується для виключення менш важливих змінних з моделі, тим самим зменшуючи складність моделі. L2 регуляризація використовує штрафну функцію,

яка базується на квадратичних значеннях параметрів моделі. Цей метод стимулює модель до зменшення всіх параметрів, але без тенденції робити їх нульовими. L2 регуляризація сприяє зменшенню значень параметрів моделі, але не виключає їх повністю.

2. Валідація. Валідація – це процес оцінки продуктивності моделі на незалежній вибірці даних, що не використовувалася під час тренування моделі. Валідація допомагає оцінити, наскільки добре модель узагальнює свої прогнози на нові, невідомі дані. Такий підхід дозволяє виявити, коли модель починає погіршувати прогнози на нових даних, що може бути ознакою перенавчання.
3. Крос-валідація. Крос-валідація є методом оцінки продуктивності моделі у машинному навчанні, який дозволяє отримати більш надійну оцінку її узагальнювальної здатності. Замість традиційного розділення набору даних на тренувальну та валідаційну вибірки, крос-валідація використовує повторні проби, щоб оцінити модель на різних підвибірках даних.
4. Обмеження складності моделі. Можна зменшити складність моделі шляхом обмеження числа параметрів або використання простішої функціональної форми.

2.2.4 Переваги та недоліки використання логістичної регресії

Як і будь-який інший алгоритм машинного навчання, логістична регресія має як свої переваги, так і недоліки [20]. Головними перевагами даного алгоритму є:

- інтерпретованість. Результати логістичної регресії легко інтерпретувати. Можна визначити вагу кожного параметра та зрозуміти, як він впливає на ймовірність належності до певного класу;

- ефективність. Логістична регресія зазвичай працює швидше, ніж складніші алгоритми, такі як нейронні мережі. Вона добре масштабується до великих наборів даних і може бути ефективно використана навіть при невеликій кількості спостережень;
- робота з категоріальними змінними. Логістична регресія легко обробляє категоріальні змінні та враховує їх вплив на прогнози. Це робить її гнучким інструментом для аналізу даних з різноманітними типами змінних.

Недоліками алгоритму є:

- лінійність. Логістична регресія моделює лише лінійні залежності між змінними, тому не здатна захопити складні нелінійні залежності в даних;
- вразливість до викидів. Логістична регресія може бути чутлива до викидів або аномалій у даних, що може вплинути на точність моделі;
- залежність від попередньої обробки даних. Деякі типи даних можуть потребувати попередньої обробки, наприклад, кодування категоріальних змінних або масштабування числових змінних.

Для прогнозування результатів тенісних матчів метод логістичної регресії є ефективним методом з ряду причин. Перш за все, задача прогнозування результатів тенісних матчів є задачею бінарної класифікації, оскільки результати можуть бути тільки “виграш” або “поразка” для кожного гравця. Логістична регресія добре підходить для таких задач, оскільки може моделювати ймовірність належності до кожного класу.

Крім того, вхідні змінні у задачі прогнозування результатів тенісних матчів можуть бути різного типу. Наприклад, можуть використовуватися числові змінні, такі як рейтинг гравців, вік, зріст, а також категоріальні змінні, такі як тип покриття корта чи рівень турніру. Логістична регресія має гнучкий підхід до роботи з різними типами змінних, що дозволяє включати їх у модель і враховувати їх вплив на прогнозування.

2.3 Feed-forward мережа

2.3.1 Визначення

Нейронна мережа [21] є потужним методом машинного навчання. Вона імітує роботу нервової системи людини: складається з взаємопов’язаних штучних нейронів, які обмінюються інформацією у формі сигналів. Нейронні мережі використовуються для розв’язання складних завдань обробки і аналізу даних, включаючи класифікацію, розпізнавання образів, прогнозування та генерацію контенту.

Feed-forward мережа (також відома як пряма мережа) є одним з основних типів штучних нейронних мереж [22]. Це простий та широко використовуваний тип нейронної мережі, який виконує пряму передачу сигналів від вхідних до вихідних шарів без циклічних зв’язків.

Архітектура feed-forward мережі (рис. 2.2) складається з трьох основних типів шарів: вхідного шару, прихованих шарів та вихідного шару. Кожен шар складається зі значень нейронів, які обчислюються за допомогою активаційної функції (таких функцій існує багато, але найпоширенішими є Softplus, ReLu та Sigmoid). Зв’язки між шарами мають напрямок від вхідного шару до вихідного шару, звідси і назва “пряма” мережа.

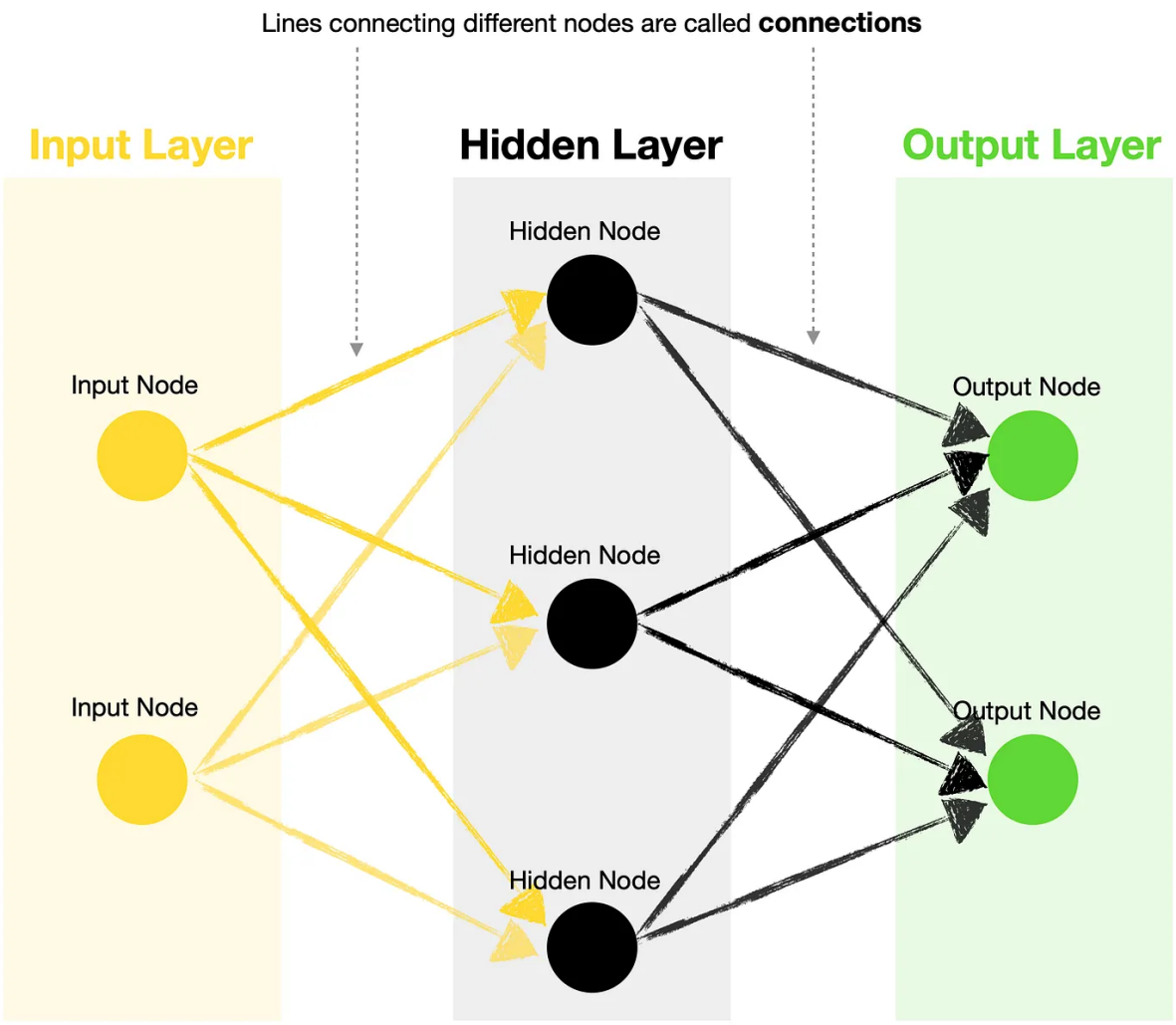


Рис 2.2 – Архітектура feed-forward мережі

Feed-forward мережі широко застосовуються у багатьох областях, таких як класифікація зображень, розпізнавання мови, прогнозування, аналіз даних тощо. Вони можуть мати різнну архітектуру, включаючи один або кілька прихованих шарів з різною кількістю нейронів у кожному шарі. Більш глибокі мережі з багатьма прихованими шарами можуть виявити складніші залежності та робити більш складні обчислення.

2.3.2 Принцип роботи feed-forward мережі

Принцип роботи feed-forward мережі полягає в наступному [23]:

1) вхідні дані передаються вхідному шару, де кожен нейрон отримує свої значення;
2) значення нейронів вхідного шару передаються до прихованих шарів через зв’язки, ваги яких регулюються під час навчання;
3) кожен нейрон у прихованому шарі обчислює ваговану суму своїх вхідних значень та застосовує активаційну функцію для вироблення вихідного значення;
4) процес передачі відбувається через всі приховані шари, поки не буде досягнуто вихідного шару;
5) нейрони вихідного шару обчислюють остаточний результат, використовуючи активаційну функцію для вироблення вихідного сигналу.

2.3.3 Процес навчання feed-forward мережі

Процес навчання feed-forward мережі [22-23] включає кілька етапів, а саме:

1. Ініціалізація ваг. Ваги мережі ініціалізуються випадковими значеннями або за допомогою певного алгоритму ініціалізації, наприклад, Гауссового розподілу.
2. Прямий прохід (forward pass). Тренувальні дані подаються на вхід мережі. Кожен шар обчислює активації своїх нейронів на основі ваг та вхідних сигналів. Інформація проходить через мережу від вхідного шару до вихідного шару без зворотного зв’язку.

3. Обчислення помилки. Порівнюються вихідні значення, отримані з мережі, зі справжніми значеннями цільової величини. Обчислюється функція втрати, яка вимірює величину помилки між прогнозованими та справжніми значеннями. Найпоширенішими видами функцій втрат є середньоквадратична помилка (MSE), бінарна перехресна ентропія (Binary Cross-entropy) та категоріальна перехресна ентропія (Categorical Cross-entropy).
4. Зворотнє поширення помилки (backpropagation). Похибка, обчислена на попередньому кроці, поширюється від вихідного шару до вхідного шару за допомогою алгоритму зворотнього поширення помилки. Цей алгоритм використовує ланцюжкове правило для обчислення градієнтів по вагах мережі.
5. Оновлення ваг. Градієнти, обчислені на попередньому кроці, використовуються для оновлення ваг мережі. Це виконується за допомогою алгоритмів оптимізації, найпоширенішими з яких є стохастичний градієнтний спуск (SGD), ADAM та RMSProp. Оновлення ваг спрямовані на зменшення значення функції втрати та покращення продуктивності мережі.
6. Повторення. Процес прямого проходу, обчислення помилки, зворотного поширення помилки і оновлення ваг повторюється для кожного тренувального зразка у тренувальному наборі даних. Цей процес називається ітераціями або епохами навчання. Чим більше ітерацій виконується, тим більше можливостей мережа має, щоб навчитися відповідати на тренувальні дані і робити більш точні прогнози.

Процес навчання продовжується до тих пір, поки не буде виконаний один з критеріїв зупинки, який може бути визначений заздалегідь. Це може бути фіксована кількість ітерацій, досягнення певного рівня точності моделі або зменшення функції втрати до певного порогового значення.

Після завершення процесу навчання мережу можна використовувати для прогнозування нових невідомих даних. Це називається процесом інференсу.

Мережа застосовує прямий прохід до нових вхідних даних і генерує прогнози на основі вже навчених ваг.

2.3.4 Перенавчання нейронних мереж та методи запобігання

Для нейронних мереж, зокрема і для feed forward мереж, так само, як і для інших методів машинного навчання, властиве явище перенавчання (overfitting), що вже було описане у розділі 2.2.3.

Для запобіганню перенавчанню нейронних мереж використовуються різні методи [24], зокрема регуляризацію, дропаут та раннє відключення:

1. Регуляризація. Цей метод вже було описано у розділі 2.2.3.
2. Дропаут (Dropout). Це метод, в якому під час навчання випадковим чином відключаються (обнуляються) деякі нейрони з певною ймовірністю. Це призводить до того, що модель навчається без певних нейронів на кожній ітерації, що допомагає запобігти занадто сильному зв’язуванню нейронів та перенавчанню. Під час тестування всі нейрони активні, але їх ваги масштабуються для компенсації відключення під час навчання.
3. Раннє відключення (Early Stopping). Це метод, в якому навчання зупиняється раніше за умови, що функція втрат на валідаційному наборі даних перестала покращуватися. Зазвичай, під час навчання моніторяться метрики на тренувальному та валідаційному наборах. Якщо значення функції втрат на валідаційному наборі перестає зменшуватися або починає зростати, це може свідчити про перенавчання. В такому випадку навчання припиняється, а найкраща модель зберігається.

2.4 Багатошаровий персептрон

2.4.1 Визначення

Багатошаровий персептрон (Multilayer Perceptron – MLP) [25] є одним з найпоширеніших різновидів feed-forward мереж. Він представляє собою архітектуру з одним або декількома прихованими шарами нейронів. Кожен нейрон у шарі пов’язаний з нейронами попереднього та наступного шарів. MLP використовує нелінійні функції активації для створення нелінійного зв’язку між нейронами. Цей нелінійний зв’язок дозволяє MLP моделювати складні нелінійні відносини в даних.

На рис. 2.3 показана архітектура багатошаровго персептрону для бінарної класифікації з трьома вхідними змінними:

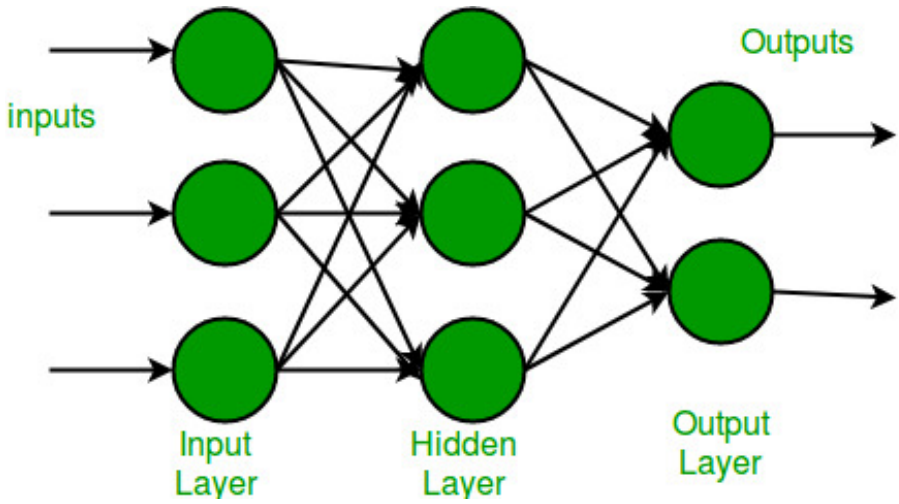


Рис. 2.3 – Архітектура MLP для бінарної класифікації з трьома вхідними змінними

2.4.2 Переваги та недоліки MLP

MLP має як переваги, так і деякі недоліки [26]. Переваги включають наступне:

- здатність моделювати складні нелінійні залежності. MLP може моделювати складні залежності між вхідними та вихідними даними завдяки своїй гнучкій архітектурі з прихованими шарами та нелінійними функціями активації;
- тренування з використанням зворотного поширення помилки. MLP використовує алгоритм зворотного поширення помилки, який є ефективним методом навчання нейронних мереж. Він дозволяє оновлювати ваги мережі, мінімізуючи функцію втрати;
- гнучкість в обробці різноманітних типів даних. MLP може працювати з різноманітними типами даних, включаючи числові, категоріальні та бінарні дані.

Недоліки MLP:

- перенавчання. MLP може бути схильним до перенавчання, коли модель надмірно вивчає навчальні дані, але показує погану загальну здатність до прогнозування на нових даних. Це може стати проблемою, якщо немає ефективних методів регуляризації та контролю перенавчання;
- вимоги до обчислювальних ресурсів. MLP зазвичай має велику кількість параметрів, особливо у великих та глибоких мережах, що може вимагати значних обчислювальних ресурсів для навчання та використання. Це може стати проблемою при обробці великих обсягів даних або на пристроях з обмеженою потужністю;
- вибір архітектури. Визначення оптимальної архітектури MLP (кількість шарів, розмір шарів, функції активації тощо) може бути складним завданням. Неправильний вибір архітектури може призвести до недо- або перенавчання, що вплине на точність прогнозування.

Варто зазначити, що багатошаровий персептрон має свої переваги в прогнозуванні результатів тенісних матчів. По-перше, MLP здатен моделювати складні нелінійні залежності між вхідними факторами (наприклад, рейтинг гравців, вік, зріст тип кортів тощо) та результатами матчу. По-друге, MLP може використовуватися для бінарної класифікації, а за допомогою відповідної функції активації в останньому шарі може повертати ймовірності виграшу або поразки гравця, що дозволяє отримати інформацію про ймовірнісний розподіл результатів матчу.

2.5 Метрики для оцінки моделей

Оцінка моделей є важливим етапом в машинному навчанні, вона дозволяє оцінити продуктивність моделей, порівняти їх ефективність та зробити вибір найкращої моделі для конкретної задачі. Для цього використовуються різноманітні метрики, деякі з них є:

- Accuracy (Точність). Accuracy визначає відсоток правильних передбачень моделі відносно загальної кількості прикладів. Метрика вимірює загальну точність моделі;
- Precision (Точність). Precision вимірює пропорцію правильно передбачених позитивних екземплярів відносно всіх екземплярів, які модель визначила як позитивні. Метрика вказує на те, наскільки точні передбачення моделі щодо позитивного класу;
- Recall (Повнота). Recall вимірює пропорцію правильно передбачених позитивних екземплярів відносно всіх фактичних позитивних екземплярів. Метрика вказує на те, наскільки добре модель знаходить всі фактичні позитивні приклади;
- F1 score. F1 score це гармонічне середнє між Precision і Recall. F1-показник зводиться до одного числа, яке враховує як точність, так і

повноту. Він використовується для оцінки збалансованості між точністю і повнотою моделі;

- Roc Auc. Roc Auc вимірює якість бінарного класифікатора шляхом оцінки його здатності розрізняти між позитивним і негативним класами. Вона обчислює площу під кривою ROC (Receiver Operating Characteristic);
- Loss. Loss вимірює ефективність моделі, враховуючи ймовірність передбачення. Метрика використовується для вимірювання відповідності між прогнозованими ймовірностями і фактичними значеннями цільової змінної. Менші значення втрати вказують на кращу продуктивність моделі.

Варто зазначити, що всі описані вище метрики можуть бути застосовані як для оцінки моделі логістиичної регресії, так і для оцінки мультишарового персептрону.

2.6 Висновки до розділу 2

У цьому розділі було з’ясовано, що машинне навчання є більш передовим і ефективним підходом у моделюванні результатів тенісних матчів порівняно зі статистичними моделями.

Також було розглянуто два методи машинного навчання: логістичну регресію та різновид feed forward мережі багатошаровий персептрон (MLP). Логістична регресія є простим і ефективним алгоритмом для бінарної класифікації результатів тенісних матчів. Вона дозволяє прогнозувати ймовірність перемоги гравців. Багатошаровий персептрон є більш потужним алгоритмом, здатними виявляти складні нелінійні залежності. Він має гнучку архітектуру, яка дозволяє працювати з різними типами даних і обробляти

складніші задачі прогнозування результатів тенісних матчів. Було наведено переваги та недоліки обох методів.

Надалі ці методи будуть застосовані для моделювання результатів тенісних матчів.

3 ПОБУДОВА ПРОГНОСТИЧНИХ МОДЕЛЕЙ НА ОСНОВІ ОБРАНИХ МЕТОДІВ МАШИННОГО НАВЧАННЯ

3.1 Вибір мови програмування та середовища розробки

Для реалізації поставленої задачі було вирішено обрати мову програмування (МП) Python. Ця МП є однією з найбільш популярних мов програмування для розробки алгоритмів машинного навчання та побудови прогностичних моделей. Доцільність вибору Python для реалізації алгоритмів машинного навчання можна обґрунтувати наступними перевагами цієї МП [27]:

1. Простота використання. Python має простий і зрозумілий синтаксис, що робить його доступним як для початківців так і для досвідчених розробників. Код, написаний мовою Python, можна легко читати та розуміти, що робить його ідеальним вибором для реалізації складних алгоритмів машинного навчання.
2. Багата екосистема. Python має широкий набір бібліотек та інструментів для машинного навчання, таких як NumPy, Pandas, Scikit-learn, TensorFlow, PyTorch та інші. Ці бібліотеки пропонують потужні функції для обробки даних, побудови моделей та оцінки їх результатів. Велика активна спільнота розробників забезпечує постійний розвиток та підтримку цих інструментів.
3. Доступність навчальних матеріалів та документаційних ресурсів. Python є популярним вибором серед багатьох спеціалістів з машинного навчання, а також існує велика кількість онлайн ресурсів, де можна знайти підручники, курси та приклади коду. Це робить його найкращим вибором для навчання та розвитку у галузі машинного навчання.

За середовище розробки моделей машинного навчання було обрано Jupyter Notebook. Такий вибір можна обґрунтувати наступними перевагами JN [28]:

1. Інтерактивний режим. Jupyter Notebook надає інтерактивну робочу область, де можна виконувати код по частинах та миттєво отримувати результати. Це дозволяє швидко перевіряти та відлагоджувати код, експериментувати з різними параметрами та налаштуваннями моделі, а також візуалізувати проміжні результати.
2. Зрозумілість та документація. Jupyter Notebook поєднує код та текстові коментарі в одному документі. Це робить його зручним інструментом для документування робіт. У Jupyter Notebook можна використовувати текстові коментарі, заголовки, форматування та математичні рівняння, щоб пояснити кроки роботи, результати та висновки.
3. Широкі можливості візуалізації. Jupyter Notebook дозволяє вбудовувати графіки, діаграми та інші візуалізації безпосередньо в документ. Можна використовувати бібліотеки, такі як Matplotlib, Seaborn, Plotly, щоб створювати візуальні зображення, які допомагають дослідникам розуміти дані, виявляти закономірності та представляти результати ефективним способом.

3.2 Огляд бібліотек Python для машинного навчання

3.2.1 NumPy

Рис. 3.1 – логотип бібліотеки NumPy

NumPy (Numerical Python) – це основна бібліотека для обробки числових даних у машинному навчанні. Вона надає потужні структури даних, функції та інструменти для ефективної роботи з векторами, матрицями та іншими масивами числових даних.

Однією з головних переваг NumPy є його високооптимізована реалізація масивів, яка дозволяє виконувати швидкі операції на великих обсягах даних. NumPy використовує векторизовані операції, що забезпечує велику швидкість виконання обчислень. Це особливо важливо для обробки великих наборів даних, які зустрічаються в машинному навчанні.

