НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.94:793.5

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
___ _____ **202_ р.**	**30 листопада 2021 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Інтелектуальна система прогнозування результатів для букмекерської контори

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма Програмне забезпечення інформаційних систем

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

доц., к.т.н.	_____	Голуб Б.Л.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Керівник магістерської кваліфікаційної роботи

проф., д.т.н.	_____	Бушма О.В.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Виконав

_____	Басов Г.І.
(підпис)	(ПІБ студента)

КИЇВ-2021

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних наук

доц., к.т.н. _____ Б. Л. Голуб
(вчене звання і ступінь) (підпис) (ініціали і прізвище)

«29» жовтня 2020 р.

ЗАВДАННЯ
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

Басову Глібу Ігоровичу
(прізвище, ім'я, по батькові)

Спеціальність 121 «Інженерія Програмного Забезпечення»
Освітня програма «Програмне забезпечення інформаційних систем»
Орієнтація освітньої програми освітньо-професійна

1. Тема магістерської роботи: Інтелектуальна система прогнозування результатів для букмекерської контори затверджена наказом ректора НУБіП від «29» жовтня 2020 р. № 1636 «С»
2. Термін подання завершеної роботи на кафедру 2021 (рік) 11 (місяць) 30 (число)
3. Вихідні дані до магістерської роботи: дані отримано з загальнодоступних сайтів статистики за допомогою спеціалізованого програмного забезпечення.
4. Перелік питань, що підлягають дослідженню:

1	Дослідження предметної області, постановка задачі.	3.02.2021р.
2	Огляд існуючих рішень.	8.04.2021р.
3	Вибір методів дослідження.	1.05.2021р.
4	Проектування системи, її дослідження.	10.06.2021р.
5	Вибір інструментарію та імплементація системи.	22.08.2021р.

Дата видачі завдання «12» листопада 2020 р.

Керівник магістерської роботи _____ проф., д.т.н. Бушма О.В.
(підпис) (вчене звання і ступінь) (прізвище та ініціали)

Завдання ~~прийняв~~ до виконання Басов Г.І.

(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП

На сьогодні, розвиток штучного інтелекту вже є приголомшливим. На даний момент переважній більшості людей навіть важко уявити в яких галузях ще не знайшов використання штучний інтелект. Він використовується в таких галузях як:

- транспорт (скорочення часу поїздки через аналіз трафіку);
- спорт (розумні пристрої);
- мобільні пристрої (голосові асистенти);
- медіа (автоматизований журналізм);
- комп'ютерні ігри (покращена якість графіки);
- комунікації (фільтри спаму);
- соціальні мережі (розпізнання фото, персоналізований потік новин);
- агрокультура (роботизований збирач врожаю, алгоритму прогнозування впливу середовища на сільськогосподарські культури);
- розумні будівлі (технологій "Розумний дім", автоматизована докупка продуктів);
- кібербезпека (аналіз інформаційного трафіку);
- банкова (сегментація клієнтів);
- медична (автономні хірургічні роботи);
- та численних інших.

Одним із найпоширеніших завдань машинного навчання, що передбачає прогнозування цільової змінної за раніше небаченими даними, є класифікація. Метою класифікації є передбачення цільової змінної (класу) шляхом побудови класифікаційної моделі на основі навчального вибірки даних, та подальшого її

використання для прогнозування значення класу для інших, небачених моделлю даних. Цей тип конфігурування штучної моделі називається контрольованим навчанням, оскільки фаза обробки даних спрямована до змінної класу під час побудови моделі. Деякі поширені заявки на класифікацію включають схвалення позики, медичні діагнози, фільтрування електронної пошти або прогнозування.

Прогнозування результатів подій зазвичай трактується як класифікаційна проблема, з прогнозуванням одного класу з кількох (виграти, програти чи розіграти). Хоча деякі дослідники, також розглянули проблему числового прогнозування, де вони прогнозують виграшний запас - числове значення. Для передбачення можна зібрати велику кількість параметрів, включаючи історичну результативність команд, результати матчів та дані про учасників подій, щоб допомогти різним зацікавленим сторонам спрогнозувати шанси на виграшний або програшний перебіг подій у майбутніх матчах. Рішення яка команда ймовірніше виграє є важливим через фінансові активи, що беруть участь у процесі ставок; таким чином букмекери, шанувальники та потенційні учасники ставок зацікавлені у тому, щоб заздалегідь збільшити точність прогнозування шансів гри. Після отримання прогнозованого результату на матч додатковою проблемою є вирішення питання про те, чи робити ставку на матч, враховуючи шанси букмекера. Крім того, менеджери намагаються моделювати відповідні стратегії, які можуть добре працювати для оцінки потенційного фаворита матчу. Тому завданням передбачення результатів є те, що вже давно цікавить різних зацікавлених сторін, у тому числі ЗМІ. Зростаючий обсяг даних, пов'язаних зі спортом, які зараз доступні в електронному вигляді, означають, що зростає й інтерес до розробки штучних інтелектуальних моделей та програмних систем для прогнозування результатів подій.