NumPy надає потужні функції для виконання математичних операцій, лінійної алгебри, випадкових чисел, оптимізації та багатьох інших обчислень, що часто використовуються у моделюванні. Він також має засоби для індексування, зрізів та інших операцій над масивами, що спрощують роботу з даними та маніпуляції з ними.

Додатково, NumPy інтегрується з іншими бібліотеками машинного навчання, такими як Pandas, SciPy та Scikit-learn, що робить його важливою складовою екосистеми машинного навчання. Він може служити основою для

багатьох операцій над даними, побудови та навчання моделей, а також оцінки результатів.

3.2.2 Pandas

Рис. 3.2 – логотип бібліотеки Pandas

Pandas - це потужна бібліотека для обробки та аналізу даних, яка займає важливе місце у машинному навчанні.

У Pandas наявні два основні типів даних: Series та DataFrame. Series є одновимірним масивом даних з індексами, а DataFrame – двовимірна таблиця з мітками для рядків та стовпців. Ці структури дозволяють зручно маніпулювати даними та виконувати операції над ними.

Також Pandas пропонує численні функції для обробки даних, такі як відсіювання, фільтрація, групування, злиття даних, обчислення статистики тощо. Бібліотека також має можливості для роботи зі строковими даними, часовими рядами, обробки відсутніх значень та виконання операцій над стовпцями.

Пандас є важливою складовою для підготовки даних для моделювання. Вона допомагає видаляти аномалії, обробляти пропущенні значення, масштабувати дані та кодувати категоріальні змінні.

Однією з головних переваг Pandas є швидкодія бібліотеки. Вона побудована на основі бібліотеки NumPy, що забезпечує оптимізовану роботу з багатовимірними масивами даних. Пандас використовує векторизовані

операції, що дозволяють ефективно обробляти великі обсяги даних без необхідності використовувати цикли.

Ще однією корисною функцією Pandas є здатність до роботи з реляційними даними. Завдяки Pandas можна здійснювати операції, подібні до тих, що виконуються в SQL. Наприклад, об'єднання таблиць, фільтрації за умовою, групування та агрегацію даних тощо. Це дозволяє легко виконувати складні маніпуляції з даними та готувати їх для моделювання.

Крім того, Pandas надає засоби для візуалізації даних. З Pandas можна легко створювати графіки та діаграми, які допомагають вивчати взаємозв'язки між змінними, виявляти шум або тренди. Це допомагає краще розуміти досліджувані дані та приймати обґрунтовані рішення щодо побудови та оцінки моделей.

3.2.3 Scikit-learn

Рис. 3.3 – логотип бібліотеки Scikit-learn

Scikit-learn є однією з найпопулярніших бібліотек для машинного навчання у середовищі Python. Вона надає широкий набір алгоритмів машинного навчання, інструменти для підготовки та обробки даних, а також засоби для оцінки та валідації моделей.

Scikit-learn пропонує реалізацію багатьох класичних алгоритмів машинного навчання, таких як методи найближчих сусідів, регресія,

класифікація, кластеризація, дерева рішень, випадкові ліси, метод опорних векторів та багато інших.

Однією з переваг Scikit-learn є її зручний та єдиностроковий інтерфейс. Це спрощує процес побудови та навчання моделей. Крім того, бібліотека надає уніфікований підхід до обробки та підготовки даних, включаючи кодування категоріальних змінних, масштабування даних, обробку відсутніх значень та вибір функцій.

Scikit-learn також надає інструменти для оцінки та валідації моделей. Наприклад, методи розбиття даних на тренувальні та тестові набори, крос-валідацію, підбір оптимальних гіперпараметрів, оцінку точності моделей та інші метрики оцінки якості тощо. Це дозволяє розробникам визначити ефективність побудованих моделей.

3.2.4 Keras

Рис. 3.4 – логотип бібліотеки Keras

Keras є високорівневою бібліотекою для машинного навчання, яка працює на основі фреймворку TensorFlow. Вона надає простий та інтуїтивний інтерфейс для побудови, навчання та оцінки нейронних мереж.

Перевагою Keras є легкість її використання. Бібліотека дозволяє швидко створювати складні моделі нейронних мереж за допомогою легко зрозумілих API та готових будівельних блоків (наприклад, шари нейронів). Дослідники можуть з легкістю визначати архітектуру мережі, налаштовувати гіперпараметри та навчати модель без необхідності писати складний код з низькорівневими деталями.

Keras підтримує широкий спектр типів нейронних мереж, включаючи звичайні шаровані нейронні мережі, рекурентні нейронні мережі, згорткові нейронні мережі та інші. Це дозволяє використовувати Keras для широкого спектру завдань машинного навчання, включаючи класифікацію, регресію, сегментацію зображень, розпізнавання об'єктів тощо.

Також Keras має вбудовані інструменти для оцінки та валідації моделей, які допомагають аналізувати результати та оцінювати ефективність моделей. Вона підтримує різні метрики та оцінки, а також можливість розрахунку власних метрик. Ще однією перевагою бібліотеки є те, що Keras дозволяє використовувати різні оптимізатори для налаштування процесу навчання моделі, це дозволяє покращити результати.

3.3 Реалізація програмного продукту

Для створення програми, здатної моделювати результати тенісних матчів потрібно було виконати наступні кроки:

1) знайти датасет, що містить відомості та статистику про історичні матчі;
2) провести підготовку датасету, а саме почистити дані та трансформувати їх до вигляду, в якому буде відбуватися прогнозування;
3) побудувати модель логістичної регресії;
4) побудувати мультишаровий персептрон;
5) реалізувати алгоритм прогнозування статистики майбутнього матчу на основі статистики попередніх ігор;
6) реалізувати алгоритм трансформування нових даних до вигляду, в якому буде проводитись прогнозування, а також створити базу даних (датасет), за допомогою якої буде проводитись прогнозування статистики.

3.3.1 Вибір датасету

Дані про історичні матчі було взято з Github-репозиторію користувача JeffSackmann [13]. Відомості про матчі зберігалися в окремих файлах, розподілених по роках. А також ігри було розділено відповідно до туру, у якому вони відбувалися.

У даній роботі, як вже було зазначено у пункті 1.4.2, використовувалися дані про матчі турів ATP та ATP Challenger. Також було прийнято рішення використовувати дані про ігри від 2000 року. Загалом після об’єднання всіх даних в один датасет було отримано записи про 225873 матчі. Кожен запис містив інформацію, яка налічувала 49 пунктів (наприклад, відомості про турнір, гравців та статистику матчу тощо). Варто зазначити, що у подальшому не всі пункти будуть використовуватися для прогнозування.

3.3.2 Підготовка датасету

Після того, як всі дані було зібрано в один датасет, необхідно було провести їх очистку, щоб забезпечити можливість подальшої роботи. Для цього було зроблено наступне:

- видалено дані про матчі, які складалися з п’яти сетів (турнірна категорія Grand Slam), оскільки, як було зазначено у пункті 1.4.3, нас цікавить тільки трьохсетовий формат;
- видалено колонки, які не несуть важливої інформації для прогнозування. Наприклад, унікальні ідентифікатори турніру та гравців, громадянство гравців, порядковий номер матчу тощо;
- видалено записи, які не містили інформацію про статистику матчів;

- видалено записи, де гравці отримували технічну перемогу, та записи, де один з гравців знімався з гри по ходу матчу (через травму). Відповідно матч не було дограно до кінця та статистика виявилася б нерелевантною для майбутнього прогнозування;
- видалено записи, у яких відсутня інформація про гравців. Наприклад, їх зріст, вік, ведуча рука, рейтинг, кількість рейтингових очок.

У результаті таких маніпуляцій із даними об’єм датасету зменшився до 103869 записів та до 37 колонок вхідної інформації.

Роботу із даними було продовжено. Тепер їх потрібно було привести до вигляду, у якому буде здійснюватися прогнозування. Для цього було зроблено наступні кроки:

- попередньо оброблено колонку, яка містила рахунок матчів. З неї було отримано та додано до датасету інформацію про кількість геймів, виграних кожним гравцем та загальну кількість геймів у матчі;
- оброблено статистику матчів та переведено параметри ігор до відсоткового співвідношення. (Наприклад, не 17 виграних очок з 30 на своїй подачі, а 0.57). Після виконання цього пункту розмір датасету складав 103869 рядків та 48 колонок;
- датасет було продубльовано. У копії змінено місцями значення колонок, що містили інформацію про гравців та їхню статистику. Тобто, якщо в оригінальному датасеті спочатку вся інформація стосується переможців, а потім програвших, то у другому датасеті все навпаки: спочатку гравці, які матч програли, а потім ті, які виграли. Така дія зумовлена тим, що майбутні моделі будуть прогнозувати перемогу лише першого у списку гравця, саме тому нам необхідно створити однакову кількість як виграшних матчів, так і програшних;
- до оригінального датасету додано цільову змінну «1» (тобто перший гравець переміг), а до дубльованого «0» (перший гравець програв). Обидва датасети було скомбіновано в один. У результаті отримано 207738 рядків та 48 колонок;

- видалено непотрібні для побудови моделі колонки (змінні). Наприклад, назву турніру, дату турніру, імена гравців;
- застосовано процедуру One Hot Encoding. Її суть полягає у перетворенні категоріальних змінних на числові. Наприклад, у датасеті існує змінна «ведуча рука гравця», де можливими значеннями є «права» та «ліва». Після застосування One Hot Encoding змінна буде мати назву «ведуча рука гравця права» та значення будуть «1» (якщо ведуча рука права) та «0» (якщо ведуча рука ліва). Цей метод було застосовано до всіх категоріальних змінних;
- застосовано процедуру, яка полягає у знаходженні різниці між числовими значеннями параметрів двох гравців та їх статистикою. Наприклад, замість змінних «вік гравця 1» та «вік гравця 2», було отримано одну змінну «різниця у віці між гравцем 1 та гравцем 2».

Остаточно після реалізації останнього кроку було отримано готовий до навчання моделей датасет розмірністю (207738, 55), де п’ятдесят п’ята змінна є цільовою. У таблиці 3.1 наведено назви всіх змінних, які застосовуватимуться для прогнозування, та розшифровано їх зміст.

Таблиця 3.1 – інформація про предикторні змінні

№	Назва	Зміст
1-4	surface_Carpet, surface_Clay, surface_Grass, surface_Hard	У залежності від покриття корту, на якому грається матч, відповідній змінній присвоюється значення «1», іншим «0»
5-16	draw_size_8, draw_size_12, draw_size_16, draw_size_18, draw_size_24, draw_size_28, draw_size_32, draw_size48, draw_size_56, draw_size_64, draw_size_96, draw_size_128	У залежності від розміру турнірної сітки, відповідній змінній присвоюється значення «1», іншим «0»

Продовження таблиці 3.1

№	Назва	Зміст
17-20	tourney_level_A, tourney_level_C, tourney_level_F, tourney_level_M	У залежності від категорії турніру, відповідній змінній присвоюється значення «1», іншим «0»
21	p1_hand_R	Ведуча рука першого гравця. «1», якщо права, «0» інакше
22	p2_hand_R	Ведуча рука другого гравця. «1», якщо права, «0» інакше
23-35	round_BR, round_ER, round_F, round_Q1, round_Q2, round_Q3, round_QF, round_R128, round_R16, round_R32, round_R64, round_RR, round_SF	У залежності від раунду турніру, відповідній змінній присвоюється значення «1», іншим «0»
36	ht_diff	Різниця між гравцем 1 та гравцем 2 у зрості
37	age_diff	Різниця між гравцем 1 та гравцем 2 у віці
38	rank_diff	Різниця між гравцем 1 та гравцем 2 у номері рейтингу
39	rank_pts_diff	Різниця між гравцем 1 та гравцем 2 у кількості рейтингових очок
40	1st_in_diff	Різниця між гравцем 1 та гравцем 2 у відсотку влучань першою подачею
41	1st_ptwon_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних м’ячів на першій подачі
42	2nd_ptwon_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних м’ячів на другій подачі
43	bp_saved_diff	Різниця між гравцем 1 та гравцем 2 у відсотку відіграних брейк-поінтів

Кінець таблиці 3.1

№	Назва	Зміст
44	ace_per_s_diff	Різниця між гравцем 1 та гравцем 2 у середній кількості ейсів за гейм на подачі
45	df_per_s_diff	Різниця між гравцем 1 та гравцем 2 у середній кількості подвійних помилок за гейм на подачі
46	1st_r_ptwon_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних м’ячів на першому прийомі
47	2nd_r_ptwon_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних м’ячів на другому прийомі
48	bp_won_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних брейк-поінтів
49	tot_pt_won_s_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних очок на подачі
50	tot_pt_won_r_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних очок на прийомі
51	tot_pt_won_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних очок загалом
52	gms_won_s_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних геймів на подачі
53	gms_won_r_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних геймів на прийомі
54	tot_gms_won_diff	Різниця між гравцем 1 та гравцем 2 у відсотку виграних геймів загалом

3.3.3 Побудова моделі логістичної регресії

Після підготовки даних було створено модель логістичної регресії. У ході експериментів з різними параметрами моделі та після використання алгоритму пошуку по сітці (GridSearch) було знайдено оптимальну модель, усі найкращі параметри якої зведено до таблиці 3.2.

Таблиця 3.2 – найкращі параметри моделі логістичної регресії

Назва параметра	Опис	Значення
test_size	Кількість даних розбиття для тестового набору	0.1
solver	Метод знаходження оптимальних ваг моделі, які мінімізують функцію втрат	newton-sg
fit_intercept	Наявність вільного коефіцієнту у рівнянні моделі	True
C	Параметр регуляризації	10
penalty	Тип регуляризації, яка використовується для контролю за перенавчанням моделі.	L2

Після побудови моделі із зазначеними параметрами було проведено оцінку її якості на тренувальному та тестовому наборах. Отримані результати занесено до таблиці 3.3.

Таблиця 3.3 – оцінка якості моделі логістичної регресії

Вибірка	Назва метрики					
	Accuracy	Precision	Recall	F1 Score	Roc Auc	Loss
Навчальна	0.95286	0.95283	0.95300	0.95292	0.99222	0.11329
Тестова	0.95047	0.95031	0.94966	0.94998	0.99195	0.11535

З огляду на отримані результати можна зазначити, що модель досить точно виконує прогнозування. Значення всіх метрик є високими. До того ж модель не схильна до перенавчання, оскільки значення метрик на навчальній та тестовій вибірках є дуже близькими.

3.3.4 Побудова мультишарового персептрону

Для побудови, навчання та прогнозування результатів за допомогою мультишарового персептрону не використовувались змінні 38 та 39, зазначені у таблиці 3.1.

Шляхом неодноразових експериментів було визначено структуру перспептрону: він складається із вхідного шару, трьох прихованих та одного вихідного. За допомогою пошуку по сітці для кожного шару було визначено оптимальні параметри: кількість нейронів, функцію активації, а також оптимізаційний алгоритм, який відповідає за оновлення ваг мережі. Також підібрано і інші параметри, всі вони занесені до таблиці 3.4.

Таблиця 3.4 – параметри мультишарового персептрону

Назва	Опис	Значення
test_size	Кількість даних розбиття для тестового набору	0.1
units (шар 1)	Кількість нейронів у першому шарі	64
units (шар 2)	Кількість нейронів у другому шарі	32
units (шар 3)	Кількість нейронів у третьому шарі	32
activation (шар 1)	Функція активації у першому шарі	tanh
activation (шар 2)	Функція активації у другому шарі	relu
activation (шар 3)	Функція активації у третьому шарі	tanh
units (вихідний шар)	Кількість нейронів у вихідному шарі	1
activation (вихідний шар)	Функція активації у вихідному шарі	sigmoid

Кінець таблиці 3.4

Назва	Опис	Значення
activation (вихідний шар)	Функція активації у вихідному шарі	sigmoid
optimizer	Оптимізаційний алгоритм, який відповідає за оновлення ваг мережі	adam
learning_rate	Параметр оптимізатора, визначає крок або швидкість оновлення вагів моделі під час навчання.	0.0001
loss	Функція втрати, яка використовується для навчання моделі.	Binary_crossentropy
epochs	Кількість повних проходів через навчальний набір даних під час навчання моделі.	30
batch_size	Кількість прикладів, які модель обробляє перед оновленням вагів.	None

Варто зазначити, що для запобігання перенавчанню моделі було застосовано метод раннього відключення (Early Stopping), описаний у розділі 2.3.4.

Після побудови мультишарового персептрону із зазначеними параметрами було проведено оцінку його якості на тренувальному та тестовому наборах. Отримані результати занесено до таблиці 3.5.

Таблиця 3.5 – оцінка якості мультишарового персептрону

Вибірка	Назва метрики					
	Accuracy	Precision	Recall	F1 Score	Roc Auc	Loss
Навчальна	0.95260	0.94839	0.95750	0.95148	0.99216	0.1137
Тестова	0.95283	0.94662	0.95974	0.95162	0.99216	0.1141

Як і у випадку з моделлю логістичної регресії перпептрон досить точно виконує прогнозування. Значення всіх метрик є високими. Перенавчання відсутнє, оскільки значення метрик на навчальній та тестовій вибірках є майже однаковими.

3.3.5 Реалізація алгоритму прогнозування статистики

Алгоритм прогнозування статистики створений на основі процедури, описаної у 1.5. Для його реалізації необхідно мати базу даних (датасет) з результатами попередніх матчів конкретного гравця та дату майбутнього матчу (або турніру). Покроково алгоритм можна описати так:

1. Обчислити різницю місяців між датами матчів у датасеті та датою майбутнього матчу.
2. Обрахувати вагу для кожного історичного матча за формулою, описаною у 1.5.
3. Просумувати всі обраховані ваги.
4. Пронормувати ваги шляхом їх ділення на значення з пункту 3.
5. Помножити нормовані ваги на кожну зі статистичних колонок.
6. Знайти суму кожної колонки. Це і буде прогнозоване значення окремого статистичного показника.

3.3.6 Моделювання результату матчу

Для того, щоб мати можливість змоделювати результат конкретного матчу, користувач має задати його параметри, які будуть оброблені програмою та приведені до вигляду, який розуміють побудовані у 3.3.3 та 3.3.4 моделі.

Також потрібно мати доступ до бази даних, у якій зберігається інформація про статистику всіх попередніх матчів гравців, щоб мати можливість застосувати алгоритм, описаний у 3.3.5.

Для створення такої бази даних були застосовані алгоритми, схожі на ті, що описувались у 3.3.2, тобто очищено початковий датасет та перетворено необхідні колонки до потрібного вигляду. У результаті отримано базу даних розмірності (280384, 24). Змінні, що містяться у ньому, наступні: назва турніру, тип покриття, розмір сітки, рівень турніру, дата його проведення, ім’я гравця, раунд, усі статистичні показники, зазначені у таблиці 3.1, але тільки для одного гравця (гравці чергуються, тобто з одного матчу виходить два записи відносно кожного гравця), а також ім’я суперника та результат матчу («0» або «1» в залежності від результату).

Для моделювання результату матчу користувач має ввести наступні змінні:

1) покриття корту, на якому відбуватиметься матч;
2) розмір сітки турніру;
3) рівень турніру;
4) раунд, у якому буде зіграно матч;
5) ведучі руки обох гравців (записуються в окремі змінні);
6) зріст обох гравців (записуються в окремі змінні);
7) дату народження обох гравців (записуються в окремі змінні);
8) номер рейтингу обох гравців (записуються в окремі змінні);
9) кількість рейтингових очок обох гравців (записуються в окремі змінні);
10) дату майбутнього матчу;
11) імена обох гравців (записуються в окремі змінні). За допомогою імен буде створюватись фільтрація бази даних із матчами для пошуку матчів вказаних гравців, щоб спрогнозувати їх статистику у майбутній грі.

Приклад результату роботи створеної програми показано на рис. 3.1.

```
ПРОГНОЗ ЗА ДОПОМОГОЮ МОДЕЛІ ЛОГІСТИЧНОЇ РЕГРЕСІЇ:

Прогноз на усіх покриттях: Ймовірність перемоги Andrea Pellegrino становить 0.79
Прогноз на покритті Clay: Ймовірність перемоги Andrea Pellegrino становить 0.89

ПРОГНОЗ ЗА ДОПОМОГОЮ МУЛЬТИШАРОВОГО ПЕРСЕПТРОНУ:

Прогноз на усіх покриттях: Ймовірність перемоги Andrea Pellegrino становить 0.67
Прогноз на покритті Clay: Ймовірність перемоги Andrea Pellegrino становить 0.82
```

Рис. 3.1 – приклад роботи створеної програми

3.4 Моделювання результатів турніру в Римі

У цьому розділі буде застосовано створену у попередніх розділах програму для моделювання результатів матчів на ґрунтовому турнірі серії Мастерс в Римі (розмір сітки 128). Після кожного раунду датасет зі статистикою зіграних матчів буде оновлюватись для більш точного моделювання. Всього буде спрогнозовано 131 матч. Результати будемо зводити до таблиць 3.6 – 3.14 в залежності від раунду змагань. Таблиці міститимуть імена гравців, що грають між собою, коефіцієнти букмекерської компанії Favbet, щодо ймовірності перемоги кожного гравця (чим нижчий коефіцієнт, тим більше шансів на перемогу за версією бк), а також результати роботи створених раніше моделей. Будемо записувати ім’я прогнозованого переможця та зазначати у дужках ймовірність такого прогнозу. Прогнозування статистики буде здійснюватися двома способами: по всіх покриттях та лише на обраному покритті (для цього турніру це ґрунт). Правдиві прогнози фарбуватимемо у зелений колір, а неправильні у червоний. Матчі, які не буде дограно до кінця (наприклад, через відмову одного з гравців) позначатимемо жовтим кольором. Зазначимо, що букмекерські коефіцієнти було взято із сайту Flashscore [29].