Об'єктом дослідження було обрано процес прогнозування результатів події.

Предметом дослідження є інтелектуальна система для прогнозування результатів подій представлених букмекерською конторою.

Мета дослідження - необхідно дослідити точність прогнозування результатів подій різними інтелектуальними моделями з ціллю використання найточнішою в модулі прогнозування створюваної системи.

Зміст поставлених завдань - процес розробки та дослідження системи, яка є об’єктом дослідження даною наукової роботи, був поділений на наступні етапи:

1. Провести аналіз та відбір інтелектуальних моделей, які найчастіше використовуються в подібних системах.
2. Реалізувати та натренувати обрані інтелектуальні моделі
3. Провести аналіз метрик порівняння інтелектуальних моделей
4. Реалізувати алгоритм побудови метрики для порівняння інтелектуальних моделей.
5. Проаналізувати отримані результати.
6. Сформулювати висновки, щодо вибору інтелектуальної моделі, яка буде використана в створюваній системі.

Методи дослідження - мова програмування Python, яка є однією з провідних в даному напрямку, була використана для реалізації обраних інтелектуальних моделей, реалізації метрики для порівняння інтелектуальних моделей та її візуалізації, реалізації всіх модулів досліджуваної інтелектуальної системи.

Наукова новизна - було досліджено та створено інтелектуальну модель, яка може бути універсально використана для широкому спектру видів подій представлених букмекерською конторою, починаючи від спортивних подій та закінчуючи прогнозуванням кандидату приймаючого участь у політичних перегонах.

Апробація результатів дослідження:

1. Басов Г.І. Інтелектуальна система прогнозування результатів для букмекерської контори // Збірник матеріалів XII Міжнародна науково-практична конференція молодих вчених «інформаційні технології: економіка, техніка, освіта». – Київ. – 2021 [42].

Структура магістерської роботи

- кількість розділів - 4
- кількість ілюстрацій - 55
- кількість таблиць - 1
- кількість джерел – 42
- кількість формул – 2

1 СИСТЕМНИЙ АНАЛІЗ ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ ДЛЯ БУКМЕКЕРСЬКОЇ КОНТОРИ

1.1 Опис букмекерської контори

1.1.1 Букмекерська контора. В даній роботі досліджується система для букмекерської контори. Тому, перш за все, необхідно розібратися, що таке букмекерська контора. Букмекерська контора – це гральний заклад, що займається прийомом ставок. Букмекер оцінює ймовірність того чи іншого результату події і на кожен з можливих результатів виставляє коефіцієнт – числове значення, на яке множиться ставка в разі успіху. Головний напрямок роботи більшості букмекерських контор – прийом ставок на спортивні події. При цьому кількість пропонованих для ставок видів спорту може досягати 30 і включати в себе не тільки популярні футбол, теніс, баскетбол і хокей, але і щось більше «екзотичне» на кшталт гольфу, дартсу або крикету. Ставками на спорт все не обмежується: деякі букмекерські контори пропонують вгадати результат знакових подій в тому числі зі світу політики і культури.

1.1.2 Види ставок. Для того щоб отримати виграш в букмекерській конторі, необхідно зробити ставку на будь-який з пропонованих результатів, який в підсумку виявиться вірним. Букмекери пропонують різноманітні результати подій і велике різноманіття видів ставок. Так, в футбольному матчі можна зробити ставку на перемогу тієї чи іншої команди (ставки на результат), на сумарну кількість голів в матчі або ж на кількість голів однієї команди (ставки на тотал), на різницю м’ячів (ставки з форою). Тотали очок, геймів і закинутих шайб мають місце в баскетболі, тенісі та хокеї відповідно. В цілому ж коло результатів, доступних для ставок, є досить широким. Короткий перелік існуючих видів ставок:

- антиекспрес;
- володіння м'ячем;

- голи у футболі;
- індивідуальний тотал;
- лайв-ставки;
- нульова фора;
- обидві заб'ють;
- пенальті та видалення;
- прохід;
- ставки на гейми у тенісі;
- ставки на жовті картки;
- ставки на результат;
- ставки на кутові;
- ставки на удари в площину;
- тайм/матч;
- тотал;
- тотал більше;
- тотал менше;
- точний рахунок;
- фора;
- експрес;
- та інші.

Крім того, гравець букмекерської контори може робити ставки на конкретну подію як до його початку (pre-match), так і під час гри (live). У випадку з live-ставками важлива швидкість прийняття рішень, адже котирування змінюються набагато швидше в залежності від ходу матиу, поточного результату і часу до закінчення зустрічі. Сукупність запропонованих подій з усіма наслідками і відповідними їм коефіцієнтами називається букмекерської лінією. Тому більшість сайтів букмекерів мають розділ «лінія», де й розташовані всі можливі ставки букмекера.

1.1.3 Коефіцієнт букмекера. Кожному конкретному результату в лінії букмекерська контора привласнює коефіцієнт, який і визначає розмір передбачуваного виграшу. Розмір виплати, яку гравець отримає від букмекера в разі виграшу ставки, визначити нескладно: досить просто помножити поставлену суму на коефіцієнт. Варто звернути увагу, що розмір виплати і прибуток за ставкою – це не одне і те ж! Щоб визначити прибуток, необхідно від суми виплати відняти розмір своєї ставки – так ми отримаємо суму чистого прибутку, яку принесла виграшна ставка. Наприклад, на матч «Боруссія» – «Шальке» пропонуються наступні котирування: 1.65 – 4.44 – 5.55 (перемога «Боруссії», нічия і перемога «Шальке» відповідно). Якщо ставимо на «Боруссію» 100 $, потенційний прибуток складе 65 $ (100 $ х 1.65 – 100 $ = 65 $). Букмекерський коефіцієнт показує не тільки розмір потенційної виплати гравцеві: крім того, він висловлює передбачувану ймовірність конкретного результату події. Чим нижчий коефіцієнт присвоєно результату, тим вищу ймовірність, за оцінкою букмекера, має цей результат. Так, в наведеному вище прикладі «Боруссія» – «Шальке» ми бачимо, що з-за більш низького коефіцієнта більше шансів на перемогу має «Боруссія». При цьому варто розуміти, що коефіцієнт – це оцінка букмекера, думка, яка не завжди збігається з реальністю. Коефіцієнти букмекерів можуть записуватися в різних форматах в залежності від країни, в якій працює букмекерська контора. В Україні і країнах СНД прийнятий десятковий формат запису коефіцієнтів.

1.1.4 Бонусні програми букмекера. Нерідко букмекери, які працюють в мережі Інтернет (онлайн-букмекери), заохочують своїх клієнтів різними бонусними пропозиціями. Найчастіше це бувають бонуси на перший депозит (привітальні бонуси) або бонуси для безкоштовних ставок. Звернімо увагу: перед тим як погоджуватися на отримання бонусу від букмекера, необхідно уважно ознайомитися з умовами акції! У більшості випадків отримання грошового бонусу «на руки» можливо тільки після розміщення на ставках певної суми. Як це працює? Припустимо, за поповнення ігрового рахунку

(депозит) на суму від 10 $ букмекер обіцяє гравцеві бонус в розмірі 100% від суми депозиту, тобто, додаткові 10 $ на рахунок. Однак умови акції вимагають відіграшу цього бонусу у вигляді, наприклад, п'ятикратного обороту внесених коштів. Це означає, що перш ніж вивести бонусні гроші, необхідно розмістити на ставках не менше 50 $. У разі бонусу для безкоштовної ставки букмекер нараховує на рахунок гравця деяку суму «віртуальних» грошей. Єдиний спосіб використання цієї суми – її розміщення на ставках. При цьому, знову ж таки, варто звертати увагу на умови акції: до безкоштовної ставкою можуть пред'являтися свої вимоги! Якщо безкоштовна ставка виграла, гравець отримує на свій рахунок лише чистий прибуток від неї – сама сума безкоштовної ставки в реальні гроші не конвертується. Як правило, цей прибуток відразу стає доступним для виведення, проте іноді і він вимагає відіграшу.