Таблиця 3.6 – Моделювання результатів першого кваліфікаційного раунду (Q1)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Попирін А.	1.40	Попирін А. (0.66)	Попирін А. (0.65)	Попирін А. (0.67)	Попирін А. (0.64)
	Бонадіо Р.	2.75				
2	Пеллегріно А.	1.68	Пеллегріно А. (0.8)	Пеллегріно А. (0.89)	Пеллегріно А. (0.81)	Пеллегріно А. (0.9)
	Бранкаччо Р.	2.10				
3	Альтмаєр Д.	1.25	Альтмаєр Д. (0.9)	Альтмаєр Д. (0.92)	Альтмаєр Д. (0.89)	Альтмаєр Д. (0.91)
	Ковалік Й.	3.70				
4	Імер Е.	2.75	Махач Т. (0.94)	Махач Т. (0.92)	Махач Т. (0.94)	Махач Т. (0.9)
	Махач Т.	1.41				
5	Мюллер А.	1.25	Мюллер А. (0.87)	Мюллер А. (0.59)	Мюллер А. (0.84)	Мюллер А. (0.48)
	Феррарі Дж.	3.70				
6	Агаменоне Ф.	1.66	Агаменоне Ф. (0.55)	Агаменоне Ф. (0.98)	Агаменоне Ф. (0.62)	Агаменоне Ф. (0.99)
	Віртанен О.	2.10				
7	Шевченко А.	1.27	Шевченко А. (0.8)	Шевченко А. (0.8)	Шевченко А. (0.81)	Шевченко А. (0.83)
	Крутих О.	3.50				
8	Пеністон Р.	4.20	Філс А. (0.9)	Філс А. (0.99)	Філс А. (0.89)	Філс А. (0.99)
	Філс А.	1.20				
9	Чжан Ч.	1.27	Чжан Ч. (0.88)	Чжан Ч. (0.93)	Чжан Ч. (0.89)	Чжан Ч. (0.93)
	Маестреллі Ф.	3.50				
10	Вавассорі А.	1.38	Вавассорі А. (0.98)	Вавассорі А. (0.98)	Вавассорі А. (0.98)	Вавассорі А. (0.98)
	Мартінес П.	2.90				
11	Галан Д. Е.	1.44	Копржіва В. (0.53)	Галан Д. Е. (0.64)	Копржіва В. (0.55)	Галан Д. Е. (0.62)
	Копржіва В.	2.65				
12	Соуза Ж.	3.70	Сафіуллін Р. (0.99)	Сафіуллін Р. (0.99)	Сафіуллін Р. (0.99)	Сафіуллін Р. (0.99)
	Сафіуллін Р.	1.25				
13	Черундоло Х. М.	1.13	Черундоло Х. М. (0.95)	Черундоло Х. М. (0.98)	Черундоло Х. М. (0.93)	Черундоло Х. М. (0.97)
	Селс Є.	5.40				
14	Белуччі М.	2.75	Барріос Вера Т. (0.99)	Барріос Вера Т. (1)	Барріос Вера Т. (0.99)	Барріос Вера Т. (1)
	Барріос Вера Т.	1.40				

Кінець таблиці 3.6

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
15	Ковачевич А. |	1.60	Наполітано С. (0.64)	Наполітано С. (1)	Наполітано С. (0.75)	Наполітано С. (1)
	Наполітано С.	2.20				
16	Мартерер М.	2.27	Уго Карбельї К. (0.68)	Уго Карбельї К. (0.99)	Уго Карбельї К. (0.71)	Уго Карбельї К. (0.99)
	Уго Карбельї К.	1.59				
17	Ханфманн Я.	1.13	Ханфманн Я. (0.68)	Ханфманн Я. (0.67)	Ханфманн Я. (0.67)	Ханфманн Я. (0.68)
	Оклеппо Ю.	5.40				
18	Кольяріні А.	1.70	Кольяріні А. (0.88)	Кольяріні А. (0.97)	Кольяріні А. (0.83)	Кольяріні А. (0.96)
	Гренье Ю.	2.05				
19	Даніель Т.	1.25	Даніель Т. (0.97)	Даніель Т. (0.99)	Даніель Т. (0.97)	Даніель Т. (1)
	Нава Е.	3.70				
20	Коболлі Ф.	1.25	Коболлі Ф. (0.95)	Коболлі Ф. (1)	Коболлі Ф. (0.91)	Коболлі Ф. (1)
	Басилашвілі Н.	3.65				
21	Гойо Б.	1.44	Гойо Б. (0.88)	Гойо Б. (0.83)	Гойо Б. (0.87)	Гойо Б. (0.81)
	Мелігені Родрігеш Алвеш Ф.	2.60				
22	Скатов Т.	1.99	Марожан Ф. (0.7)	Марожан Ф. (0.74)	Марожан Ф. (0.79)	Марожан Ф. (0.82)
	Марожан Ф.	1.76				
23	Коккінакіс Т.	1.34	Гіганте М. (0.58)	Гіганте М. (0.51)	Гіганте М. (0.54)	Гіганте М. (0.54)
	Гіганте М.	3.10				
24	Хойнскі Я.	2.31	Хойнскі Я. (0.87)	Хойнскі Я. (0.97)	Хойнскі Я. (0.9)	Хойнскі Я. (0.98)
	Родіонов Ю.	1.57				

Таблиця 3.7 – Моделювання результатів другого кваліфікаційного раунду (Q2)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Попирін А.	1.40	Попирін А. (0.51)	Попирін А. (0.51)	Попирін А. (0.54)	Попирін А. (0.51)
	Пеллегріно А.	2.75				
2	Альтмаєр Д.	1.25	Альтмаєр Д. (0.94)	Альтмаєр Д. (0.91)	Альтмаєр Д. (0.95)	Альтмаєр Д. (0.91)
	Імер Е.	3.70				
3	Мюллер А.	1.66	Агаменоне Ф. (0.56)	Агаменоне Ф. (0.7)	Агаменоне Ф. (0.71)	Агаменоне Ф. (0.82)
	Агаменоне Ф.	2.10				
4	Шевченко А.	1.45	Шевченко А. (0.51)	Шевченко А. (0.68)	Філс А. (0.55)	Шевченко А. (0.62)
	Філс А.	2.60				
5	Маестреллі Ф.	2.20	Мартінес П. (0.59)	Мартінес П. (0.73)	Мартінес П. (0.66)	Мартінес П. (0.78)
	Мартінес П.	1.60				
6	Копржіва В.	3.05	Сафіуллін Р. (0.77)	Сафіуллін Р. (0.85)	Сафіуллін Р. (0.88)	Сафіуллін Р. (0.92)
	Сафіуллін Р.	1.35				
7	Черундоло Х. М.	1.66	Барріос Вера Т. (0.66)	Барріос Вера Т. (0.66)	Барріос Вера Т. (0.8)	Барріос Вера Т. (0.82)
	Барріос Вера Т.	2.10				
8	Наполітано С.	2.75	Наполітано С. (0.92)	Наполітано С. (0.99)	Наполітано С. (0.94)	Наполітано С. (1)
	Мартерер М.	1.40				
9	Ханфманн Я.	1.20	Ханфманн Я. (0.94)	Ханфманн Я. (0.99)	Ханфманн Я. (0.94)	Ханфманн Я. (0.99)
	Греньє Ю.	4.20				
10	Нава Е.	3.00	Коболлі Ф. (0.94)	Коболлі Ф. (0.98)	Коболлі Ф. (0.94)	Коболлі Ф. (0.98)
	Коболлі Ф.	1.35				
11	Мелігені Родрігеш Алвеш Ф.	1.81	Марожан Ф. (0.82)	Марожан Ф. (0.82)	Марожан Ф. (0.88)	Марожан Ф. (0.89)
	Марожан Ф.	1.93				
12	Коккінакіс Т.	1.20	Хойнскі Я. (0.71)	Хойнскі Я. (0.68)	Хойнскі Я. (0.81)	Хойнскі Я. (0.84)
	Хойнскі Я.	4.20				

Таблиця 3.8 – Моделювання результатів раунду 1/64 фіналу (R128)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Ван Аш Л.	2.23	Етчеверрі Т. М. (0.61)	Етчеверрі Т. М. (0.61)	Етчеверрі Т. М. (0.69)	Етчеверрі Т. М. (0.7)
	Етчеверрі Т. М.	1.66				
2	Івашка І.	2.60	Ваврінка С. (0.57)	Івашка І. (0.62)	Ваврінка С. (0.68)	Ваврінка С. (0.54)
	Ваврінка С.	1.50				
3	Країнович Ф.	1.98	Фучовіч М. (0.77)	Країнович Ф. (0.52)	Фучовіч М. (0.81)	Країнович Ф. (0.52)
	Фучовіч М.	1.82				
4	Едмунд К.	4.10	Мюллер А. (0.99)	Мюллер А. (0.99)	Мюллер А. (0.99)	Мюллер А. (0.99)
	Мюллер А.	1.24				
5	Попирін А.	1.80	О'Коннелл К. (0.52)	О'Коннелл К. (0.54)	О'Коннелл К. (0.58)	О'Коннелл К. (0.63)
	О'Коннелл К.	2.00				
6	Сафіуллін Р.	1.57	Сафіуллін Р. (0.65)	Сафіуллін Р. (0.7)	Сафіуллін Р. (0.61)	Сафіуллін Р. (0.62)
	Гірон М.	2.41				
7	Фоньїні Ф.	2.65	Маррей Е. (0.8)	Маррей Е. (0.62)	Маррей Е. (0.83)	Маррей Е. (0.75)
	Маррей Е.	1.48				
8	Черундоло Х. М.	2.00	Філс А. (0.63)	Черундоло Х. М. (0.56)	Філс А. (0.62)	Черундоло Х. М. (0.59)
	Філс А.	1.81				
9	Коболлі Ф.	1.39	Коболлі Ф. (0.74)	Рендеркнек А. (0.79)	Коболлі Ф. (0.6)	Рендеркнек А. (0.85)
	Рендеркнек А.	3.00				
10	Мартінес П.	1.69	Мартінес П. (0.61)	Мартінес П. (0.93)	Бублік А. (0.52)	Мартінес П. (0.9)
	Бублік А.	2.18				
11	Лестьєнн К.	6.00	Дьєре Л. (0.74)	Дьєре Л. (0.98)	Дьєре Л. (0.73)	Дьєре Л. (0.98)
	Дьєре Л.	1.13				
12	Качін П.	2.35	Гарін К. (0.64)	Качін П. (0.65)	Гарін К. (0.63)	Качін П. (0.69)
	Гарін К.	1.60				
13	Накашима Б.	1.72	Баррер Г. (0.8)	Баррер Г. (0.73)	Баррер Г. (0.84)	Баррер Г. (0.8)
	Баррер Г.	2.11				
14	Гаске Р.	1.40	Ву Ї. (0.51)	Гаске Р. (1)	Гаске Р. (0.82)	Гаске Р. (1)
	Ву Ї.	3.00				
15	Баес С.	1.40	Варільяс Х. П. (0.71)	Варільяс Х. П. (0.56)	Варільяс Х. П. (0.8)	Варільяс Х. П. (0.7)
	Варільяс Х. П.	2.95				

Продовження таблиці 3.8

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
16	Мунар Х.	2.11	Коккінакіс Т. (0.85)	Коккінакіс Т. (0.78)	Коккінакіс Т. (0.9)	Коккінакіс Т. (0.8)
	Коккінакіс Т.	1.73				
17	Наполітано С.	5.20	Наполітано С. (0.82)	Наполітано С. (0.69)	Наполітано С. (0.84)	Наполітано С. (0.75)
	Молчан А.	1.16				
18	Пелья Г.	2.00	Пелья Г. (0.69)	Пелья Г. (0.96)	Пелья Г. (0.71)	Пелья Г. (0.97)
	Кресі М.	1.81				
19	Чеккінато М.	1.60	? (0.5)	Чеккінато М. (0.95)	Макдональд М. (0.57)	Чеккінато М. (0.95)
	Макдональд М.	2.30				
20	Харрі Н.	1.82	Ханфманн Я. (0.68)	Ханфманн Я. (0.69)	Ханфманн Я. (0.73)	Ханфманн Я. (0.75)
	Ханфманн Я.	1.98				
21	Греньє Ю.	2.21	Вулф Дж. Дж. (0.79)	Вулф Дж. Дж. (0.91)	Вулф Дж. Дж. (0.83)	Вулф Дж. Дж. (0.93)
	Вулф Дж. Дж.	1.67				
22	Нарді Л.	2.60	Гоффен Д. (0.64)	Гоффен Д. (0.66)	Гоффен Д. (0.81)	Гоффен Д. (0.82)
	Гоффен Д.	1.50				
23	Хюслер М.	1.99	Кублер Дж. (0.63)	Кублер Дж. (0.53)	Кублер Дж. (0.66)	Кублер Дж. (0.51)
	Кублер Дж.	1.82				
24	Юмбер Ю.	2.35	Юмбер Ю. (0.6)	Руусувуорі Е. (0.64)	Юмбер Ю. (0.62)	Руусувуорі Е. (0.6)
	Руусувуорі Е.	1.60				
25	Лайович Д.	1.36	Лайович Д. (0.67)	Лайович Д. (0.98)	Лайович Д. (0.77)	Лайович Д. (0.99)
	Боргеш Н.	3.20				
26	Сонего Л.	1.09	Сонего Л. (0.99)	Сонего Л. (0.93)	Сонего Л. (0.99)	Сонего Л. (0.88)
	Шарді Ж.	7.40				
27	Арнальді М.	1.59	Арнальді М. (0.92)	Арнальді М. (0.96)	Арнальді М. (0.92)	Арнальді М. (0.96)
	Шварцман Д.	2.35				
28	Дзепп'єрі Дж.	2.50	Альтмаєр Д. (0.62)	Альтмаєр Д. (0.76)	Альтмаєр Д. (0.73)	Альтмаєр Д. (0.85)
	Альтмаєр Д.	1.53				
29	Маннаріно А.	2.60	Монтейро Т. (0.51)	Монтейро Т. (0.96)	Маннаріно А. (0.61)	Монтейро Т. (0.96)
	Монтейро Т.	1.50				
30	Дельєн У.	2.80	Дельєн У. (0.9)	Дельєн У. (0.57)	Дельєн У. (0.92)	Дельєн У. (0.57)
	Карбальєс Баена Р.	1.44				

Кінець таблиці 3.8

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
31	Муте К.	2.39	Марожан Ф. (0.8)	Марожан Ф. (0.9)	Марожан Ф. (0.82)	Марожан Ф. (0.92)
	Марожан Ф.	1.57				
32	Рамос А.	1.82	Пассаро Ф. (0.9)	Пассаро Ф. (0.94)	Пассаро Ф. (0.79)	Пассаро Ф. (0.87)
	Пассаро Ф.	1.99				

Таблиця 3.9 – Моделювання результатів раунду 1/32 фіналу (R64)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Джокович Н.	1.11	Джокович Н. (0.74)	Етчеверрі Т. М. (0.67)	Джокович Н. (0.79)	Етчеверрі Т. М. (0.62)
	Етчеверрі Т. М.	6.60				
2	Ваврінка С.	2.01	Дімітров Г. (0.76)	Дімітров Г. (0.73)	Дімітров Г. (0.68)	Дімітров Г. (0.66)
	Дімітров Г.	1.80				
3	Де Мінор А.	1.63	Де Мінор А. (0.62)	Де Мінор А. (0.57)	Де Мінор А. (0.61)	Де Мінор А. (0.61)
	Фучовіч М.	2.28				
4	Мюллер А.	3.70	Мюллер А. (0.6)	Мюллер А. (0.57)	Мюллер А. (0.62)	Мюллер А. (0.6)
	Норрі К.	1.28				
5	Оже Альяссім Ф.	1.37	Попирін А. (0.58)	Попирін А. (0.82)	Попирін А. (0.66)	Попирін А. (0.83)
	Попирін А.	3.10				
6	Сафіуллін Р.	1.69	Сафіуллін Р. (0.59)	Сафіуллін Р. (0.94)	Сафіуллін Р. (0.57)	Сафіуллін Р. (0.94)
	Корда С.	2.18				
7	Кецманович М.	1.44	Кецманович М. (0.76)	Кецманович М. (0.88)	Кецманович М. (0.69)	Кецманович М. (0.86)
	Фоньїні Ф.	2.80				
8	Філс А.	5.40	Руне Х. (0.72)	Руне Х. (0.77)	Руне Х. (0.72)	Руне Х. (0.78)
	Руне Х.	1.15				
9	Рууд К.	1.15	Рууд К. (0.85)	Рендеркнек А. (0.79)	Рууд К. (0.77)	Рендеркнек А. (0.85)
	Рендеркнек А.	5.50				

Продовження таблиці 3.9

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
10	Бублік А.	2.02	Шелтон Б. (0.69)	Шелтон Б. (0.71)	Шелтон Б. (0.73)	Шелтон Б. (0.77)
	Шелтон Б.	1.79				
11	Ван Де Зандсхулп Б.	1.96	Дьєре Л. (0.51)	Дьєре Л. (0.52)	Дьєре Л. (0.59)	Дьєре Л. (0.58)
	Дьєре Л.	1.85				
12	Гарін К.	1.73	Пол Т. (0.67)	Пол Т. (0.72)	Пол Т. (0.69)	Пол Т. (0.74)
	Пол Т.	2.10				
13	Хачанов К.	1.16	Хачанов К. (0.72)	Хачанов К. (0.91)	Хачанов К. (0.64)	Хачанов К. (0.86)
	Баррер Г.	5.30				
14	Ву Ї.	3.60	Черундоло Ф. (0.54)	Черундоло Ф. (0.98)	Черундоло Ф. (0.62)	Черундоло Ф. (0.99)
	Черундоло Ф.	1.29				
15	Шевченко А.	2.75	Шевченко А. (0.83)	Шевченко А. (0.83)	Шевченко А. (0.81)	Шевченко А. (0.81)
	Баес С.	1.45				
16	Коккінакіс Т.	6.40	Зіннер Я. (0.92)	Зіннер Я. (0.96)	Зіннер Я. (0.94)	Зіннер Я. (0.97)
	Зіннер Я.	1.12				
17	Рубльов А.	1.18	Рубльов А. (0.82)	Рубльов А. (0.92)	Рубльов А. (0.77)	Рубльов А. (0.94)
	Молчан А.	4.90				
18	Пелья Г.	4.90	Давідовіч Фокіна А. (0.57)	Пелья Г. (0.62)	Пелья Г. (0.54)	Пелья Г. (0.79)
	Давідовіч Фокіна А.	1.18				
19	Баутіста-Агут Р.	1.60	Чеккінато М. (0.67)	Чеккінато М. (0.51)	Чеккінато М. (0.71)	Чеккінато М. (0.59)
	Чеккінато М.	2.30				
20	Ханфманн Я.	2.75	Фріц Т. (0.64)	Ханфманн Я. (0.52)	Фріц Т. (0.58)	Ханфманн Я. (0.63)
	Фріц Т.	1.45				
21	Хуркач Х.	1.25	Вулф Дж. Дж. (0.51)	Вулф Дж. Дж. (0.8)	Вулф Дж. Дж. (0.56)	Вулф Дж. Дж. (0.84)
	Вулф Дж. Дж.	3.95				
22	Гоффен Д.	4.30	Звєрєв А. (0.69)	Звєрєв А. (0.92)	Звєрєв А. (0.72)	Звєрєв А. (0.93)
	Звєрєв А.	1.22				
23	Сапата Міральєс Б.	1.57	Кублер Дж. (0.57)	Кублер Дж. (0.57)	Кублер Дж. (0.73)	Кублер Дж. (0.71)
	Кублер Дж.	2.40				

Кінець таблиці 3.9

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
24	Руусувуорі Е.	3.50	Медвєдєв Д. (0.92)	Медвєдєв Д. (0.66)	Медвєдєв Д. (0.93)	Медвєдєв Д. (0.68)
	Медвєдєв Д.	1.30				
25	Ціціпас С.	1.11	Ціціпас С. (0.62)	Ціціпас С. (0.95)	Боргеш Н. (0.55)	Ціціпас С. (0.92)
	Боргеш Н.	6.60				
26	Сонего Л.	1.44	Сонего Л. (0.65)	Нісіока Й. (0.62)	Сонего Л. (0.69)	Нісіока Й. (0.56)
	Нісіока Й.	2.80				
27	Музетті Л.	1.37	Музетті Л. (0.82)	Музетті Л. (0.55)	Музетті Л. (0.75)	Арнальді М. (0.58)
	Арнальді М.	3.10				
28	Альтмаєр Д.	2.50	Альтмаєр Д. (0.65)	Альтмаєр Д. (0.76)	Альтмаєр Д. (0.6)	Альтмаєр Д. (0.76)
	Тіафо Ф.	1.53				
29	Чоріч Б.	1.36	Монтейро Т. (0.53)	Чоріч Б. (0.52)	Монтейро Т. (0.7)	Монтейро Т. (0.66)
	Монтейро Т.	3.15				
30	Карбальєс Баена Р.	1.64	Еванс Д. (0.63)	Карбальєс Баена Р. (0.64)	Еванс Д. (0.74)	Карбальєс Баена Р. (0.56)
	Еванс Д.	2.26				
31	Легечка Ї.	3.00	Марожан Ф. (0.78)	Марожан Ф. (0.85)	Марожан Ф. (0.83)	Марожан Ф. (0.89)
	Марожан Ф.	1.40				
32	Рамос А.	18.00	Алькарас Гарфія К. (1)	Алькарас Гарфія К. (0.99)	Алькарас Гарфія К. (0.98)	Алькарас Гарфія К. (0.97)
	Алькарас Гарфія К.	1.02				

Таблиця 3.10 – Моделювання результатів раунду 1/16 фіналу (R32)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Джокович Н.	1.15	Джокович Н. (0.91)	Джокович Н. (0.62)	Джокович Н. (0.87)	Джокович Н. (0.54)
	Дімітров Г.	5.50				
2	Фучовіч М.	2.80	Норрі К. (0.55)	Норрі К. (0.56)	Норрі К. (0.54)	Норрі К. (0.58)
	Норрі К.	1.44				
	Черундоло Ф.	1.35				

Кінець таблиці 3.10

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
3	Попирін А.	1.72	Сафіуллін Р. (0.67)	Сафіуллін Р. (0.75)	Сафіуллін Р. (0.7)	Сафіуллін Р. (0.8)
	Сафіуллін Р.	2.10				
4	Фоньїні Ф.	4.80	Руне Х. (0.93)	Руне Х. (0.87)	Руне Х. (0.83)	Руне Х. (0.7)
	Руне Х.	1.18				
5	Рууд К.	1.16	Рууд К. (0.95)	Рууд К. (0.98)	Рууд К. (0.91)	Рууд К. (0.97)
	Бублік А.	5.30				
6	Дьєре Л.	1.68	Дьєре Л. (0.53)	Дьєре Л. (0.73)	Дьєре Л. (0.55)	Дьєре Л. (0.73)
	Гарін К.	2.15				
7	Баррер Г.	3.25	Черундоло Ф. (0.54)	Черундоло Ф. (0.9)	? (0.5)	Черундоло Ф. (0.89)
8	Шевченко А.	7.80	Зіннер Я. (0.89)	Зіннер Я. (0.95)	Зіннер Я. (0.89)	Зіннер Я. (0.95)
	Зіннер Я.	1.08				
9	Рубльов А.	1.49	Рубльов А. (0.8)	Рубльов А. (0.95)	Рубльов А. (0.79)	Рубльов А. (0.94)
	Давідовіч Фокіна А.	2.60				
10	Чеккінато М.	2.05	Ханфманн Я. (0.79)	Ханфманн Я. (0.87)	Ханфманн Я. (0.84)	Ханфманн Я. (0.9)
	Ханфманн Я.	1.75				
11	Вулф Дж. Дж.	4.30	Вулф Дж. Дж. (0.51)	Вулф Дж. Дж. (0.66)	Звєрєв А. (0.59)	Звєрєв А. (0.51)
	Звєрєв А.	1.22				
12	Сапата Міральєс Б.	4.10	Медвєдєв Д. (0.97)	Медвєдєв Д. (0.7)	Медвєдєв Д. (0.97)	Медвєдєв Д. (0.71)
	Медвєдєв Д.	1.23				
13	Ціціпас С.	1.18	Ціціпас С. (0.88)	Ціціпас С. (0.94)	Ціціпас С. (0.79)	Ціціпас С. (0.9)
	Сонего Л.	5.00				
14	Музетті Л.	1.50	Музетті Л. (0.66)	Музетті Л. (0.61)	? (0.5)	Тіафо Ф. (0.53)
	Тіафо Ф.	2.60				
15	Чоріч Б.	1.44	Чоріч Б. (0.69)	Карбальєс Баена Р. (0.56)	Чоріч Б. (0.57)	Карбальєс Баена Р. (0.68)
	Карбальєс Баена Р.	2.75				
16	Марожан Ф.	17.50	Алькарас Гарфія К. (0.96)	Алькарас Гарфія К. (0.93)	Алькарас Гарфія К. (0.94)	Алькарас Гарфія К. (0.91)
	Алькарас Гарфія К.	1.01				