1.1.5 Основа прибутку букмекерської контори. Зрозуміло, значну частину прибутку букмекерської контори становлять програші гравців. Однак спочатку гарантований заробіток закладається в коефіцієнти. Оцінюючи ймовірність кожної події і виставляючи коефіцієнт на цю подію, букмекер занижує цей коефіцієнт, закладаючи в нього маржу. Ця маржа – і є гарантований заробіток, який принесе прибуток незалежно від результату події. У разі матчу суперників з рівними шансами об'єктивний коефіцієнт повинен становити 2.0 на перемогу обох суперників (50/50), проте таких випадків в лініях букмекерів не зустрінеш – коефіцієнти на рівні шанси завжди будуть нижчими 2.0 через закладеної в них маржі.Для більшої наочності наведемо приклад. Візьмемо тенісний поєдинок Дель Потро – Нисикори. Букмекер пропонує на перемогу обох тенісистів коефіцієнти 1.87 – 1.87, тим самим оцінюючи їх шанси на виграш однаково (50/50). Котирування без урахування маржі в такому випадку були б 2.0 – 2.0, а чистий виграш при ставці в 100 $ склав би 100 $. Однак якщо враховувати маржу, гравець замість цих 100 $ отримує прибуток в розмірі 87 $ (100 $ х 1.87 – 100 $), а що

залишилися 13 $ (100 $ – 87 $) букмекер забирає собі. Це і є маржа букмекера, що забезпечує йому гарантований заробіток.

1.1.6 Офлайн та онлайн букмекери. Букмекерські контори можуть здійснювати свою діяльність в наземних пунктах прийому ставок (ППС), а також в мережі Інтернет (онлайн-букмекери). В пунктах прийому ставок здійснюється готівковий розрахунок, а виплата виграшів проводиться через касу. Деякі компанії називають свої ППС клубами і оформляють їх так, щоб у клієнтів була можливість не тільки робити ставки, але і повноцінно відпочивати, переглядаючи прямі трансляції спортивних подій. Незважаючи на комфорт, який букмекери намагаються забезпечити відвідувачам своїх пунктів прийому ставок, з року в рік зростає популярність онлайн-букмекерів – тих, які приймають ставки через Інтернет. Приймати ставки через Інтернет в Україні має право тільки букмекерська компанія, яка:

- має ліцензію на ведення букмекерської діяльності в Україні;
- входить до складу саморегулівної організації букмекерів;
- підключена до центру обліку переказів інтерактивних ставок.

Всі виграші за ставками, що вчинені на їх сайтах, підлягають судовому захисту.

1.2 Огляд існуючих рішень

Впродовж кількох останніх десятиліть, розвиток технологій та вивчення нейронних мереж надали змогу знайти використання штучного інтелекту й у напрямку букмекерства. На сьогодні, використання нейронних мереж букмекерськими конторами для прогнозування в результатів подій не є чимось незвичайним.

У даній справі існують лише два аспекти від яких найбільше залежать результати дослідження. Першим аспектом є відсутність достатнього обсягу даних для навчання нейронної мережі та другим - апаратне забезпечення, яке

дозволить у допустимо у невеликий проміжок часу навчити мережу на настільки великому обсязі даних. Проте в наш час апаратне забезпечення необхідної потужності вже перестало бути проблемою, тому зосереджую увагу на вибірці даних для навчання. Щодо достатньо великого обсягу даних, то букмекерські контори вже володіють, оскільки вони мають доступ до даних про матчі подій різного рівня, які проводяться по всьому світі. У зв'язку з цим, питання пов'язане з даними вже не є проблемою.

Проте, через те, що програмне забезпечення букмекерських контор не знаходиться в відкритому доступі було відібрано та розібрано три дослідження, які мали за мету прогнозування результатів тенісних матчів турнірів Великого шолому та деяких інших.

Перше робота [1] представляла собою розробку та дослідження тришарової нейронної мережі для прогнозування тенісних матчів із використанням алгоритму зворотного розповсюдження. Автор моделі Sombonphokaphan дослідив та порівняв різні конфігурації мережі за різних наборів вхідних параметрів. Найточніша версія моделі складається із 27 вхідних вузлів, які представляють такі ознаки матчів та гравців як поверхня корту, відсоток виграшу за умови першої подачі, другої подачі, подачі у відповідь, брейк-пойнти тощо. Автор стверджував точність близьку до 75 відсотків при прогнозуванні матчів турнірів Великого шолому у 2007-2008 роках.