Таблиця 3.11 – Моделювання результатів раунду 1/8 фіналу (R16)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Джокович Н.	1.15	Джокович Н. (0.86)	? (0.5)	Джокович Н. (0.89)	Джокович Н. (0.58)
	Норрі К.	5.50				
2	Попирін А.	5.00	Руне Х. (0.78)	Руне Х. (0.81)	Руне Х. (0.74)	Руне Х. (0.79)
	Руне Х.	1.18				
3	Рууд К.	1.40	Рууд К. (0.8)	Рууд К. (0.79)	Рууд К. (0.69)	Рууд К. (0.71)
	Дьєре Л.	3.00				
4	Черундоло Ф.	6.50	Зіннер Я. (0.94)	Зіннер Я. (0.97)	Зіннер Я. (0.93)	Зіннер Я. (0.96)
	Зіннер Я.	1.11				
5	Рубльов А.	1.37	Рубльов А. (0.57)	Рубльов А. (0.72)	Ханфманн Я. (0.56)	Рубльов А. (0.59)
	Ханфманн Я.	3.10				
6	Звєрєв А.	2.15	Медвєдєв Д. (0.95)	Медвєдєв Д. (0.57)	Медвєдєв Д. (0.95)	? (0.5)
	Медвєдєв Д.	1.68				
7	Ціціпас С.	1.38	Музетті Л. (0.59)	Ціціпас С. (0.54)	Музетті Л. (0.57)	Ціціпас С. (0.57)
	Музетті Л.	3.10				
8	Чоріч Б.	1.52	Марожан Ф. (0.79)	Марожан Ф. (0.76)	Марожан Ф. (0.8)	Марожан Ф. (0.76)
	Марожан Ф.	2.60				

Таблиця 3.12 – Моделювання результатів чвертьфіналу (QF)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Джокович Н.	1.33	Джокович Н. (0.7)	Джокович Н. (0.55)	Джокович Н. (0.83)	Джокович Н. (0.72)
	Руне Х.	3.45				
2	Рууд К.	1.64	Рууд К. (0.81)	Рууд К. (0.79)	Рууд К. (0.76)	Рууд К. (0.76)
	Черундоло Ф.	2.33				
3	Ханфманн Я.	3.05	Медвєдєв Д. (0.85)	Ханфманн Я. (0.57)	Медвєдєв Д. (0.81)	Ханфманн Я. (0.63)
	Медвєдєв Д.	1.40				

Кінець таблиці 3.12

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
4	Ціціпас С.	1.28	Ціціпас С. (0.87)	Ціціпас С. (0.9)	Ціціпас С. (0.87)	Ціціпас С. (0.9)
	Чоріч Б.	3.70				

Таблиця 3.13 – Моделювання результатів півфіналу (SF)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Руне Х.	1.60	Руне Х. (0.57)	Рууд К. (0.56)	Руне Х. (0.51)	Рууд К. (0.62)
	Рууд К.	2.39				
2	Медвєдєв Д.	2.20	Медвєдєв Д. (0.8)	Ціціпас С. (0.6)	Медвєдєв Д. (0.85)	Ціціпас С. (0.6)
	Ціціпас С.	1.65				

Таблиця 3.14 – Моделювання результатів фіналу (F)

			Логістична регресія		Мультишаровий персептрон	
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Руне Х.	1.80	Медвєдєв Д. (0.77)	Руне Х. (0.55)	Медвєдєв Д. (0.73)	Руне Х. (0.62)
	Медвєдєв Д.	2.00				

3.5 Аналіз результатів

У цьому розділі буде проведено аналіз роботи створеного програмного продукту на основі отриманих результатів моделювання, наведених у розділі 3.4.

Одним з найбільш важливих показників якості моделі є кількість точних прогнозів, які вона робить. Обрахуємо значення цього показника для кожної власної моделі (всього 4), а також знайдемо кількість точних прогнозів, які робить модель, що використовується компанією Favbet. Результати зведемо у таблицю 3.15.

Таблиця 3.15 – Кількість точних прогнозів моделей

Раунд	Логістична регресія		Мультишаровий персептрон		Модель бк
	Всі покриття	Обране покриття	Всі покриття	Обране покриття	
Q1	15/23 (65%)	14/23 (61%)	15/23 (65%)	13/23 (57%)	13/23 (57%)
Q2	8/12 (67%)	8/12 (67%)	9/12 (75%)	8/12 (67%)	9/12 (75%)
R128	19/29 (66%)	17/30 (57%)	18/30 (60%)	18/30 (60%)	22/30 (73%)
R64	21/32 (66%)	20/32 (63%)	19/32 (59%)	18/32 (56%)	23/32 (72%)
R32	12/15 (80%)	11/15 (73%)	11/13 (85%)	11/15 (73%)	14/15 (93%)
R16	4/8 (50%)	4/7 (57%)	5/8 (63%)	4/7 (57%)	6/8 (75%)
QF	3/4 (75%)	2/4 (50%)	3/4 (75%)	2/4 (50%)	3/4 (75%)
SF	2/2 (100%)	0/2 (0%)	2/2 (100%)	0/2 (0%)	1/2 (50%)
F	1/1 (100%)	0/1 (0%)	1/1 (100%)	0/1 (0%)	0/1 (0%)
Загалом	85/126 (67%)	76/126 (60%)	83/125 (66%)	74/126 (59%)	91/127 (72%)

Після порівняння отриманих результатів бачимо, що модель Favbet показує найбільшу кількість правильних прогнозів – 72%. Однак модель логістичної регресії та мультишаровий персептрон, що використовують алгоритм прогнозування статистики по всіх покриттях, видають не набагато гірший результат: 67% та 66% відповідно. У свою чергу моделі, що використовують алгоритм прогнозування статистики по одному обраному

покриттю, показують найгірші результати: 60% та 59%. Такі моделі не є надійними, хоча результат і є вищим за звичайне вгадування.

Хоча відсоток правильних прогнозів і є показовою метрикою роботи алгоритмів моделювання, але якщо розглядати створені програми як спосіб заробітку (тобто якщо ці програми будуть використовуватися людьми, які роблять ставки), то найбільш важливим показником є не кількість правильно зроблених прогнозів, а саме їх якість, тобто прибуток, який моделі можуть забезпечити на дистанції. Проаналізуємо і цей аспект. Для цього знову створимо таблицю, у яку занесемо прибуток, забезпечений моделями. Будемо аналізувати лише три моделі: модель букмекерської контори Favbet, модель логістичної регресії та мультишаровий персептрон, що використовують алгоритм прогнозування статистики по всіх покриттях.

Нехай початковий капітал гравця (беттора) ставить 13100 умовних одиниць і на кожен матч беттор ставить по 100 умовних одиниць. Тоді отримаємо результати, що наведені у таблиці 3.16

Таблиця 3.16 – Прибуток, який забезпечують моделі

Раунд	ЛР	МП	БК
Q1	+54	+54	-528
Q2	+83	+343	+67
R128	+356	+24	+429
R64	+248	+19	-51
R32	+95	+132	+389
R16	-259	+51	+31
QF	+32	+32	+32
SF	+180	+180	-40
F	+100	+100	-100
Загалом	+889	+935	+229

З таблиці бачимо, що всі три моделі показали позитивний результат. Однак, наші моделі майже у 4 рази перевершили показник моделі Favbet. Якщо перевести отримані виграші у відсотки, то використовуючи модель Favbet для прогнозування турніру в Римі можна було б збільшити початковий капітал на

1.75%. Якщо використовувати модель логістичної регресії у комбінації з алгоритмом прогнозування статистики по всіх покриттях, то капітал зріс би на 6.77%, а якщо використовувати мультишаровий персептрон у комбінації з алгоритмом прогнозування статистики по всіх покриттях, то початковий капітал збільшився б на рекордні 7.14%. Також варто зазначити, що мультишаровий персептрон у комбінації з алгоритмом прогнозування статистики по всіх покриттях у жодному з раундів не програвав, а тільки нарощував виграш.

3.6 Висновки до розділу 3

У цьому розділі було описано інструменти та бібліотеки машинного навчання, які використовувались для побудови моделей МН. Також було описано сам процес побудови моделі логістичної регресії та різновиду нейронної мережі мультишаровий персептрон. Було виконано оцінку якості моделей: обидві вийшли вдалими та показали точність прогнозів на рівні 95%.

У даному розділі було вирішено проблему, з якою стикаються дослідники при моделюванні результатів тенісних матчів, описану у 1.5, а саме прогнозування статистики майбутніх матчів. Було створено алгоритм, який забезпечує можливість такого прогнозування на основі процедури накладання ваг на статистику минулих ігор.

Також у розділі було змодельовано результати всіх матчів на тенісному турнірі серії Мастерс у Римі, проведено аналіз отриманих результатів. Виявилось, що комбінація моделі логістичної регресії та алгоритму прогнозування статистики матчу по всіх покриттях правильно моделює 67% матчів, а комбінація мультишарового персептрону з ідентичним алгоритмом прогнозування статистики 66%. Ці показники хоча й трохи менші, ніж видає модель букмекерської контори Favbet (72% правильних прогнозів), однак у

площині можливого прибутку, який може бути забезпечений в результаті моделювання, наші моделі перевершують букмекерську у 4 рази.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп’ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

- F1 – вибір мови програмування;
- F2 – вибір операційної системи;
- F3 – вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації.

Функція F1 :

А) Python

Б) R

Функція F2 :

А) MacOS

Б) Linux

Функція F3 :

А) Jupyter Notebook

Б) PyCharm

Варіанти реалізації основних функцій наведені у морфологічній карті системи (див. рис. 4.1).

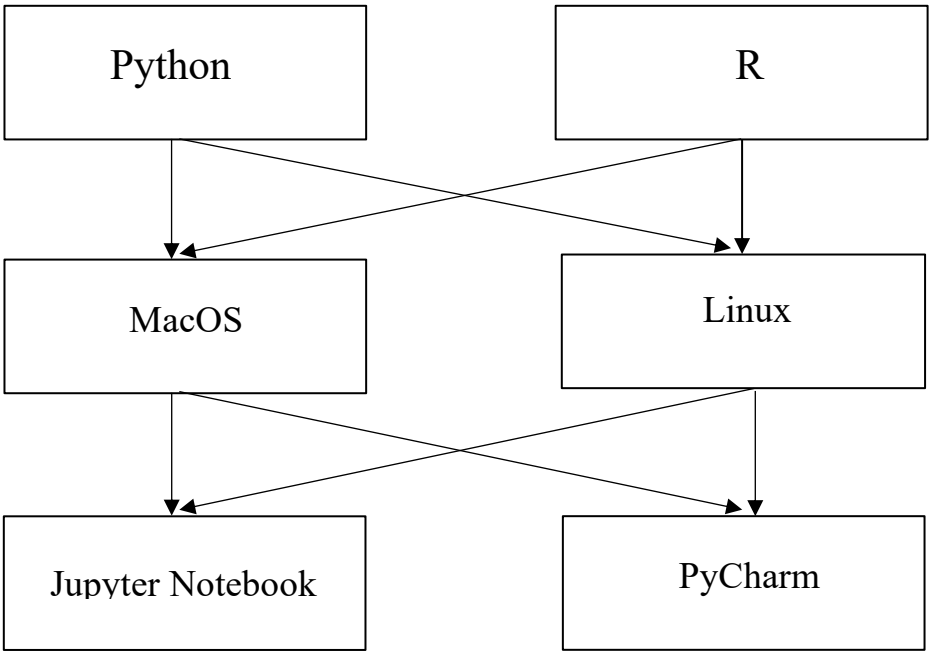


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	А	Широкий вибір бібліотек, простота використання, широке застосування	Повільна швидкодія, особливо при обробці великих обсягів даних або виконанні обчислювально важливих операцій.
	Б	Спрямованість на статистику та аналіз даних, зручність у використанні інтерактивних середовищ	Відсутність підтримки для загального програмування, повільна швидкодія
F_2	А	Інтуїтивний та естетичний інтерфейс, безпека та стабільність використання	Висока вартість, обмежений вибір апаратного забезпечення
	Б	Відкрите програмне забезпечення, надійність та стабільність використання, широкий вибір дистрибутивів	Сумісність програмного забезпечення, складність налаштування
F_3	А	Інтерактивний інтерфейс, легка візуалізація та документація, виконання коду в окремих блоках	Відсутність повноцінного IDE
	Б	Повноцінне IDE, швидкість та продуктивність	Висока вартість повної версії, високі вимоги до ресурсів системи

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Перевагу даємо мові програмування Python. Варіант Б має бути відкинутий.

Функція F_2:

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3:

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

F_1А – F_2А – F_3А

F_1А – F_2Б – F_3А

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X1 – швидкодія мови програмування;
- X2 – об’єм пам’яті, необхідний для обчислень та збереження даних;
- X3 – час навчання даних;
- X4 – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	50	70	100
Об’єм пам’яті	X2	мб	64	32	16
Час навчання даних	X3	мс	500	400	300
Потенційний об’єм програмного коду	X4	кількість рядків коду	1400	1100	800

За даними таблиці 4.2 будуються графічні характеристики параметрів – (див. рисунок 4.2 – рисунок 4.5).

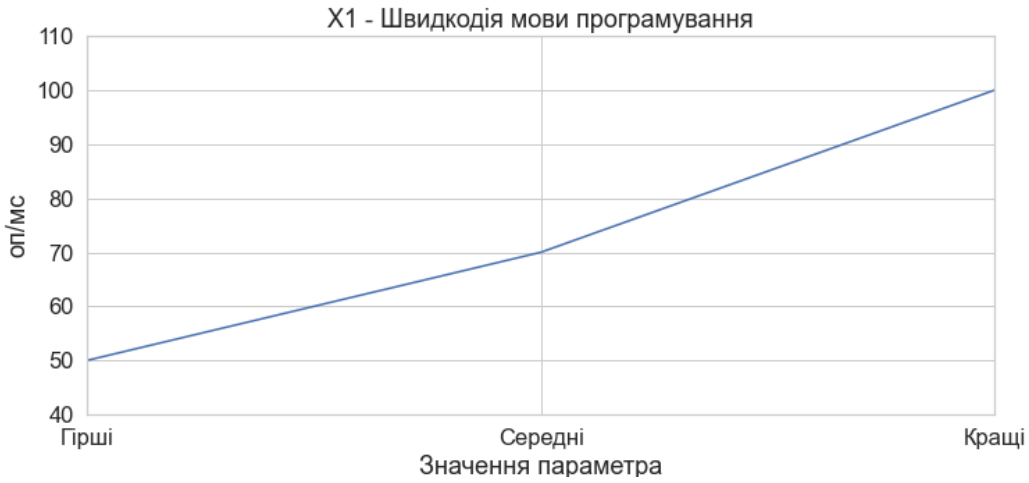


Рисунок 4.2 – X1, швидкодія мови програмування

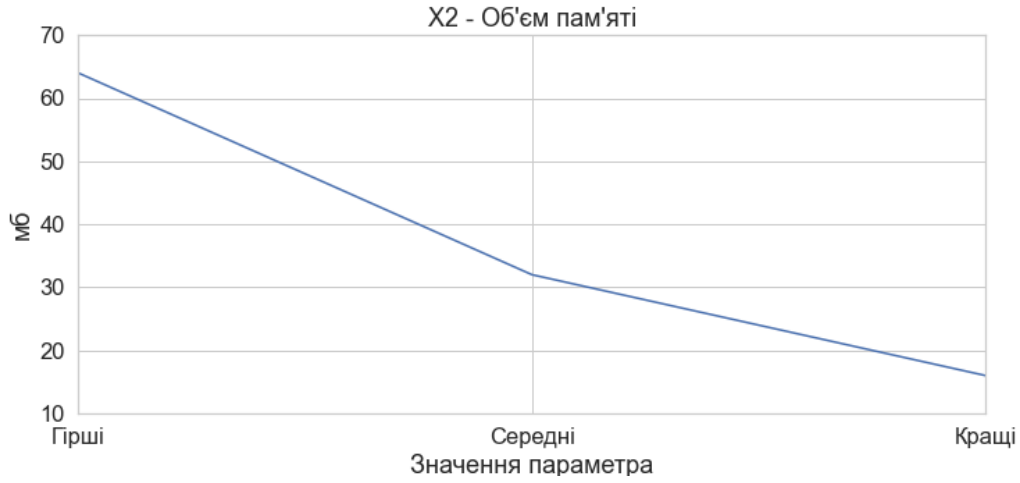


Рисунок 4.3 – X2, об’єм пам’яті

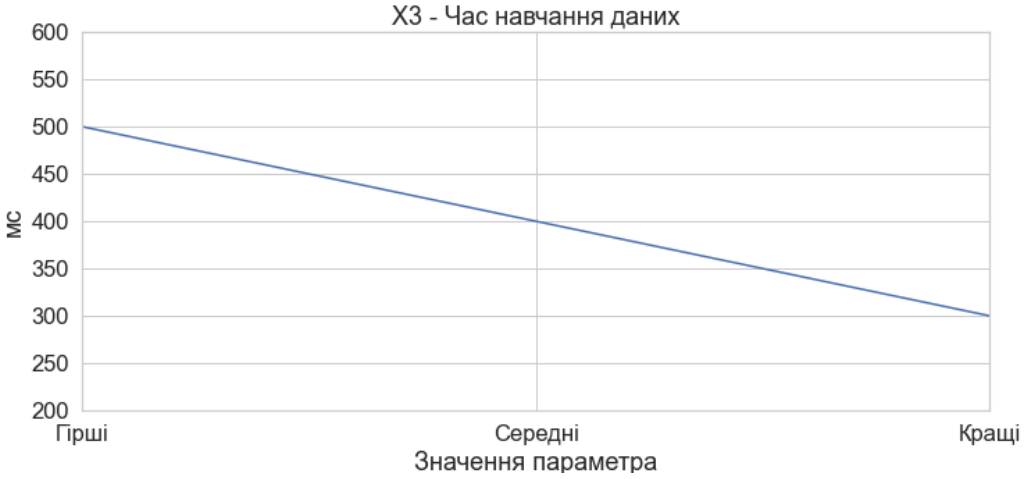


Рисунок 4.4 – X3, час навчання даних

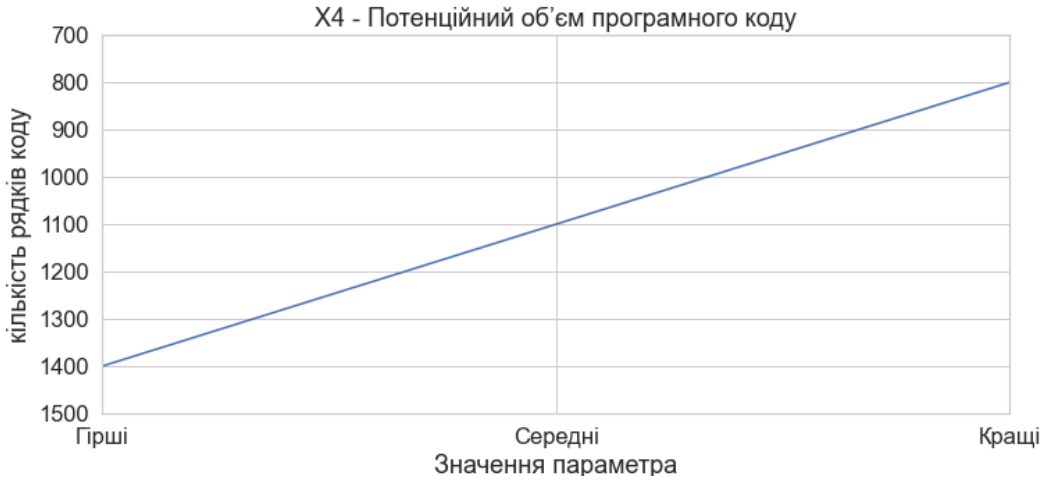


Рисунок 4.5 – X4, потенційний об'єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

− визначення рівня значимості параметра шляхом присвоєння різних рангів;

− перевірку придатності експертних оцінок для подальшого використання;

− визначення оцінки попарного пріоритету параметрів;

− обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	оп/мс	3	4	4	4	3	3	4	25	7.5	56.25
X2	Об’єм пам’яті, необхідний для обчислень та збереження даних	мб	2	2	1	2	1	1	2	11	-6.5	42.25
X3	Час навчання даних	мс	4	3	3	3	4	4	3	24	6.5	42.25
X4	Потенційний об’єм програмного коду	Кількість рядків коду	1	1	2	1	2	2	1	10	-7.5	56.25
	Разом		10	10	10	10	10	10	10	70	0	197

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \tag{4.1}$$

де N – число експертів; n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17.5 \tag{4.2}$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \tag{4.3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 197 \tag{4.4}$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 197}{7^2(4^3 - 4)} = 0.804 > W_k = 0.67. \tag{4.5}$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	>	>	>	>	>	>	>	>	1.5
X1 і X3	<	>	>	>	<	<	>	>	1.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	>	>	<	>	<	<	>	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається за формулою:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A=\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i} \quad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{ві}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \tag{4.9}$$

$$b_i' = \sum_{i=1}^{N} a_{ij} b_j \tag{4.10}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{ві}}$	b_i^1	$K_{\text{ві}}^1$	b_i^2	$K_{\text{ві}}^2$
X1	1	1.5	1.5	1.5	5.5	0.34	21.25	0.36	77.875	0.36
X2	0.5	1	0,5	1.5	3.5	0.22	12.25	0.21	44.875	0.21
X3	0.5	1.5	1	1.5	4.5	0.28	16.25	0.27	59.125	0.27
X4	0.5	0.5	0.5	1	2.5	0.16	9.25	0.16	34.125	0.16
Всього:					16	1	59	1	213	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X1 (Швидкодія мови програмування), X2 (Об’єм пам’яті, необхідний для обчислень та збереження даних) та X3 (Час навчання даних) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X4 (потенційний об’єм програмного коду) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (4.11)$$

де n – кількість параметрів; $K_{вi}$– коефіцієнт вагомості і–го параметра; B_i – оцінка і–го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	100	6	0.36	2.16
F2	А	X2	16	7	0.21	1.47
	Б	X3	300	5	0.27	1.35
F3	А	X4	1100	4	0.16	0.64

За даними з таблиці 4.6 за формулою:

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \qquad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 2.16 + 1.47 + 0.64 = 4.27;$$

$$K_{K2} = 2.16 + 1.35 + 0.64 = 4.15.$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$\mathrm{Т_О} = \mathrm{Т_Р} \cdot \mathrm{К_П} \cdot \mathrm{К_{СК}} \cdot \mathrm{К_М} \cdot \mathrm{К_{СТ}} \cdot \mathrm{К_{СТ.М}}, \qquad (4.13)$$

де $\mathrm{Т_Р}$ – трудомісткість розробки ПП; $\mathrm{К_П}$ – поправочний коефіцієнт; $\mathrm{К_{СК}}$ – коефіцієнт на складність вхідної інформації; $\mathrm{К_М}$ – коефіцієнт рівня мови програмування; $\mathrm{К_{СТ}}$ – коефіцієнт використання стандартних модулів і прикладних програм; $\mathrm{К_{СТ.М}}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P = 90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $К_П$ =0.9, $К_{СК}$ = 1, $К_{СТ}$ = 0.8:

$$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328.64 \text{ людино-годин.}$$

$$T_{II} = (122.4 + 19.44 + 6,91 + 19.44) \cdot 8 = 1345.52 \text{ людино-годин.}$$

У розробці бере участь один програміст з окладом 21200 грн та один аналітик даних з окладом 17800. Визначимо середню зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} \text{ грн.}, \quad (4.14)$$

де М – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$С_ч = \frac{21200 + 17800}{2 \cdot 21 \cdot 8} = 116.07 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, \quad (4.16)$$

де $С_ч$– величина погодинної оплати праці програміста; $Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 116.07 \cdot 1328.64 \cdot 1.2 = 185058.30$ грн.

II. $С_{ЗП} = 116.07 \cdot 1345.52 \cdot 1.2 = 187409.41$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 185058.30 \cdot 0.22 = 40712.83$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 187409.41 \cdot 0.22 = 41230.07$ грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 21200 грн., з коефіцієнтом зайнятості 0.2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 21200 \cdot 0.2 = 50880 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 50880 \cdot (1 + 0.2) = 61056 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 61056 \cdot 0.22 = 13432.32 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 35000 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 35000 = 10062.5 \text{ грн.},$$

де $К_{ТМ}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$ – річна норма амортизації; $Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 35000 \cdot 0.05 = 2012.5 \text{ грн.},$$

де $К_Р$ – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.8 =$$

$$= 1516.8 \text{ години,}$$

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t – кількість робочих годин в день; $К_В$ – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1516.8 \cdot 0.2 \cdot 0.8 \cdot 4.87 = 1181.89 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 35000 \cdot 0.67 = 23450 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н, \qquad (4.17)$$

$$С_{ЕКС} = 61056 + 13432.32 + 10062.5 + 2012.5 + 1181.89 + 23450 =$$
$$= 111195.21 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 111195.21 / 1516.8 = 73.31 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М-Г} \cdot Т, \quad (4.18)$$

I. $C_М = 73.31 \cdot 1328.64 = 97402.60$ грн.

II. $C_М = 73.31 \cdot 1345.52 = 98640.07$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0.67, \quad (4.19)$$

I. $C_Н = 185058.30 \cdot 0.67 = 123989.06$ грн.

II. $C_Н = 187409.41 \cdot 0.67 = 125564.31$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н, \quad (4.20)$$

I. $C_{ПП} = 185058.30 + 40712.83 + 97402.60 + 123989.06 = 447162.79$ грн.

II. $C_{ПП} = 187409.41 + 41230.07 + 98640.07 + 125564.31 = 452843.86$ грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР j} = К_{Кj} / C_{Фj}, \quad (4.21)$$

$$К_{ТЕР1} = 4.27 / 447162.79 = 9.549 \cdot 10^{-6},$$

$$К_{ТЕР2} = 4.15 / 452843.86 = 9.164 \cdot 10^{-6}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 9.549 \cdot 10^{-6}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розробляється, можна зробити висновок, що з альтернатив, які залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 9.549 \cdot 10^{-6}$.

Цей варіант реалізації програмного продукту має такі параметри:

– вибір мови програмування – Python;

– вибір операційної системи – MacOS;

– вибір середовища розробки – Jupyter Notebook.

Даний варіант виконання програмного комплексу дає користувачу зручну реалізацію програми, зрозумілий інтерфейс та широкий і доступний функціонал для роботи.

4.8 Висновки до розділу 4

В даній частині було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

Чоловічий теніс є одним з найпопулярніших спортивних видів діяльності у світі. Він привертає увагу великої кількості фанатів з різних куточків планети. Це сприяє розвитку і дотичних до тенісу галузей. Виникає необхідність прогнозувати результати матчів для переслідування різних цілей.

Це стає можливим із стрімким розвитком іншої популярної галузі – машинного навчання. Машинне навчання є передовим способом прогнозування, оскільки воно використовує алгоритми, які можуть автоматично виявляти складні залежності в даних і робити точні прогнози.

Для реалізації задуму моделювання результатів тенісних матчів за допомогою методів машинного навчання було обрано два алгоритми: логістичну регресію та різновид feed forward мережі багатошаровий персептрон (MLP). У даній роботі було представлено реалізацію цих двох методів та застосовано побудовані моделі для прогнозування переможців тенісних матчів одиночного розряду серед чоловіків.

Окремо моделі продемонстрували високу точність прогнозування: більше 95% кожна. Однак, побудовані моделі використовують статистичні дані матчів як вхідні змінні для прогнозування переможця, і тому при моделюванні результатів ще не зіграного тенісного матчу дослідник стикається із проблемою, що пов’язана із неможливістю отримати точні статистичні дані майбутньої гри. Цю проблему було вирішено шляхом реалізації алгоритму прогнозування статистики матчу на основі попередніх ігор гравців. Комбінація цього алгоритму з моделями логістичної регресії та мультишаровим персептроном показала результати у 67% та 66% правильно змодельованих результатів відповідно. Варто зазначити, що відповідна модель букмекерської компанії Favbet видає 72% правильних прогнозів, однак, дуже ймовірно, що букмекерська компанія володіє значно більшим обсягом інформації, ніж та, що була використана у цій роботі, а тому експеримент з

моделювання результатів тенісних матчів серед чоловіків можна вважати успішним.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. US Open IBM SlamTracker. URL: https://www.usopen.org/en_US/scores/index.html. (Last accessed: 19.04.2023).
2. IBM Transforms the 2015 US Open with Tennis Apps. URL: https://www.prnewswire.com/news-releases/ibm-transforms-the-2015-us-open-with-tennis-apps-analytics-and-acumen-300136871.html. (Last accessed: 19.04.2023).
3. Odds Tips. URL: https://www.oddstips.co.uk. (Last accessed: 19.04.2023).
4. toptennistips.com. URL: https://www.toptennistips.com. (Last accessed: 20.04.2023).
5. Association of Tennis Professionals. *Wikipedia, The Free Encyclopedia.* URL: https://en.wikipedia.org/wiki/Association_of_Tennis_Professionals (Last accessed: 23.04.2023).
6. Women's Tennis Association. *Wikipedia, The Free Encyclopedia.* URL: https://en.wikipedia.org/wiki/Women%27s_Tennis_Association (Last accessed: 23.04.2023).
7. International Tennis Federation. *Wikipedia, The Free Encyclopedia.* URL: https://en.wikipedia.org/wiki/International_Tennis_Federation. (Last accessed: 23.04.2023).
8. ATP Tour. *Wikipedia, The Free Encyclopedia.* URL: https://en.wikipedia.org/wiki/ATP_Tour (Last accessed: 24.04.2023).
9. WTA Tour. *Wikipedia, The Free Encyclopedia.* URL: https://en.wikipedia.org/wiki/WTA_Tour. (Last accessed: 24.04.2023).
10. Prize Money. URL: https://www.perfect-tennis.com/prize-money/. (Last accessed: 25.04.2023).
11. Prize Money 2023. URL: https://tenniscompanion.org/prize-money/. (Last accessed: 25.04.2023).

12. Теніс. *Вікіпедія – вільна енциклопедія.* URL: https://uk.wikipedia.org/wiki/Теніс. (дата звернення: 27.04.2023).
13. JeffSackmann. tennis_atp. URL: https://github.com/JeffSackmann/tennis_atp. (Last accessed: 27.05.2023).
14. Sipko M: Machine Learning for the Prediction of Professional Tennis Matches. URL: https://www.doc.ic.ac.uk/teaching/distinguished-projects/2015/m.sipko.pdf. (Last accessed: 04.05.2023).
15. What Is Logistic Regression?. URL: https://aws.amazon.com/what-is/logistic-regression/?nc1=h_ls. (Last accessed: 01.05.2023).
16. Reddy S. Understanding the log loss function. URL: https://medium.com/analytics-vidhya/understanding-the-loss-function-of-logistic-regression-ac1eec2838ce. (Last accessed: 01.05.2023).
17. Hale J. Tips for Better Logistic Regression Models in Scikit-Learn. URL: https://towardsdatascience.com/dont-sweat-the-solver-stuff-aea7cddc3451. (Last accessed: 02.05.2023).
18. A. Aylin Tokuç. Gradient Descent Equation in Logistic Regression. URL: https://www.baeldung.com/cs/gradient-descent-logistic-regression. (Last accessed: 02.05.2023).
19. Logistic Regression and regularization: Avoiding overfitting and improving generalization. URL: https://medium.com/@rithpansanga/logistic-regression-and-regularization-avoiding-overfitting-and-improving-generalization-e9afdcddd09d. (Last accessed: 02.05.2023).
20. Khushnuma Grover. Advantages and Disadvantages of Logistic Regression. URL: https://iq.opengenus.org/advantages-and-disadvantages-of-logistic-regression/. (Last accessed: 02.05.2023).
21. Chen J. What Is a Neural Network?. URL: https://www.investopedia.com/terms/n/neuralnetwork.asp. (Last accessed: 03.05.2023).
22. Dobilas S. Feed Forward Neural Networks — How To Successfully Build Them in Python. URL: https://towardsdatascience.com/feed-forward-neural-

networks-how-to-successfully-build-them-in-python-74503409d99a. (Last accessed: 03.05.2023).

23. Pranshu S. Basic Introduction to Feed-Forward Network in Deep Learning. URL: https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-feed-forward-network-in-deep-learning/. (Last accessed: 08.05.2023).
24. Techniques to prevent overfitting in Neural Networks. URL: https://www.kaggle.com/discussions/general/175912. (Last accessed: 10.05.2023).
25. Bento C. Multilayer Perceptron Explained with a Real-Life Example and Python Code: Sentiment Analysis. URL: https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141. (Last accessed: 11.05.2023).
26. What is Multilayer Perceptron?. URL: https://deepchecks.com/glossary/multilayer-perceptron/. (Last accessed: 10.05.2023).
27. Python (in Machine Learning). URL: https://corporatefinanceinstitute.com/resources/data-science/python-in-machine-learning/. (Last accessed: 11.05.2023).
28. Sejuti Das. Why Jupyter Notebooks Are So Popular Among Data Scientists. URL: https://analyticsindiamag.com/why-jupyter-notebooks-are-so-popular-among-data-scientists/. (Last accessed: 05.05.2023).
29. Flashscore. URL: https://www.flashscore.com. (Last accessed: 31.05.2023).

ДОДАТОК А
ЛІСТИНГ ПРОГРАМНОГО МОДУЛЮ

Лістинг 1 – Створення єдиного датасету та початок його обробки.

```
# In[1]:
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# In[2]:
df2000_atp = pd.read_csv('atp_2000.csv')
df2000_chall = pd.read_csv('atp_chall_2000.csv')
data = pd.concat([df2000_atp, df2000_chall], ignore_index =
True)
data.shape

# In[3]:
for i in range(2001, 2024):
    year = str(i)
    name1, name2 = f'atp_{year}.csv', f'atp_chall_{year}.csv'
    df_atp = pd.read_csv(f'atp_{year}.csv')
    df_chall = pd.read_csv(f'atp_chall_{year}.csv')
    data = pd.concat([data, df_atp, df_chall], ignore_index =
True)
    print(data.shape)

# In[4]:
data.head()

# In[5]:
data.tail()
```

```
# In[6]:
# find null values by columns
data.isnull().sum()

# In[7]:
# delete columns 'winner_seed', 'winner_entry', 'loser_seed',
'loser_entry' because of lots of missing values
data.drop(['winner_seed', 'winner_entry', 'loser_seed',
'loser_entry'], axis=1, inplace=True)
data.shape

# In[8]:
# Now we want to find out on which tourney level there are in
the dataset
data['tourney_level'].unique()

# A: ATP tourneys
#
# M: Masters
#
# G: Grand Slams
#
# F: Finals
#
# D: Davis Cup
#
# C: Challangers

# In[9]:
# we don`t want to predict G and D so we drop data from these
tourneys
to_drop = data[(data['tourney_level'] == 'D') |
(data['tourney_level'] == 'G')].index
to_drop
```

```
# In[10]:
data.drop(to_drop, inplace=True)
data.reset_index(drop=True, inplace=True)
data.shape

# In[11]:
# check
data['tourney_level'].unique()

# In[12]:
data.isnull().sum()

# In[13]:
# some of the final matches of bo3 torney were played in format
of bo5.
# We do not want this data to be in our dataset, so we will
delete it
bo5 = data[data['best_of'] == 5]
bo5

# In[14]:
to_drop = data[data['best_of'] == 5].index
to_drop

# In[15]:
data.drop(to_drop, inplace=True)
data.reset_index(drop=True, inplace=True)
data.shape

# In[16]:
# check
data['best_of'].unique()

# In[17]:
# df where w_ace is missing
```

```
df = data[data['w_ace'].isnull()]
df.tail()

# In[18]:
# as we can see there are no match stats and the score is shown
as W/O (withdrawn)
# We do not want such records to be in our dataframe: delete
them
to_drop = data[data['score'].str.contains('W/O',
na=False)].index
to_drop

# In[19]:
data.drop(to_drop, inplace=True)
data.reset_index(drop=True, inplace=True)
data.shape

# In[20]:
# also some records contnain RET in score column. RET is for
retired.
# We don`t want such records in our data: delete them
to_drop = data[data['score'].str.contains('RET',
na=False)].index
to_drop

# In[21]:
data.drop(to_drop, inplace=True)
data.reset_index(drop=True, inplace=True)
data.shape

# In[22]:
# also some records contnain DEF or Def. in score column.
# We don`t want such records in our data: delete them
to_drop = data[data['score'].str.contains('DEF',
na=False)].index
data.drop(to_drop, inplace=True)
```

```
to_drop = data[data['score'].str.contains('Def.',
na=False)].index

# In[23]:
data.drop(to_drop, inplace=True)
data.reset_index(drop=True, inplace=True)
data.shape

# In[24]:
data.isnull().sum()

# In[25]:
# we are not interested in matches where there are no match
stats, so we will delete records without
# stats
to_drop = data[data['w_ace'].isnull()].index
data.drop(to_drop, inplace=True)
data.reset_index(drop=True, inplace=True)
data.shape

# In[26]:
data.isnull().sum()

# In[27]:
# we do not need these columns to predict outcomes or deal with
data: tourney_id, match_num,
# winner_id, winner_ioc, loser_id, loser_ioc, best_of, minutes.
Drop them
data.drop(['tourney_id', 'match_num', 'winner_id', 'winner_ioc',
'loser_id',
           'loser_ioc', 'best_of', 'minutes'], axis=1,
inplace=True)
data.shape

# In[28]:
data.isnull().sum()
```

```
# In[29]:
data.head()

# In[30]:
data.dtypes

# In[31]:
# let`s change tourney_date type to datetime
data = data.astype({'tourney_date': 'string'})
data['tourney_date'] = pd.to_datetime(data['tourney_date'])
data.dtypes

# In[32]:
data.head()

# In[33]:
# sort data by tourney_date and name
data.sort_values(by=['tourney_date', 'tourney_name'],
inplace=True)
data.reset_index(drop=True, inplace=True)
data.tail()

# In[34]:
data.shape

# In[35]:
# save this df for future stats prediction
data.to_csv('stats_pred.csv', index=False)
```

Лістинг 2 – Обробка даних

```
# In[1]:

import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# In[2]:
data = pd.read_csv('stats_pred.csv')
data.shape

# In[3]:
data.head()

# In[4]:
data.isnull().sum()

# In[5]:
# витягнемо кількість геймів з рахунку ('score')
scores_list = data['score'].values.tolist()
scores_list[:6]

# In[6]:
import re

# In[7]:
scores_clear = []
for score in scores_list:

    curr_scr = re.sub('\(\d*\)', '', score) # delete scores in
brackets ()
```

```
    curr_scr = curr_scr.rstrip() # delete space after the last
number if exists

    # do not take into consideration tiebreaks
    curr_scr = curr_scr.split()
    for i in range(len(curr_scr)):
        if '7-6' in curr_scr[i]:
            curr_scr[i] = curr_scr[i].replace('7-', '6-')
        elif '6-7' in curr_scr[i]:
            curr_scr[i] = curr_scr[i].replace('-7', '-6')

    new_scr = ' '.join(curr_scr)

    new_scr = re.sub(r'\D', ' ', new_scr) # delete all symbols
exept numeric

    scores_clear.append(new_scr)
scores_clear[:6]

# In[8]:
winner_games, loser_games = [], []
# count total games for each player
# scores_clear - list of clear scores ['7 6 6 3 6 2', '7 6 6 0 6
2', '3 6 7 6 6 2 4 1']
# splitted list of scores ['7', '6', '6', '2', '6', '3']
for score in scores_clear:
    win_g, los_g = 0, 0
    splitted = score.split(' ')
    for el in range(len(splitted)):
        if (el+1)%2 == 1:
            win_g += int(splitted[el])
        elif (el+1)%2 == 0:
            los_g += int(splitted[el])
    winner_games.append(win_g)
    loser_games.append(los_g)
```

```
# In[9]:
# adding cols of total games by each player and total
data['w_games'] = winner_games
data['l_games'] = loser_games
data['total_games'] = data['w_games'] + data['l_games']
data.tail()

# In[10]:
data.isnull().sum()

# In[11]:
import numpy as np
# all values measured between 0 and 1 (percentage)
# s for serve; r for receive
#
#
#
# p1_1st_in  - player1 1st serve in
#
# p1_1st_ptwon - player1 1st serve points won
#
# p1_2nd_ptwon - player1 2nd serve points won
#
# p1_bp_saved - player1 break points saved, 1 if there were no
bp faced
#
# p1_ace_per_s - player1 aces per serve
#
# p1_df_per_s - player1 double faults per serve
#
# p1_1st_r_ptwon - player1 1st serve points won on receive
#
# p1_2nd_r_ptwon - player1 2nd serve points won on receive
#
# p1_bp_won - player1 break points won
```

```
#
# p1_tot_pt_won_s - player1 total points won on serve
#
# p1_tot_pt_won_r - player1 total points won on receive
#
# p1_tot_pt_won - player1 total points won
#
# p1_gms_won_s - player1 games won on serve
#
# p1_gms_won_r - player1 games won on receive
#
# p1_tot_gms_won - player1 total games won

# In[12]:
data['p1_1st_in'] = round(data['w_1stIn'] / data['w_svpt'], 2)
data['p1_1st_ptwon'] = round(data['w_1stWon'] / data['w_1stIn'],
2)
data['p1_2nd_ptwon'] = round(data['w_2ndWon'] / (data['w_svpt']
- data['w_1stIn']), 2)
data['p1_bp_saved'] = np.where(data['w_bpFaced'] == 0, 1.0,
round(data['w_bpSaved'] / data['w_bpFaced'], 2))
data['p1_ace_per_s'] = round(data['w_ace'] / data['w_svpt'], 2)
data['p1_df_per_s'] = round(data['w_df'] / data['w_svpt'], 2)
data['p1_1st_r_ptwon'] = round((data['l_1stIn'] -
data['l_1stWon']) / data['l_1stIn'], 2)
data['p1_2nd_r_ptwon'] = round((data['l_svpt'] - data['l_1stIn']
- data['l_2ndWon']) / (data['l_svpt'] - data['l_1stIn']), 2)
data['p1_bp_won'] = np.where(data['l_bpFaced'] == 0, 0.0,
round((data['l_bpFaced'] - data['l_bpSaved']) /
data['l_bpFaced'], 2))
data['p1_tot_pt_won_s'] = round((data['w_1stWon'] +
data['w_2ndWon']) / data['w_svpt'], 2)
data['p1_tot_pt_won_r'] = round((data['l_svpt'] -
data['l_1stWon'] - data['l_2ndWon']) / data['l_svpt'], 2)
```

```
data['p1_tot_pt_won'] = round((data['w_1stWon'] +
data['w_2ndWon'] + (data['l_svpt'] - data['l_1stWon'] -
data['l_2ndWon'])) / (data['w_svpt'] + data['l_svpt']), 2)
data['p1_gms_won_s'] = round((data['w_SvGms'] -
(data['w_bpFaced'] - data['w_bpSaved'])) / data['w_SvGms'], 2)
data['p1_gms_won_r'] = round((data['l_bpFaced'] -
data['l_bpSaved']) /  data['l_SvGms'], 2)
data['p1_tot_gms_won'] = round(data['w_games'] /
data['total_games'], 2)

data['p2_1st_in'] = round(data['l_1stIn'] / data['l_svpt'], 2)
data['p2_1st_ptwon'] = round(data['l_1stWon'] / data['l_1stIn'],
2)
data['p2_2nd_ptwon'] = round(data['l_2ndWon'] / (data['l_svpt']
- data['l_1stIn']), 2)
data['p2_bp_saved'] = np.where(data['l_bpFaced'] == 0, 1.0,
round(data['l_bpSaved'] / data['l_bpFaced'], 2))
data['p2_ace_per_s'] = round(data['l_ace'] / data['l_svpt'], 2)
data['p2_df_per_s'] = round(data['l_df'] / data['l_svpt'], 2)
data['p2_1st_r_ptwon'] = round((data['w_1stIn'] -
data['w_1stWon']) / data['w_1stIn'], 2)
data['p2_2nd_r_ptwon'] = round((data['w_svpt'] - data['w_1stIn']
- data['w_2ndWon']) / (data['w_svpt'] - data['w_1stIn']), 2)
data['p2_bp_won'] = np.where(data['w_bpFaced'] == 0, 0.0,
round((data['w_bpFaced'] - data['w_bpSaved']) /
data['w_bpFaced'], 2))
data['p2_tot_pt_won_s'] = round((data['l_1stWon'] +
data['l_2ndWon']) / data['l_svpt'], 2)
data['p2_tot_pt_won_r'] = round((data['w_svpt'] -
data['w_1stWon'] - data['w_2ndWon']) / data['w_svpt'], 2)
data['p2_tot_pt_won'] = round((data['l_1stWon'] +
data['l_2ndWon'] + (data['w_svpt'] - data['w_1stWon'] -
data['w_2ndWon'])) / (data['l_svpt'] + data['w_svpt']), 2)
data['p2_gms_won_s'] = round((data['l_SvGms'] -
(data['l_bpFaced'] - data['l_bpSaved'])) / data['l_SvGms'], 2)
```

```
data['p2_gms_won_r'] = round((data['w_bpFaced'] -
data['w_bpSaved']) /  data['w_SvGms'], 2)
data['p2_tot_gms_won'] = round(data['l_games'] /
data['total_games'], 2)
data.tail()

# In[13]:
data.isnull().sum()

# In[14]:
# p1_2nd_ptwon missing
df = data[data['p1_2nd_ptwon'].isnull()]
df

# In[15]:
# that is because those players(winners) did not win any point
on second service
# lets replace those nan values by 0
indexx = data[data['p1_2nd_ptwon'].isnull()].index
data.loc[indexx, 'p1_2nd_ptwon'] = 0
data['p1_2nd_ptwon'].isnull().sum()

# In[16]:
# p1_1st_r_ptwon missing
df = data[data['p1_1st_r_ptwon'].isnull()]
df

# In[17]:
# that's because there is no data about loser's performance
# lets delete those  rows
data.drop(df.index, inplace=True)
data['p1_1st_r_ptwon'].isnull().sum()

# In[18]:
# p1_2nd_r_ptwon missing
df = data[data['p1_2nd_r_ptwon'].isnull()]
```

```
df

# In[19]:
# that's because all losers' balls were served with first
service, so we get like 0/0 in our formula
# let's replace nan by 0
indexx = data[data['p1_2nd_r_ptwon'].isnull()].index
data.loc[indexx, 'p1_2nd_r_ptwon'] = 0
data['p1_2nd_r_ptwon'].isnull().sum()

# In[20]:
# p2_2nd_r_ptwon missing
df = data[data['p2_2nd_r_ptwon'].isnull()]
df

# In[21]:
# p2_2nd_r_ptwon missing the same as p1_2nd_r_ptwon missing
indexx = data[data['p2_2nd_r_ptwon'].isnull()].index
data.loc[indexx, 'p2_2nd_r_ptwon'] = 0
data['p2_2nd_r_ptwon'].isnull().sum()

# In[22]:
# p2_2nd_ptwon missing
df = data[data['p2_2nd_ptwon'].isnull()]
df

# In[23]:
# p2_2nd_ptwon missing the same as p1_2nd_ptwon missing
indexx = data[data['p2_2nd_ptwon'].isnull()].index
data.loc[indexx, 'p2_2nd_ptwon'] = 0
data['p2_2nd_ptwon'].isnull().sum()

# In[24]:
data.isnull().sum()

# In[25]:
```

```
# p1_gms_won_r missing
df = data[data['p1_gms_won_r'].isnull()]
df

# In[26]:
# p2_gms_won_s missing
df = data[data['p2_gms_won_s'].isnull()]
df

# In[27]:
# in p1_gms_won_r and p2_gms_won_s missing is the same play.
# As we see, the data is incorrect, so we will delete this row
data.drop([34327], inplace=True)
data.reset_index(drop=True, inplace=True)
data.tail()

# In[28]:
data.isnull().sum()

# In[29]:
data.dtypes

# In[30]:
df = data.select_dtypes((int, float))
df.dtypes

# In[31]:
# check inf
count = np.isinf(df).values.sum()
count

# In[32]:
r = df.index[np.isinf(df).any(1)]
r

# In[33]:
```

```
data.drop(r, inplace=True)
data.reset_index(drop=True, inplace=True)

# In[34]:
df = data.select_dtypes((int, float))
np.isinf(df).values.sum()

# In[35]:
data.columns

# In[36]:
# delete columns that we are no longer interested in or that are
irrelevant
data.drop(['w_ace', 'w_df', 'w_svpt', 'w_1stIn', 'w_1stWon',
'w_2ndWon',
           'w_SvGms', 'w_bpSaved', 'w_bpFaced', 'l_ace', 'l_df',
'l_svpt',
           'l_1stIn', 'l_1stWon', 'l_2ndWon', 'l_SvGms',
'l_bpSaved', 'l_bpFaced',
           'score', 'w_games', 'l_games', 'total_games'],
axis=1, inplace=True)
data.tail()

# In[37]:
# rename columns
data.rename(columns = {'winner_name': 'p1_name', 'winner_hand':
'p1_hand',
                       'winner_ht': 'p1_ht', 'winner_age':
'p1_age', 'loser_id': 'p2_id',
                       'loser_name': 'p2_name', 'loser_hand':
'p2_hand', 'loser_ht': 'p2_ht',
                       'loser_age': 'p2_age', 'winner_rank':
'p1_rank',
                       'winner_rank_points': 'p1_rank_points',
'loser_rank': 'p2_rank',
```

```
                        'loser_rank_points': 'p2_rank_points'},
inplace=True)
data.tail()
# remember that p1 is winner and p2 is loser

# In[38]:
data.to_csv('data_pred_not_ready.csv', index=False)

# In[39]:
# now we delete all records where there are missing values
data.dropna(inplace=True)
data.reset_index(drop=True, inplace=True)
data.shape

# In[40]:
data.isnull().sum().sum()

# In[41]:
data['p1_hand'].unique()

# In[42]:
data['p2_hand'].unique()

# In[43]:
# now we delete records where hand is U (unknown)
to_drop = data[(data['p1_hand'] == 'U') | (data['p2_hand'] ==
'U')].index
to_drop

# In[44]:
data.drop(to_drop, inplace=True)
data.reset_index(drop=True, inplace=True)
data.shape

# In[45]:
# check
```

```
data['p1_hand'].unique(), data['p2_hand'].unique()

# In[46]:
data.to_csv('prepared_data.csv', index=False)
```

Лістинг 3 – Трансформація даних

```
# In[1]:
import pandas as pd

import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

# In[2]:
data = pd.read_csv('prepared_data.csv')
data.tail()

# remember: player1 is a winner

# In[3]:
data.shape

# In[4]:
# delete irrelevant columns for logreg

data.drop(['tourney_name', 'tourney_date', 'p1_name',
'p2_name'], axis=1, inplace=True)
data.tail()

# In[5]:
```

```
data.shape

# In[6]:
data0 = data.copy()

# In[7]:
# swipe columns for losers
p1_lst = ['p1_hand', 'p1_ht', 'p1_age', 'p1_rank',
'p1_rank_points', 'p1_1st_in', 'p1_1st_ptwon',
          'p1_2nd_ptwon', 'p1_bp_saved', 'p1_ace_per_s',
'p1_df_per_s', 'p1_1st_r_ptwon',
          'p1_2nd_r_ptwon', 'p1_bp_won', 'p1_tot_pt_won_s',
'p1_tot_pt_won_r', 'p1_tot_pt_won',
          'p1_gms_won_s', 'p1_gms_won_r', 'p1_tot_gms_won']
p2_lst = ['p2_hand', 'p2_ht', 'p2_age', 'p2_rank',
'p2_rank_points', 'p2_1st_in', 'p2_1st_ptwon',
          'p2_2nd_ptwon', 'p2_bp_saved', 'p2_ace_per_s',
'p2_df_per_s', 'p2_1st_r_ptwon',
          'p2_2nd_r_ptwon', 'p2_bp_won', 'p2_tot_pt_won_s',
'p2_tot_pt_won_r', 'p2_tot_pt_won',
          'p2_gms_won_s', 'p2_gms_won_r', 'p2_tot_gms_won']

# In[8]:
for i, j in zip(p1_lst, p2_lst):
    data0.loc[:,[i, j]] = data0.loc[:,[j, i]].values

data0.head()

# In[9]:
# adding target variable
data['result'] = 1
data0['result'] = 0

# In[10]:
data = pd.concat([data, data0], axis=0)
data.head()
```

```
# In[11]:
data.shape

# In[12]:
# shuffle dataframe
data = data.sample(frac=1)
data.reset_index(drop=True, inplace=True)
data.head()

# In[13]:
data.shape

# In[14]:
# one hot encoding
colmns = ['surface', 'draw_size', 'tourney_level', 'p1_hand',
'p2_hand', 'round']
colmns

# In[15]:
data = pd.get_dummies(data, columns = colmns)
data.head()

# In[16]:
# if player plays with right hand, indicates like 1, 0 else(with
left)
data.drop(['p1_hand_L', 'p2_hand_L'], axis=1, inplace=True)
data.head()

# In[17]:
# calculate difference between players
p1_lst = ['p1_ht', 'p1_age', 'p1_rank', 'p1_rank_points',
'p1_1st_in', 'p1_1st_ptwon',
          'p1_2nd_ptwon', 'p1_bp_saved', 'p1_ace_per_s',
'p1_df_per_s', 'p1_1st_r_ptwon',
```

```
          'p1_2nd_r_ptwon', 'p1_bp_won', 'p1_tot_pt_won_s',
'p1_tot_pt_won_r', 'p1_tot_pt_won',
          'p1_gms_won_s', 'p1_gms_won_r', 'p1_tot_gms_won']

p2_lst = ['p2_ht', 'p2_age', 'p2_rank', 'p2_rank_points',
'p2_1st_in', 'p2_1st_ptwon',
          'p2_2nd_ptwon', 'p2_bp_saved', 'p2_ace_per_s',
'p2_df_per_s', 'p2_1st_r_ptwon',
          'p2_2nd_r_ptwon', 'p2_bp_won', 'p2_tot_pt_won_s',
'p2_tot_pt_won_r', 'p2_tot_pt_won',
          'p2_gms_won_s', 'p2_gms_won_r', 'p2_tot_gms_won']

new_names = ['ht_diff', 'age_diff', 'rank_diff',
'rank_pts_diff', '1st_in_diff', '1st_ptwon_diff',
          '2nd_ptwon_diff', 'bp_saved_diff', 'ace_per_s_diff',
'df_per_s_diff', '1st_r_ptwon_diff',
          '2nd_r_ptwon_diff', 'bp_won_diff',
'tot_pt_won_s_diff', 'tot_pt_won_r_diff', 'tot_pt_won_diff',
          'gms_won_s_diff', 'gms_won_r_diff',
'tot_gms_won_diff']

# In[18]:
for i, j, n in zip(p1_lst, p2_lst, new_names):
    data[n] = round(data[i] - data[j], 2)
    data.drop([i, j], axis=1, inplace=True)

data.tail()

# In[19]:
data.shape

# In[20]:
data.to_csv('ready_for_train.csv', index=False)
```

Лістинг 4 – Побудова моделі логістичної регресії

```
# In[1]:
import pandas as pd
pd.set_option('display.max_columns', None)

import numpy as np

import warnings
warnings.filterwarnings('ignore')

# In[2]:
data = pd.read_csv('ready_for_train.csv')
data.head()

# In[3]:
data.shape

# In[4]:
data.isnull().sum().sum()

# In[5]:
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split,
GridSearchCV, cross_val_score
from sklearn.metrics import accuracy_score, f1_score,
roc_auc_score, log_loss, precision_score, recall_score

# In[6]:
Y = data['result']
X = data.drop(['result'], axis=1)
Y.value_counts()

# In[7]:
```

```
X.columns

# In[8]:
# Обираємо метод оптимізації і розбиття тестової та тренувальної
вибірок
size = [round(i, 2) for i in np.arange(0.1, 0.25, 0.05)]
minimization_metod = ['lbfgs', 'liblinear', 'newton-cg', 'sag',
'saga']

for k in size:
    X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size = k, random_state=42)
    print(f'\ntest_size = {k}:\n')
    for i in minimization_metod:
        logreg = LogisticRegression(solver=i)
        logreg.fit(X_train, y_train)
        y_pred = logreg.predict(X_test)
        print(f'solver = {i}, accuracy =
{round(accuracy_score(y_test, y_pred), 5)}')


# Як бачимо, для тестової вибірки будь-якого розміру найкращим
алгоритмом мінімізації завжди є newton-cg

# In[9]:
# GridSearchCV
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size = 0.1, random_state=42)

logreg = LogisticRegression(solver='newton-cg')

grid = GridSearchCV(logreg, param_grid = {'fit_intercept':
[True, False], \
                                          'C': [i for i in
range(0, 11)], \
```

```
                                        'penalty': ['l1',
'l2', 'elasticnet', None]})

grid.fit(X_train, y_train)
best = grid.best_params_
best

# In[10]:
y_pred = grid.predict(X_test)
round(accuracy_score(y_test, y_pred), 5)

# In[11]:
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size = 0.1, random_state=42)

logreg = LogisticRegression(solver='newton-cg', C=10,
penalty='l2')

logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)
round(accuracy_score(y_test, y_pred), 5)

# In[50]:
# check overfitting
best_inter, best_C, acc, r_auc, f1, prec, rec, logl = ([] for i
in range(8))

scoring = ['accuracy', 'roc_auc', 'f1', 'precision', 'recall',
'neg_log_loss']
metrics_l = [acc, r_auc, f1, prec, rec, logl]

for i in zip(metrics_l, scoring):
    el = cross_val_score(logreg, X_test, y_test, cv = 5, scoring
= i[1])
```

```
    if i[1] == 'neg_log_loss':
        el = -1 * el

    i[0].append(el.mean())

for k in zip(metrics_l, scoring):
    el = cross_val_score(logreg, X_train, y_train, cv = 5,
scoring = k[1])

    if k[1] == 'neg_log_loss':
        el = -1 * el

    k[0].append(el.mean())

dictt = {
         'Accuracy': acc,
         'Precision': prec,
         'Recall': rec,
         'F1 score': f1,
         'Roc auc': r_auc,
         'Log loss': logl
        }

result = pd.DataFrame(dictt, index = ['test', 'train'])

result.round(5)

# In[13]:
# перенавчання немає

# In[14]:
# метрики на прогнозованих значеннях
print(f'accuracy: {round(accuracy_score(y_test, y_pred), 5)}')
print(f'precision: {round(precision_score(y_test, y_pred), 5)}')
print(f'recall: {round(recall_score(y_test, y_pred), 5)}')
print(f'f1: {round(f1_score(y_test, y_pred), 5)}')
```

```
print(f'roc auc: {round(roc_auc_score(y_test, y_pred), 5)}')
print(f'log loss: {round(log_loss(y_test, y_pred), 5)}')

# In[15]:
# confusion matrix
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import cross_val_predict
from mlxtend.plotting import plot_confusion_matrix
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')

import seaborn as sns
sns.set(rc = {'figure.figsize':(15,9)})
sns.set(font_scale = 1.15)

# In[16]:
conf_m = confusion_matrix(y_test, y_pred)
conf_m

# In[17]:
fig, ax = plot_confusion_matrix(conf_mat = conf_m, show_normed =
True, colorbar = True)
plt.title('Confusion matrix')
plt.show()

# In[18]:
from sklearn.metrics import classification_report, roc_curve

# In[19]:
print(classification_report(y_test, y_pred))

# In[20]:
y_pred_prob = logreg.predict_proba(X_test)[:,1]
y_pred_prob.round(2)

# In[21]:
```

```
sns.histplot(data = y_pred_prob, element="poly")
plt.xlim(0,1)
plt.xlabel('Predicted probabilities of winning by player 1')
plt.show()

# In[22]:
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

# In[23]:
auc = roc_auc_score(y_test, y_pred_prob)
auc.round(5)

# In[24]:
plt.plot(fpr, tpr, label = "AUC = " + str(auc.round(5)), color =
'blue')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.plot([0,1], [0,1], 'k--', color = 'blue', label = "AUC = " +
str(0.5))
plt.legend(loc = 4)
plt.show()

# ### Рівняння регресійної моделі

# In[25]:
columns = X.columns.to_numpy()
columns

# In[26]:
coeffs = logreg.coef_.reshape((-1, 1))
coeffs

# In[27]:
s = '(' + str(logreg.intercept_[0].round(5)) + ')'
for i in range(len(columns)):
```

```
    s += ' + (' + str(coeffs[i][0].round(5)) + ' * ' +
columns[i] + ')'
print('y^ = ', s)

# In[28]:
from joblib import dump

# In[29]:
dump(logreg, 'logreg.joblib')
```

Лістинг 5 – Побудова мультишарового персептрону

```
# In[1]:
import pandas as pd
pd.set_option('display.max_columns', None)

import numpy as np

import warnings
warnings.filterwarnings('ignore')

# In[2]:
data = pd.read_csv('ready_for_train.csv')
data.head()

# In[3]:
data.drop(['rank_diff', 'rank_pts_diff'], axis=1, inplace=True)

# In[4]:
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV,
train_test_split
from keras.models import Sequential
```

```
from keras.layers import Dense
from keras.metrics import AUC
from keras import backend as K
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam

# In[5]:
Y = data['result']
X = data.drop(['result'], axis=1)
Y.value_counts()

# In[6]:
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.1, random_state=42)

# In[7]:
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0,
1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0,
1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives +
K.epsilon())
    return precision

def f1score_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

```
# In[8]:
def create_model(neurons1, neurons2, neurons3, activation1,
activation2, activation3, optimizer):
    MLP = Sequential()
    MLP.add(Dense(neurons1, activation=activation1,
input_dim=X_train.shape[1]))
    MLP.add(Dense(neurons2, activation=activation2))
    MLP.add(Dense(neurons3, activation=activation3))
    MLP.add(Dense(1, activation='sigmoid'))
    MLP.compile(optimizer=optimizer, loss='binary_crossentropy',
                metrics=['accuracy', precision_m, recall_m,
f1score_m, AUC()])
    return MLP

# In[9]:
MLP = KerasClassifier(build_fn=create_model, verbose=0)

# In[10]:
param_grid = {
    'neurons1': [32, 64, 128],
    'neurons2': [32, 64, 128],
    'neurons3': [32, 64, 128],
    'activation1': ['relu', 'tanh'],
    'activation2': ['relu', 'tanh'],
    'activation3': ['relu', 'tanh'],
    'optimizer': ['adam', 'sgd', 'rmsprop']
}

# In[11]:
grid = GridSearchCV(estimator=MLP, param_grid=param_grid)

grid.fit(X_train, y_train)
best = grid.best_params_
best

# In[12]:
```

```
for i in [0.1, 0.01, 0.001, 0.0001]:

    opt = Adam(learning_rate=i)

    MLP = Sequential()
    MLP.add(Dense(64, activation='tanh',
input_dim=X_train.shape[1]))
    MLP.add(Dense(32, activation='relu'))
    MLP.add(Dense(32, activation='tanh'))
    MLP.add(Dense(1, activation='sigmoid'))
    MLP.compile(optimizer=opt, loss='binary_crossentropy',
                metrics=['accuracy', precision_m, recall_m,
f1score_m, AUC()])

    early_stopping = EarlyStopping(monitor='val_loss',
patience=3)
    MLP.fit(X_train, y_train, epochs=20, batch_size=None,
        validation_data=(X_test, y_test),
callbacks=[early_stopping])

    metricss = MLP.evaluate(X_test, y_test)
    print(f'learning_rate: {i}\n')
    print(f'loss:          {metricss[0]}\n')
    print(f'accuracy:      {metricss[1]}\n')
    print(f'precision:     {metricss[2]}\n')
    print(f'recall:        {metricss[3]}\n')
    print(f'f1-score:      {metricss[4]}\n')
    print(f'roc_auc:       {metricss[5]}\n\n\n')

# In[13]:
# learning_rate=0.0001 найкращий варіант

# In[14]:
MLP = Sequential()
MLP.add(Dense(64, activation='tanh',
input_dim=X_train.shape[1]))
```

```
MLP.add(Dense(32, activation='relu'))
MLP.add(Dense(32, activation='tanh'))
MLP.add(Dense(1, activation='sigmoid'))

MLP.compile(optimizer=Adam(learning_rate=0.0001),
loss='binary_crossentropy',
            metrics=['accuracy', precision_m, recall_m,
f1score_m, AUC()])

early_stopping = EarlyStopping(monitor='val_loss', patience=3)

MLP.fit(X_train, y_train, epochs=30, batch_size=None,
    validation_data=(X_test, y_test),
callbacks=[early_stopping])

metrics_test = MLP.evaluate(X_test, y_test)
metrics_train = MLP.evaluate(X_train, y_train)

dictt = {'accuracy': [metrics_test[1], metrics_train[1]],
         'precision': [metrics_test[2], metrics_train[2]],
         'recall': [metrics_test[3], metrics_train[3]],
         'f1-score': [metrics_test[4], metrics_train[4]],
         'roc_auc': [metrics_test[5], metrics_train[5]],
         'loss': [metrics_test[0], metrics_train[0]]}

result = pd.DataFrame(dictt, index = ['test', 'train'])
result

# In[15]:
MLP.save('MLP.h5')
```

Лістинг 6 – Підготовка даних для прогнозування статистики

```
# In[1]:
import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import warnings
warnings.filterwarnings('ignore')

# In[2]:
data = pd.read_csv('data_pred_not_ready.csv')
data.shape

# In[3]:
data.head()

# In[4]:
data.isnull().sum()

# In[5]:
df_p1 = data[['tourney_name', 'surface', 'draw_size',
'tourney_level', 'tourney_date', 'p1_name',
              'round', 'p1_1st_in', 'p1_1st_ptwon',
'p1_2nd_ptwon', 'p1_bp_saved', 'p1_ace_per_s',
              'p1_df_per_s', 'p1_1st_r_ptwon', 'p1_2nd_r_ptwon',
'p1_bp_won', 'p1_tot_pt_won_s',
              'p1_tot_pt_won_r', 'p1_tot_pt_won',
'p1_gms_won_s', 'p1_gms_won_r', 'p1_tot_gms_won',
              'p2_name']]
df_p1.rename(columns = {'p1_name': 'player_name', 'p1_1st_in':
'1st_in', 'p1_1st_ptwon': '1st_ptwon',
                        'p1_2nd_ptwon': '2nd_ptwon',
'p1_bp_saved': 'bp_saved',
```

```
                        'p1_ace_per_s': 'ace_per_s',
'p1_df_per_s': 'df_per_s',
                        'p1_1st_r_ptwon': '1st_r_ptwon',
'p1_2nd_r_ptwon': '2nd_r_ptwon',
                        'p1_bp_won': 'bp_won',
'p1_tot_pt_won_s': 'tot_pt_won_s',
                        'p1_tot_pt_won_r': 'tot_pt_won_r',
'p1_tot_pt_won': 'tot_pt_won',
                        'p1_gms_won_s': 'gms_won_s',
'p1_gms_won_r': 'gms_won_r',
                        'p1_tot_gms_won': 'tot_gms_won',
'p2_name': 'opponent_name'}, inplace=True)
df_p1['result'] = 1
df_p1.head()

# In[6]:
df_p1.shape

# In[7]:
df_p2 = data[['tourney_name', 'surface', 'draw_size',
'tourney_level', 'tourney_date', 'p2_name',
              'round', 'p2_1st_in', 'p2_1st_ptwon',
'p2_2nd_ptwon', 'p2_bp_saved', 'p2_ace_per_s',
              'p2_df_per_s', 'p2_1st_r_ptwon', 'p2_2nd_r_ptwon',
'p2_bp_won', 'p2_tot_pt_won_s',
              'p2_tot_pt_won_r', 'p2_tot_pt_won',
'p2_gms_won_s', 'p2_gms_won_r', 'p2_tot_gms_won',
              'p1_name']]
df_p2.rename(columns = {'p2_name': 'player_name', 'p2_1st_in':
'1st_in', 'p2_1st_ptwon': '1st_ptwon',
                        'p2_2nd_ptwon': '2nd_ptwon',
'p2_bp_saved': 'bp_saved',
                        'p2_ace_per_s': 'ace_per_s',
'p2_df_per_s': 'df_per_s',
                        'p2_1st_r_ptwon': '1st_r_ptwon',
'p2_2nd_r_ptwon': '2nd_r_ptwon',
```

```
                        'p2_bp_won': 'bp_won',
'p2_tot_pt_won_s': 'tot_pt_won_s',
                        'p2_tot_pt_won_r': 'tot_pt_won_r',
'p2_tot_pt_won': 'tot_pt_won',
                        'p2_gms_won_s': 'gms_won_s',
'p2_gms_won_r': 'gms_won_r',
                        'p2_tot_gms_won': 'tot_gms_won',
'p1_name': 'opponent_name'}, inplace=True)
df_p2['result'] = 0
df_p2.head()

# In[8]:
df_p2.shape

# In[9]:
data = pd.concat([df_p1, df_p2], ignore_index=True)
data.sort_values(by=['tourney_date', 'tourney_name'], inplace =
True, ignore_index=True)
data.head()

# In[10]:
data.tail()

# In[11]:
data.shape

# In[12]:
data.to_csv('ready_for_pred.csv', index=False) # data to predict
stats

# In[13]:
data.loc[(data['tourney_level'] == 'A')].tail()

# In[14]:
data.loc[(data['tourney_date'] == '2023-04-17') &
(data['tourney_level'] == 'A')]['tourney_name'].unique()
```

```
# In[15]:
data.loc[(data['tourney_date'] == '2023-05-01') &
(data['tourney_level'] == 'C')]['tourney_name'].unique()

# In[16]:
data.loc[(data['tourney_date'] == '2023-04-24') &
(data['tourney_level'] == 'M')]['tourney_name'].unique()

# In[17]:
# tournament to predict is Rome (M) (Italian Open). Date: 2023-
05-10
```

Лістинг 7 – Моделювання результатів матчів

```
# In[1]:
import pandas as pd
pd.set_option('display.max_columns', None)

import numpy as np

import warnings
warnings.filterwarnings('ignore')

from datetime import datetime

from sklearn.linear_model import LogisticRegression
from joblib import load

logreg = load('logreg.joblib')

def recall_m(y_true, y_pred):
```

```
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0,
1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0,
1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives +
K.epsilon())
    return precision

def f1score_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))

import keras
from keras.models import load_model

with keras.utils.custom_object_scope({'precision_m':
precision_m, 'recall_m': recall_m,
                                       'f1score_m': f1score_m}):
    MLP = load_model('MLP.h5')

from dateutil.relativedelta import relativedelta

# In[2]:
surface = 'Clay'
draw_size = 128
tourney_level = 'M'
round_ = 'Q1'
```

```
p1_hand = 'R'
p1_ht = 191
p1_birthdate = datetime(1993, 4, 15)  # рік місяць день
p1_rank = 170
p1_pts = 348

p2_hand = 'R'
p2_ht = 191
p2_birthdate = datetime(2001, 6, 21)
p2_rank = 117
p2_pts = 528

p1_name = 'Franco Agamenone'
p2_name = 'Otto Virtanen'

# p1_hand = 'R'
# p1_ht = 188
# p1_birthdate = datetime(2003, 4, 29)
# p1_rank = 7
# p1_pts = 3865

# p2_hand = 'R'
# p2_ht = 198
# p2_birthdate = datetime(1996, 2, 11)  # рік місяць день
# p2_rank = 3
# p2_pts = 5330

# p1_name = 'Holger Rune'
# p2_name = 'Daniil Medvedev'

curr_date = datetime(2023, 5, 10)

# In[3]:
# data[data['player_name'] == 'Carlos Alcaraz']
```

```
# In[4]:
data = pd.read_csv('ready_for_pred_added.csv')
data.shape

# In[5]:
data.tail()

# In[6]:
data.columns

# In[7]:
data.isnull().sum().sum()

# In[8]:
data[['tourney_date']].dtypes

# In[9]:
data['tourney_date'] = pd.to_datetime(data['tourney_date'])
data[['tourney_date']].dtypes

# In[10]:
d_surface = {'Carpet': [1, 0, 0, 0],
               'Clay': [0, 1, 0, 0],
              'Grass': [0, 0, 1, 0],
               'Hard': [0, 0, 0, 1]}

d_draw = { 8:  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          12:  [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          16:  [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
          18:  [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
          24:  [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
          28:  [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
          32:  [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
          48:  [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
          56:  [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
          64:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
```

```
          96:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
          128: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]}

d_tourn_lvl = {'A': [1, 0, 0, 0],
               'C': [0, 1, 0, 0],
               'F': [0, 0, 1, 0],
               'M': [0, 0, 0, 1]}

d_round = { 'BR': [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            'ER': [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
             'F': [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            'Q1': [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            'Q2': [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
            'Q3': [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
            'QF': [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
          'R128': [0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
           'R16': [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
           'R32': [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
           'R64': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
            'RR': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
            'SF': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]}

# In[11]:
def get_data(surface, draw_size, tourney_level, round_, p1_hand,
p1_ht, p1_birthdate, p1_rank,
             p1_pts, p2_hand, p2_ht, p2_birthdate, p2_rank,
p2_pts):

    final_lst = []
    final_lst = d_surface.get(surface)
    final_lst = final_lst + d_draw.get(draw_size)
    final_lst = final_lst + d_tourn_lvl.get(tourney_level)

    if p1_hand == 'R':
        final_lst.append(1)
    else:
```

```
        final_lst.append(0)

    if p2_hand == 'R':
        final_lst.append(1)
    else:
        final_lst.append(0)

    final_lst = final_lst + d_round.get(round_)
    final_lst.append(p1_ht-p2_ht)

    difference = p2_birthdate - p1_birthdate
    difference = difference.days / 365.25
    final_lst.append(round(difference, 1))

    final_lst.append(p1_rank-p2_rank)
    final_lst.append(p1_pts-p2_pts)

    return final_lst

# In[12]:
part1 = get_data(surface=surface, draw_size=draw_size,
tourney_level=tourney_level, round_=round_,
         p1_hand=p1_hand, p1_ht=p1_ht,
p1_birthdate=p1_birthdate, p1_rank=p1_rank, p1_pts=p1_pts,
         p2_hand=p2_hand, p2_ht=p2_ht,
p2_birthdate=p2_birthdate, p2_rank=p2_rank, p2_pts=p2_pts)

print(part1)

# In[13]:
df_p1_all = data[data['player_name']==p1_name]  # стата з усіх
покриттів
df_p1_surf = data[(data['player_name']==p1_name) &
(data['surface']==surface)]  # стата з обраного покриття
df_p2_all = data[data['player_name']==p2_name]
```

```
df_p2_surf = data[(data['player_name']==p2_name) &
(data['surface']==surface)]

# In[14]:
# time discounting function
def W(t, f=0.7):
    return round(min(f, f**t), 10)

# In[15]:
stats_list = ['1st_in', '1st_ptwon', '2nd_ptwon', 'bp_saved',
'ace_per_s', 'df_per_s',
              '1st_r_ptwon', '2nd_r_ptwon', 'bp_won',
'tot_pt_won_s', 'tot_pt_won_r',
              'tot_pt_won', 'gms_won_s', 'gms_won_r',
'tot_gms_won']

# In[16]:
def predict_stats(df, date, stats_list=stats_list):
    '''''
    years(list) - список, який зберігає різницю між заданою
датою і датою матчу, що пройшов
    weights(list) - список, який обраховує ваги для кожного
матчу гравця
    stats_pred(list) - список, який містить прогнозовану
статистику
    '''''

    df_copy = df.copy()
    mnth = []
    for i in df_copy['tourney_date'].tolist():
        diff = relativedelta(date, i)
        months = diff.years * 12 + diff.months
        mnth.append(round(months, 1))

    weights = []
    for i in mnth:
```

```
        w = W(i)
        weights.append(w)

    s = sum(weights)

    weights_norm = [round(i / s, 10) for i in weights]  # нормалізовані ваги, в сумі має бути 1

    df_copy['weights_norm'] = weights_norm  # додаємо колонку weights_norm в датасет

    for i in stats_list:
        df_copy[i] = df_copy[i] * df_copy['weights_norm']  # множимо статистику на ваги

    stats_pred = []
    for i in stats_list:
        stats_pred.append(round(df_copy[i].sum(), 2))

    return stats_pred

# In[17]:
p1_stats_all = predict_stats(df_p1_all, date=curr_date)
print(p1_stats_all)
p2_stats_all = predict_stats(df_p2_all, date=curr_date)
print(p2_stats_all)
stats_diff_all = [round(el1 - el2, 2) for el1, el2 in zip(p1_stats_all, p2_stats_all)]
print(stats_diff_all)

# In[18]:
p1_stats_surf = predict_stats(df_p1_surf, date=curr_date)
p2_stats_surf = predict_stats(df_p2_surf, date=curr_date)
stats_diff_surf = [round(el1 - el2, 2) for el1, el2 in zip(p1_stats_surf, p2_stats_surf)]
print(stats_diff_surf)
```

```
# In[19]:
final_data_all = part1 + stats_diff_all
final_data_surf = part1 + stats_diff_surf
print(final_data_all)

# In[20]:
cols = ['surface_Carpet', 'surface_Clay', 'surface_Grass',
'surface_Hard',
       'draw_size_8', 'draw_size_12', 'draw_size_16',
'draw_size_18',
       'draw_size_24', 'draw_size_28', 'draw_size_32',
'draw_size_48',
       'draw_size_56', 'draw_size_64', 'draw_size_96',
'draw_size_128',
       'tourney_level_A', 'tourney_level_C', 'tourney_level_F',
       'tourney_level_M', 'p1_hand_R', 'p2_hand_R', 'round_BR',
'round_ER',
       'round_F', 'round_Q1', 'round_Q2', 'round_Q3',
'round_QF', 'round_R128',
       'round_R16', 'round_R32', 'round_R64', 'round_RR',
'round_SF',
       'ht_diff', 'age_diff', 'rank_diff', 'rank_pts_diff',
'1st_in_diff',
       '1st_ptwon_diff', '2nd_ptwon_diff', 'bp_saved_diff',
'ace_per_s_diff',
       'df_per_s_diff', '1st_r_ptwon_diff', '2nd_r_ptwon_diff',
'bp_won_diff',
       'tot_pt_won_s_diff', 'tot_pt_won_r_diff',
'tot_pt_won_diff',
       'gms_won_s_diff', 'gms_won_r_diff', 'tot_gms_won_diff']

# In[21]:
match_to_predict_all_lr = pd.DataFrame(final_data_all,
index=cols)
```

```
match_to_predict_all_lr =
match_to_predict_all_lr.swapaxes('index', 'columns')
match_to_predict_all_lr

# In[22]:
match_to_predict_all_mlp =
match_to_predict_all_lr.drop(['rank_diff', 'rank_pts_diff'],
axis=1)
match_to_predict_all_mlp

# In[23]:
match_to_predict_surf_lr = pd.DataFrame(final_data_surf,
index=cols)
match_to_predict_surf_lr =
match_to_predict_surf_lr.swapaxes('index', 'columns')
match_to_predict_surf_lr

# In[24]:
match_to_predict_surf_mlp =
match_to_predict_surf_lr.drop(['rank_diff', 'rank_pts_diff'],
axis=1)
match_to_predict_surf_mlp

# In[25]:
pred_lr_all = logreg.predict_proba(match_to_predict_all_lr)[:,
1]
pred_lr_surf = logreg.predict_proba(match_to_predict_surf_lr)[:,
1]
pred_mlp_all = MLP.predict(match_to_predict_all_mlp)
pred_mlp_surf = MLP.predict(match_to_predict_surf_mlp)

print('\nПРОГНОЗ ЗА ДОПОМОГОЮ МОДЕЛІ ЛОГІСТИЧНОЇ РЕГРЕСІЇ:\n')
print(f'Прогноз на усіх покриттях: Ймовірність перемоги
{p1_name} становить {round(pred_lr_all[0], 2)}')
print(f'Прогноз на покритті {surface}: Ймовірність перемоги
{p1_name} становить {round(pred_lr_surf[0], 2)}')
```

```
print('\nПРОГНОЗ ЗА ДОПОМОГОЮ МУЛЬТИШАРОВОГО ПЕРСЕПТРОНУ:\n')
print(f'Прогноз на усіх покриттях: Ймовірність перемоги
{p1_name} становить', round(pred_mlp_all[0][0], 2))
print(f'Прогноз на покритті {surface}: Ймовірність перемоги
{p1_name} становить', round(pred_mlp_surf[0][0], 2))
```

Лістинг 8 – Додавання матчів кваліфікації турніру в Римі до датасету

```
# In[1]:
import numpy as np

import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import warnings
warnings.filterwarnings('ignore')

# In[2]:
data_main = pd.read_csv('ready_for_pred.csv')
data_main.shape

# In[3]:
data_Q1 = pd.read_excel('Q1_rome.xlsx', header=1)
data_Q1.shape

# In[4]:
data_Q1

# In[5]:
data_main = pd.concat([data_main, data_Q1], axis=0,
ignore_index=True)
data_main.tail(50)
```

```
# In[6]:
data_main.to_csv('ready_for_pred_added.csv', index=False)

# In[7]:
data_main_q1 = pd.read_csv('ready_for_pred_added.csv')
data_main_q1.tail()

# In[8]:
data_Q2 = pd.read_excel('Q2_rome.xlsx', header=1)
data_Q2.shape

# In[9]:
data_Q2

# In[10]:
data_main_q1q2 = pd.concat([data_main_q1, data_Q2], axis=0,
ignore_index=True)
data_main_q1q2.tail(25)

# In[11]:
data_main_q1q2.to_csv('ready_for_pred_added.csv', index=False)
```

Лістинг 9 – Додавання матчів основного раунду турніру в Римі до датасету

```
# In[1]:
import numpy as np

import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

```
import warnings
warnings.filterwarnings('ignore')

# In[2]:
data_old = pd.read_csv('ready_for_pred_added.csv')
data_old.tail()

# In[3]:
data_old.shape

# In[4]:
data = pd.read_csv('atp_2023_wRome.csv')
data_rome = data[data['tourney_name'] == 'Rome Masters']
data_rome.shape

# In[5]:
data_rome.tail()

# In[6]:
data_rome.drop(['tourney_id', 'match_num', 'winner_id',
'winner_seed', 'winner_entry', 'winner_hand',
                'winner_ht', 'winner_ioc', 'winner_age',
'loser_id', 'loser_seed', 'loser_entry',
                'loser_hand', 'loser_ht', 'loser_ioc',
'loser_age',  'best_of', 'minutes',
                'winner_rank', 'winner_rank_points',
'loser_rank', 'loser_rank_points'], axis=1,
                inplace=True)

data_rome

# In[7]:
data_rome.isnull().sum()

# In[8]:
```

```
to_drop = data_rome[data_rome['score'].str.contains('RET',
na=False)].index
to_drop

# In[9]:
data_rome.drop(to_drop, inplace=True)
data_rome.reset_index(drop=True, inplace=True)
data_rome.shape

# In[10]:
import re

# In[11]:
scores_list = data_rome['score'].values.tolist()
scores_list[:6]

# In[12]:
scores_clear = []
for score in scores_list:

    curr_scr = re.sub('\(\d*\)', '', score) # delete scores in
brackets ()

    curr_scr = curr_scr.rstrip() # delete space after the last
number if exists

    # do not take into consideration tiebreaks
    curr_scr = curr_scr.split()
    for i in range(len(curr_scr)):
        if '7-6' in curr_scr[i]:
            curr_scr[i] = curr_scr[i].replace('7-', '6-')
        elif '6-7' in curr_scr[i]:
            curr_scr[i] = curr_scr[i].replace('-7', '-6')


    new_scr = ' '.join(curr_scr)
```

```
    new_scr = re.sub(r'\D', ' ', new_scr) # delete all symbols
exept numeric
    scores_clear.append(new_scr)
scores_clear[:6]

# In[13]:
winner_games, loser_games = [], []
# count total games for each player
# scores_clear - list of clear scores ['7 6 6 3 6 2', '7 6 6 0 6
2', '3 6 7 6 6 2 4 1']
# splitted list of scores ['7', '6', '6', '2', '6', '3']
for score in scores_clear:
    win_g, los_g = 0, 0
    splitted = score.split(' ')
    for el in range(len(splitted)):
        if (el+1)%2 == 1:
            win_g += int(splitted[el])
        elif (el+1)%2 == 0:
            los_g += int(splitted[el])
    winner_games.append(win_g)
    loser_games.append(los_g)

# In[14]:
# adding cols of total games by each player and total
data_rome['w_games'] = winner_games
data_rome['l_games'] = loser_games
data_rome['total_games'] = data_rome['w_games'] +
data_rome['l_games']
data_rome.tail()

# In[15]:
data_rome['p1_1st_in'] = round(data_rome['w_1stIn'] /
data_rome['w_svpt'], 2)
data_rome['p1_1st_ptwon'] = round(data_rome['w_1stWon'] /
data_rome['w_1stIn'], 2)
```

```
data_rome['p1_2nd_ptwon'] = round(data_rome['w_2ndWon'] /
(data_rome['w_svpt'] - data_rome['w_1stIn']), 2)
data_rome['p1_bp_saved'] = np.where(data_rome['w_bpFaced'] == 0,
1.0, round(data_rome['w_bpSaved'] / data_rome['w_bpFaced'], 2))
data_rome['p1_ace_per_s'] = round(data_rome['w_ace'] /
data_rome['w_svpt'], 2)
data_rome['p1_df_per_s'] = round(data_rome['w_df'] /
data_rome['w_svpt'], 2)
data_rome['p1_1st_r_ptwon'] = round((data_rome['l_1stIn'] -
data_rome['l_1stWon']) / data_rome['l_1stIn'], 2)
data_rome['p1_2nd_r_ptwon'] = round((data_rome['l_svpt'] -
data_rome['l_1stIn'] - data_rome['l_2ndWon']) /
(data_rome['l_svpt'] - data_rome['l_1stIn']), 2)
data_rome['p1_bp_won'] = np.where(data_rome['l_bpFaced'] == 0,
0.0, round((data_rome['l_bpFaced'] - data_rome['l_bpSaved']) /
data_rome['l_bpFaced'], 2))
data_rome['p1_tot_pt_won_s'] = round((data_rome['w_1stWon'] +
data_rome['w_2ndWon']) / data_rome['w_svpt'], 2)
data_rome['p1_tot_pt_won_r'] = round((data_rome['l_svpt'] -
data_rome['l_1stWon'] - data_rome['l_2ndWon']) /
data_rome['l_svpt'], 2)
data_rome['p1_tot_pt_won'] = round((data_rome['w_1stWon'] +
data_rome['w_2ndWon'] + (data_rome['l_svpt'] -
data_rome['l_1stWon'] - data_rome['l_2ndWon'])) /
(data_rome['w_svpt'] + data_rome['l_svpt']), 2)
data_rome['p1_gms_won_s'] = round((data_rome['w_SvGms'] -
(data_rome['w_bpFaced'] - data_rome['w_bpSaved'])) /
data_rome['w_SvGms'], 2)
data_rome['p1_gms_won_r'] = round((data_rome['l_bpFaced'] -
data_rome['l_bpSaved']) /  data_rome['l_SvGms'], 2)
data_rome['p1_tot_gms_won'] = round(data_rome['w_games'] /
data_rome['total_games'], 2)

data_rome['p2_1st_in'] = round(data_rome['l_1stIn'] /
data_rome['l_svpt'], 2)
```

```
data_rome['p2_1st_ptwon'] = round(data_rome['l_1stWon'] /
data_rome['l_1stIn'], 2)
data_rome['p2_2nd_ptwon'] = round(data_rome['l_2ndWon'] /
(data_rome['l_svpt'] - data_rome['l_1stIn']), 2)
data_rome['p2_bp_saved'] = np.where(data_rome['l_bpFaced'] == 0,
1.0, round(data_rome['l_bpSaved'] / data_rome['l_bpFaced'], 2))
data_rome['p2_ace_per_s'] = round(data_rome['l_ace'] /
data_rome['l_svpt'], 2)
data_rome['p2_df_per_s'] = round(data_rome['l_df'] /
data_rome['l_svpt'], 2)
data_rome['p2_1st_r_ptwon'] = round((data_rome['w_1stIn'] -
data_rome['w_1stWon']) / data_rome['w_1stIn'], 2)
data_rome['p2_2nd_r_ptwon'] = round((data_rome['w_svpt'] -
data_rome['w_1stIn'] - data_rome['w_2ndWon']) /
(data_rome['w_svpt'] - data_rome['w_1stIn']), 2)
data_rome['p2_bp_won'] = np.where(data_rome['w_bpFaced'] == 0,
0.0, round((data_rome['w_bpFaced'] - data_rome['w_bpSaved']) /
data_rome['w_bpFaced'], 2))
data_rome['p2_tot_pt_won_s'] = round((data_rome['l_1stWon'] +
data_rome['l_2ndWon']) / data_rome['l_svpt'], 2)
data_rome['p2_tot_pt_won_r'] = round((data_rome['w_svpt'] -
data_rome['w_1stWon'] - data_rome['w_2ndWon']) /
data_rome['w_svpt'], 2)
data_rome['p2_tot_pt_won'] = round((data_rome['l_1stWon'] +
data_rome['l_2ndWon'] + (data_rome['w_svpt'] -
data_rome['w_1stWon'] - data_rome['w_2ndWon'])) /
(data_rome['l_svpt'] + data_rome['w_svpt']), 2)
data_rome['p2_gms_won_s'] = round((data_rome['l_SvGms'] -
(data_rome['l_bpFaced'] - data_rome['l_bpSaved'])) /
data_rome['l_SvGms'], 2)
data_rome['p2_gms_won_r'] = round((data_rome['w_bpFaced'] -
data_rome['w_bpSaved']) /  data_rome['w_SvGms'], 2)
data_rome['p2_tot_gms_won'] = round(data_rome['l_games'] /
data_rome['total_games'], 2)
data_rome.tail()
```

```
# In[16]:
df = data_rome.select_dtypes((int, float))
# check inf
count = np.isinf(df).values.sum()
count

# In[17]:
data_rome.drop(['w_ace', 'w_df', 'w_svpt', 'w_1stIn',
'w_1stWon', 'w_2ndWon',
           'w_SvGms', 'w_bpSaved', 'w_bpFaced', 'l_ace', 'l_df',
'l_svpt',
           'l_1stIn', 'l_1stWon', 'l_2ndWon', 'l_SvGms',
'l_bpSaved', 'l_bpFaced',
           'score', 'w_games', 'l_games', 'total_games'],
axis=1, inplace=True)
data_rome.tail()

# In[18]:
# let`s change tourney_date type to datetime
data_rome = data_rome.astype({'tourney_date': 'string'})
data_rome['tourney_date'] =
pd.to_datetime(data_rome['tourney_date'])
data_rome.tail()

# In[19]:
df_p1 = data_rome[['tourney_name', 'surface', 'draw_size',
'tourney_level', 'tourney_date', 'winner_name',
              'round', 'p1_1st_in', 'p1_1st_ptwon',
'p1_2nd_ptwon', 'p1_bp_saved', 'p1_ace_per_s',
              'p1_df_per_s', 'p1_1st_r_ptwon', 'p1_2nd_r_ptwon',
'p1_bp_won', 'p1_tot_pt_won_s',
              'p1_tot_pt_won_r', 'p1_tot_pt_won',
'p1_gms_won_s', 'p1_gms_won_r', 'p1_tot_gms_won',
              'loser_name']]
df_p1.rename(columns = {'winner_name': 'player_name',
'p1_1st_in': '1st_in', 'p1_1st_ptwon': '1st_ptwon',
```

```
                        'p1_2nd_ptwon': '2nd_ptwon',
'p1_bp_saved': 'bp_saved',
                        'p1_ace_per_s': 'ace_per_s',
'p1_df_per_s': 'df_per_s',
                        'p1_1st_r_ptwon': '1st_r_ptwon',
'p1_2nd_r_ptwon': '2nd_r_ptwon',
                        'p1_bp_won': 'bp_won',
'p1_tot_pt_won_s': 'tot_pt_won_s',
                        'p1_tot_pt_won_r': 'tot_pt_won_r',
'p1_tot_pt_won': 'tot_pt_won',
                        'p1_gms_won_s': 'gms_won_s',
'p1_gms_won_r': 'gms_won_r',
                        'p1_tot_gms_won': 'tot_gms_won',
'loser_name': 'opponent_name'}, inplace=True)
df_p1['result'] = 1
df_p1.head()

# In[20]:
df_p2 = data_rome[['tourney_name', 'surface', 'draw_size',
'tourney_level', 'tourney_date', 'loser_name',
              'round', 'p2_1st_in', 'p2_1st_ptwon',
'p2_2nd_ptwon', 'p2_bp_saved', 'p2_ace_per_s',
              'p2_df_per_s', 'p2_1st_r_ptwon', 'p2_2nd_r_ptwon',
'p2_bp_won', 'p2_tot_pt_won_s',
              'p2_tot_pt_won_r', 'p2_tot_pt_won',
'p2_gms_won_s', 'p2_gms_won_r', 'p2_tot_gms_won',
              'winner_name']]
df_p2.rename(columns = {'loser_name': 'player_name',
'p2_1st_in': '1st_in', 'p2_1st_ptwon': '1st_ptwon',
                        'p2_2nd_ptwon': '2nd_ptwon',
'p2_bp_saved': 'bp_saved',
                        'p2_ace_per_s': 'ace_per_s',
'p2_df_per_s': 'df_per_s',
                        'p2_1st_r_ptwon': '1st_r_ptwon',
'p2_2nd_r_ptwon': '2nd_r_ptwon',
```

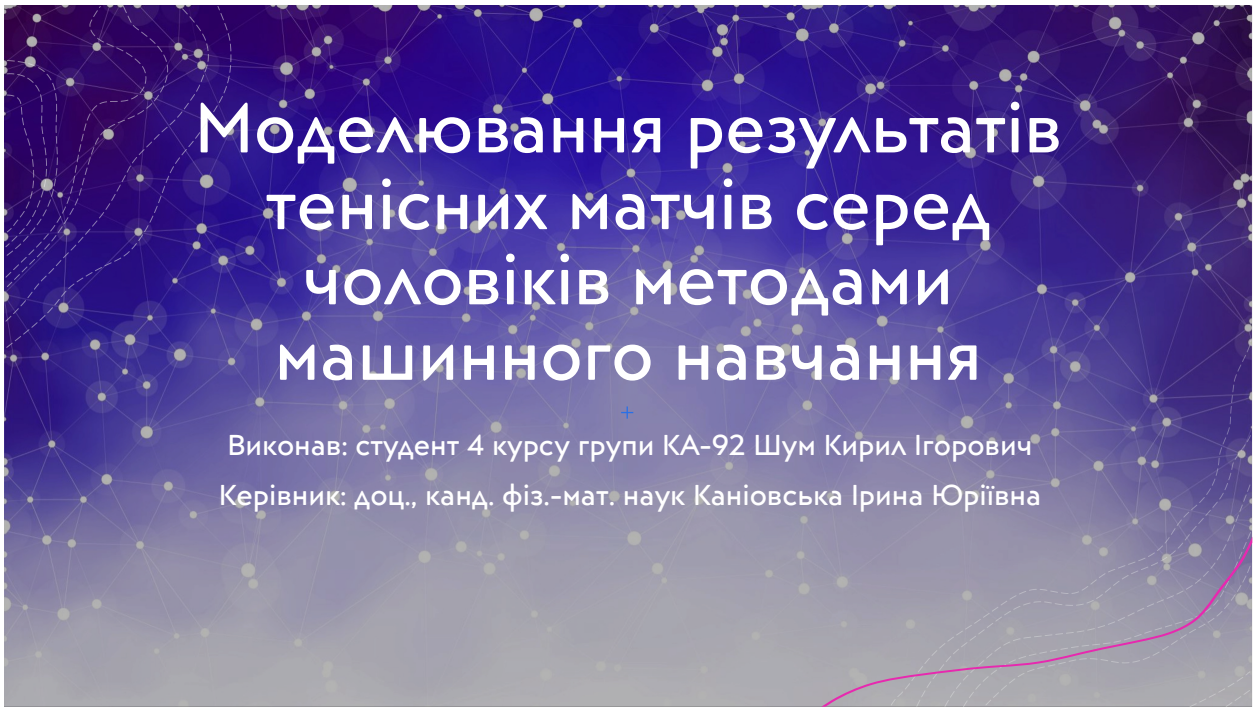
```
                        'p2_bp_won': 'bp_won',
'p2_tot_pt_won_s': 'tot_pt_won_s',
                        'p2_tot_pt_won_r': 'tot_pt_won_r',
'p2_tot_pt_won': 'tot_pt_won',
                        'p2_gms_won_s': 'gms_won_s',
'p2_gms_won_r': 'gms_won_r',
                        'p2_tot_gms_won': 'tot_gms_won',
'winner_name': 'opponent_name'}, inplace=True)
df_p2['result'] = 0
df_p2.head()

# In[21]:
data_rome = pd.concat([df_p1, df_p2], ignore_index=True)
data_rome.shape

# In[22]:
# add R128
R128 = data_rome[data_rome['round']=='R128']
R128.shape

# In[23]:
data_old = pd.concat([data_old, R128], axis=0,
ignore_index=True)
data_old.shape

# In[24]:
data_old.tail()

# In[25]:
# data_old.to_csv('ready_for_pred_added.csv', index=False)

# In[26]:
# add R64
R64 = data_rome[data_rome['round']=='R64']
R64.shape
```

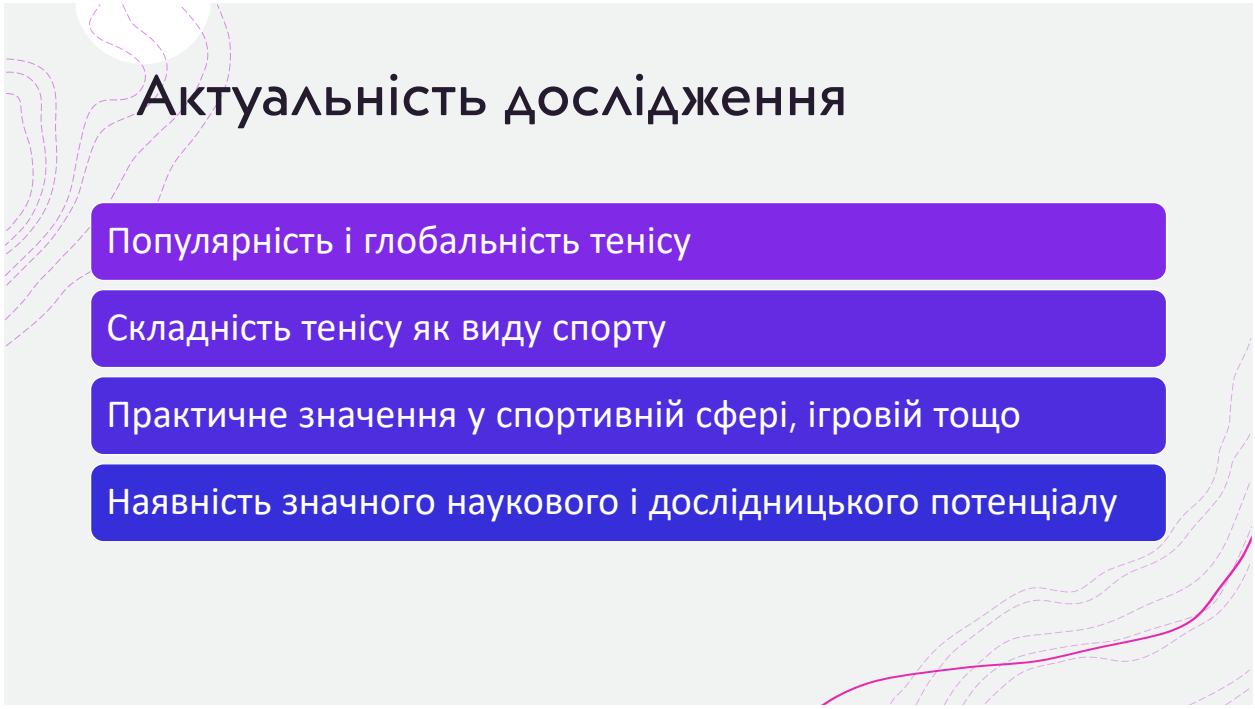
```
# In[27]:
data_old = pd.concat([data_old, R64], axis=0, ignore_index=True)
data_old.shape

# In[28]:
data_old.tail()

# In[29]:
# data_old.to_csv('ready_for_pred_added.csv', index=False)

# In[30]:
# add R32
R32 = data_rome[data_rome['round']=='R32']
R32.shape

# In[31]:
data_old = pd.concat([data_old, R32], axis=0, ignore_index=True)
data_old.shape

# In[32]:
data_old.to_csv('ready_for_pred_added.csv', index=False)

# In[33]:
# add R16
R16 = data_rome[data_rome['round']=='R16']
R16.shape

# In[34]:
data_old = pd.concat([data_old, R16], axis=0, ignore_index=True)
data_old.shape

# In[36]:
# data_old.to_csv('ready_for_pred_added.csv', index=False)

# In[37]:
# add QF
```

```
QF = data_rome[data_rome['round']=='QF']
QF.shape

# In[38]:
data_old = pd.concat([data_old, QF], axis=0, ignore_index=True)
data_old.shape

# In[39]:
# data_old.to_csv('ready_for_pred_added.csv', index=False)

# In[40]:
# add SF
SF = data_rome[data_rome['round']=='SF']
SF.shape

# In[41]:
data_old = pd.concat([data_old, SF], axis=0, ignore_index=True)
data_old.shape

# In[42]:
# data_old.to_csv('ready_for_pred_added.csv', index=False)

# In[43]:
# add F
F = data_rome[data_rome['round']=='F']
F.shape

# In[44]:
data_old = pd.concat([data_old, F], axis=0, ignore_index=True)
data_old.shape

# In[45]:
data_old.to_csv('ready_for_pred_added.csv', index=False)
```

ДОДАТОК Б

ПРЕЗЕНТАЦІЯ





Об'єкт дослідження: Тенісні матчі серед чоловіків

Предмет дослідження: Методи машинного навчання для моделювання та прогнозування результатів тенісних матчів серед чоловіків

Мета дослідження: створення програмного продукту, здатного моделювати результати тенісних матчів серед чоловіків на основі аналізу історичних даних та за допомогою застосування методів машинного навчання

Постановка задачі

- Дослідити предметну область
- Обрати алгоритми машинного навчання, на основі яких буде проведено прогнозування
- Побудувати прогностичні моделі на основі обраних методів МН
- Проаналізувати отримані результати та зробити висновки

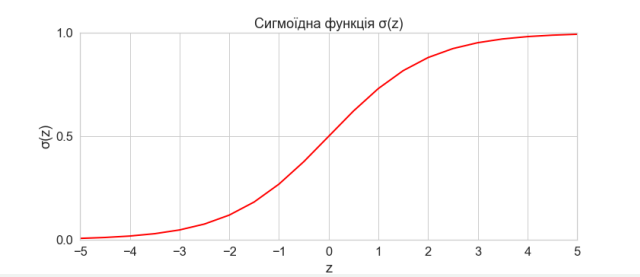
Складові розробленої програми моделювання результатів тенісних матчів

1. Модель логістичної регресії
2. Модель мультишарового персептрону
3. Алгоритм прогнозування статистичних показників майбутнього матчу
4. База даних з усіма статистичними викладками попередніх матчів

Логістична регресія

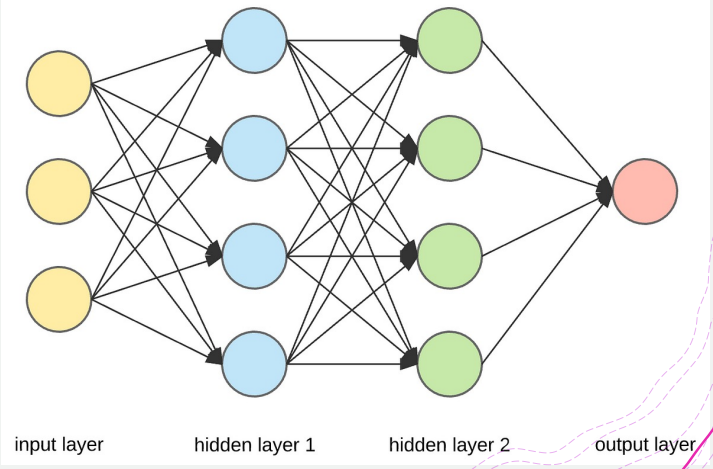
$\sigma(z) = \frac{1}{1 + e^{-z}}$ – логістична функція, де $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$ – лінійна комбінація незалежних змінних x_i та їх коефіцієнтів β_i, $i = \overline{1, n}$, а n – кількість вхідних змінних (предикторів).

$LogLoss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$ – функція логістичних втрат, де N – кількість вхідних прикладів, y_i – реальне значення i–того прикладу, а $p_i = \sigma(z_i)$ – прогнозована ймовірність для i–того прикладу.



Мультишаровий персептрон

- є різновидом feed-forward мережі або ж нейронної мережі «прямого поширення»
- використовує нелінійні функції активації для створення нелінійного зв'язку між нейронами
- архітектура складається з вхідного та вихідного шарів, а також з довільної кількості прихованих шарів нейронів



Алгоритм прогнозування статистичних показників майбутнього матчу

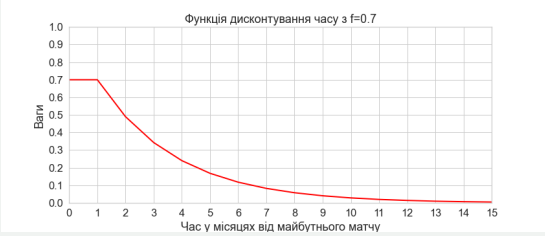
1. Обчислити різницю місяців між датами попередніх матчів та датою майбутнього матчу.
2. Обрахувати вагу для кожного історичного матча, використовуючи функцію дисконтування часу:

$$W(t) = \min(f^t,\ f),$$

де $W(t)$ – шукане вагове значення, t – час між матчами у місяцях, f – дисконтний коефіцієнт.

$f \in [0,1]$ та обирається дослідником самостійно. Чим менше значення f, тим меншу вагу матимуть давніші матчі.

3. Знайти суму всіх обрахованих ваг та пронормувати кожну вагу шляхом її ділення на значення обрахованої суми.
4. Для кожного матчу помножити статистичні показники на відповідну нормовану вагу.
5. Для відповідних статистичних показників просумувати знайдений у пункті 4. добуток по всіх матчах.



Метрики оцінювання якості моделей машинного навчання

Метрики		
Назва	**Опис**	**Формула**
Accuracy (Точність)	визначає відсоток правильних передбачень моделі відносно загальної кількості прикладів	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision (Точність)	вказує на те, наскільки точні передбачення моделі щодо позитивного класу	$\frac{TP}{TP+FP}$
Recall (Повнота)	показує, скільки позитивних класів передбачено правильно	$\frac{TP}{TP+FN}$
F1-Score (F1-показник)	використовується для оцінки збалансованості між Precision і Recall	$2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$
Loss (Втрата)	показує відповідність між прогнозованими ймовірностями і фактичними значеннями цільової змінної. Чим менше значення, тим краще	$-\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1-y_i)\log(1-p_i))$
AUC ROC	вимірює здатність моделі розрізняти класи та ранжувати їх відповідно до ймовірності належності до певного класу. AUC ROC це площа під графіком ROC, який відображає істиннопозитивний (TPR) і хибнопозитивний (FPR) рівні моделі	$TPR = \frac{TP}{TP+FN};\ FPR = \frac{FP}{FP+TN}$

Показники класифікації, що використовуються для розрахунку метрик			
TP (True Positive)	TN (True Negative)	FP (False Positive)	FN (False Negative)
передбачає 1 як 1	передбачає 0 як 0	передбачає 0 як 1	передбачає 1 як 0

Побудована модель логістичної регресії

Найкращі параметри побудованої моделі логістичної регресії		
Назва	**Опис**	**Значення**
test_size	Кількість даних розбиття для тестового набору	0.1
solver	Метод знаходження оптимальних ваг моделі, які мінімізують функцію втрат	newton-sg
fit_intercept	Наявність вільного коефіцієнту у рівнянні моделі	True
C	Параметр регуляризації	10
penalty	Тип регуляризації, яка використовується для контролю за перенавчанням моделі.	L2

Оцінка якості побудованої моделі логістичної регресії						
Вибірка	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Roc-Auc**	**Loss**
Навчальна	0.95286	0.95283	0.95300	0.95292	0.99222	0.11329
Тестова	0.95047	0.95031	0.94966	0.94998	0.99195	0.11535

Побудована модель мультишарового персептрону

- архітектура складається з вхідного шару, трьох прихованих та вихідного
- (64, 'tanh'), (32, 'relu'), (32, 'tanh'), (1, 'simoid') – кількість нейронів та функція активації для трьох прихованих шарів та вихідного шару відповідно
- 0.1 – кількість даних розбиття для тестового набору
- Adam – оптимізаційний алгоритм, який відповідає за оновлення ваг мережі
- 'Binary_crossentropy' – функція втрат, яка використовується для навчання моделі
- застосовано метод раннього відключення (Early Stopping) для запобігання перенавчання мережі

Оцінка якості побудованої моделі мультишарового персептрону						
Вибірка	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Roc-Auc**	**Loss**
Навчальна	0.95260	0.94839	0.95750	0.95148	0.99216	0.1137
Тестова	0.95283	0.94662	0.95974	0.95162	0.99216	0.1141

Застосування розробленої програми для моделювання результатів матчів на турнірі в Римі 2023

- Турнір у Римі є престижним грунтовим турніром категорії Мастерс. У 2023 році у ньому прийняли участь 130 тенісистів.
- Турнір складається з 9 раундів: двох кваліфікаційних та семи основних.
- Загалом було спрогнозовано результати 131 матчу.
- Для моделювання результатів було застосовано розроблені моделі логістичної регресії та мультишарового персептрону у комбінації із алгоритмом прогнозування статистичних показників (на всіх покриттях та тільки на одному).
- Загалом вийшло 4 такі комбінації.
- Результати прогнозування було зведено до таблиць, що містять імена гравців, коефіцієнти на перемогу кожного гравця, запропоновані букмекерською компанією Favbet, а також прогнози на переможців матчу, отримані в результаті застосування розробленої програми, із зазначенням ймовірності такого прогнозу.

Приклад таблиці, де спрогнозовано результати перших чотирьох матчів першого кваліфікаційного раунду						
№	Матч	БК	Всі покриття	Обране покриття	Всі покриття	Обране покриття
1	Попирін А. Бонадіо Р.	1.40 2.75	Попирін А. (0.66)	Попирін А. (0.65)	Попирін А. (0.67)	Попирін А. (0.64)
2	Пеллегріно А. Бранкаччо Р.	1.68 2.10	Пеллегріно А. (0.8)	Пеллегріно А. (0.89)	Пеллегріно А. (0.81)	Пеллегріно А. (0.9)
3	Альтмаєр Д. Ковалік Й.	1.25 3.70	Альтмаєр Д. (0.9)	Альтмаєр Д. (0.92)	Альтмаєр Д. (0.89)	Альтмаєр Д. (0.91)
4	Імер Е. Махач Т.	2.75 1.41	Махач Т. (0.94)	Махач Т. (0.92)	Махач Т. (0.94)	Махач Т. (0.9)

Критерії оцінки розробленого програмного продукту

Точність. Точність прогнозування свідчить про надійність розробленої програми і показує, чи можна довіряти результатам її роботи і приймати рішення на їх основі

Прибуток. Прибуток, який приносить модель прогнозування, є важливим показником якості розробленого продукту, оскільки цей показник відображає ефективність та доцільність використання такої програми

Загалом кажучи, побудовані моделі можуть давати більший прибуток у порівнянні з іншими моделями навіть не зважаючи на те, що їх точність може виявитись нижчою. Такий парадокс може бути зумовлений розподілом коефіцієнтів на гравців. Наприклад, якщо явний фаворит буде програвати аутсайдеру, але при цьому модель передбачатиме саме перемогу слабшого суперника.

Результати роботи розробленої програми на турнірі в Римі

	Кількість правильних прогнозів	Відсоток правильних прогнозів	Прибуток
Логістична регресія та алгоритм прогнозування по всіх покриттях	85/126	67	+889
Мультишаровий персептрон та алгоритм прогнозування по всіх покриттях	83/125	66	+935
Модель букмекерської компанії Favbet	91/127	72	+229

Висновки

1

Було створено програмний продукт, що здійснює моделювання результатів тенісних матчів серед чоловіків, використовуючи методи машинного навчання.

2

Окремо моделі показали високі результати: по 95% точно зроблених прогнозів.

3

У комбінації з алгоритмом прогнозування статистики майбутнього матчу побудовані моделі показали точність моделювання на рівні 66-67 відсотків та додатній прибуток, що свідчить про ефективність та доцільність їх використання.

Подальші дослідження

- Розробка зручного користувацького інтерфейсу
- Оптимізація та автоматизація роботи програми
- Покращення алгоритму прогнозування статистики (наприклад, шляхом накладання ваг в залежності від типу покриття корту)

Дякую за увагу!