Другий вчений [2], а саме, Sipko використовував модель логістичної регресії для її перевірки на вибірці з близько шести тисяч офіційних матчів туриірів ATP 2013-2014 років. ROI найточнішої моделі склав 4.35%. Проте коли вихідний код роботи розібрав інший спеціаліст було виявлено, що результат прогнозувань відповідав 65%, при негативному ROI.

Третім дослідженим джерелом [3] була курсова робота, опублікована випускники MIT Narayanan та Wagner, в якій було описано як вони використовували SVM для прогнозування переможця інтерактивної гри ATP World Tour Draw Challenge, яка проводилася асоціацією тенісистів-

професіоналів до 2014 року. Сенс даної гри полягав у тому, що перед початком світового турніру ATP будь-хто з бажаючих міг спробувати за турнірною таблицею спрогнозувати переможців у всіх майбутніх матчів аж до переможця турніру. Дана модель використовувала 15 ознак, переважно посетову статистику тенісистів. Навчальна вибірка склала близько 40000 прикладів, для тестування моделі використовували перехресну перевірку на шести тисячах прикладів. Максимальна точність, яку змогла набути модель склала 65 відсотків.

1.3 Постановка завдання

Для дослідження даної системи прогнозування результатів для букмекерської контори, основною є модель за допомогою якої будуть прогнозуватися результати. Тому для того, щоб обрати кращу модель були виконані наступні кроки.

1.3.1 Провести аналіз та відбір інтелектуальних моделей, які найчастіше використовуються в подібних системах. На даному кроці було проведено дослідження існуючих рішень в виді робіт різних вчених, кожен з яких досліджували окрему модель та викладав отримані результати.

1.3.2 Реалізувати та натренувати обрані інтелектуальні моделі. На даному кроці було реалізовано власну інтелектуальну модель та ініціалізовано 3 інших моделі, що дозволило надалі натренувати кожну з них.

1.3.3 Реалізувати алгоритм побудови матриці невідповідностей для реалізованих моделей. Даний крок, був представлений розробкою алгоритму, який дозволяє на основі натренованої інлетектуальної моделі побудувати матрицю невідповідностей, щоб надалі можна було порівняти кілька моделей на основі відповідних метрик.

1.3.4 Проаналізувати отримані результати. На даному етапі з побудувавши матриці невідповідностей для кожної з досліджуваних моделей, було вирахувано відповідні метрики для їх порівняння та на основі цих метрик було обрану найкращу з досліджуваних моделей.

1.4 Моделі предметної області

Use-case (прецедент) – описує певну послідовність виконуваних системою дій, що генерують певний результат, значущий для відповідного актора. Тобто метою використання прецеденту є структуризація сутності поведінки моделі. «Що зробити?» - це питання на яке відповідає прецедент, скриваючи засоби дії.

Актор – безліч речей, що взаємодіючи виконуються прецедентом. Зазвичай, роль актора виконується людиною або певним апаратним чи програмним модулем, які безпосередньо пов'язані з системою.

Прецедент – випадок використання, дія. Позначення - овал.

Межі системи – перекривають усі випадки використання в системі. Позначення – прямокутник.

Діаграма прецедентів налічує такі елементи взаємодії:

- Використовує – виконання певної дії користувачем.
- Розширення – метод для відображення дочірніх прецедентів.
- Вимагає – необхідність прецеденту в виконанні попереднього прецеденту.
- Включає – використання одним прецедентом іншого.
- Рівнозначний – схожий функціонал, про користувач розуміє, як різний.

Рис. 1.1 Основні елементи діаграми прецедентів

Для прикладу, в розроблюваній системі роль актора виконує користувач системи

Побудова діаграми прецедентів базується на виявленні примітивних дій, які внаслідок співставляються з їх прецедентами

Діаграма прецедентів подається у вигляді прецедентів – що зображені еліпсом, в одно час, актор представляється у вигляді піктограми людини.

На діаграмі представлено 7 прецедентів кожний з яких відповідає за:

1) Розміщення нової події букмекером;
2) Створення бонусної програми букмекером;
3) Корегування коефіцієнтів події букмекером;
4) Внесення депозиту клієнтом;
5) Реєстрація ставки клієнтом;
6) Розрахунок виграшу клієента букмекером;
7) Отримання виграшу клієнтом.

Діаграму прецедентів букмекерської контори можна переглянути на рис.1.2.

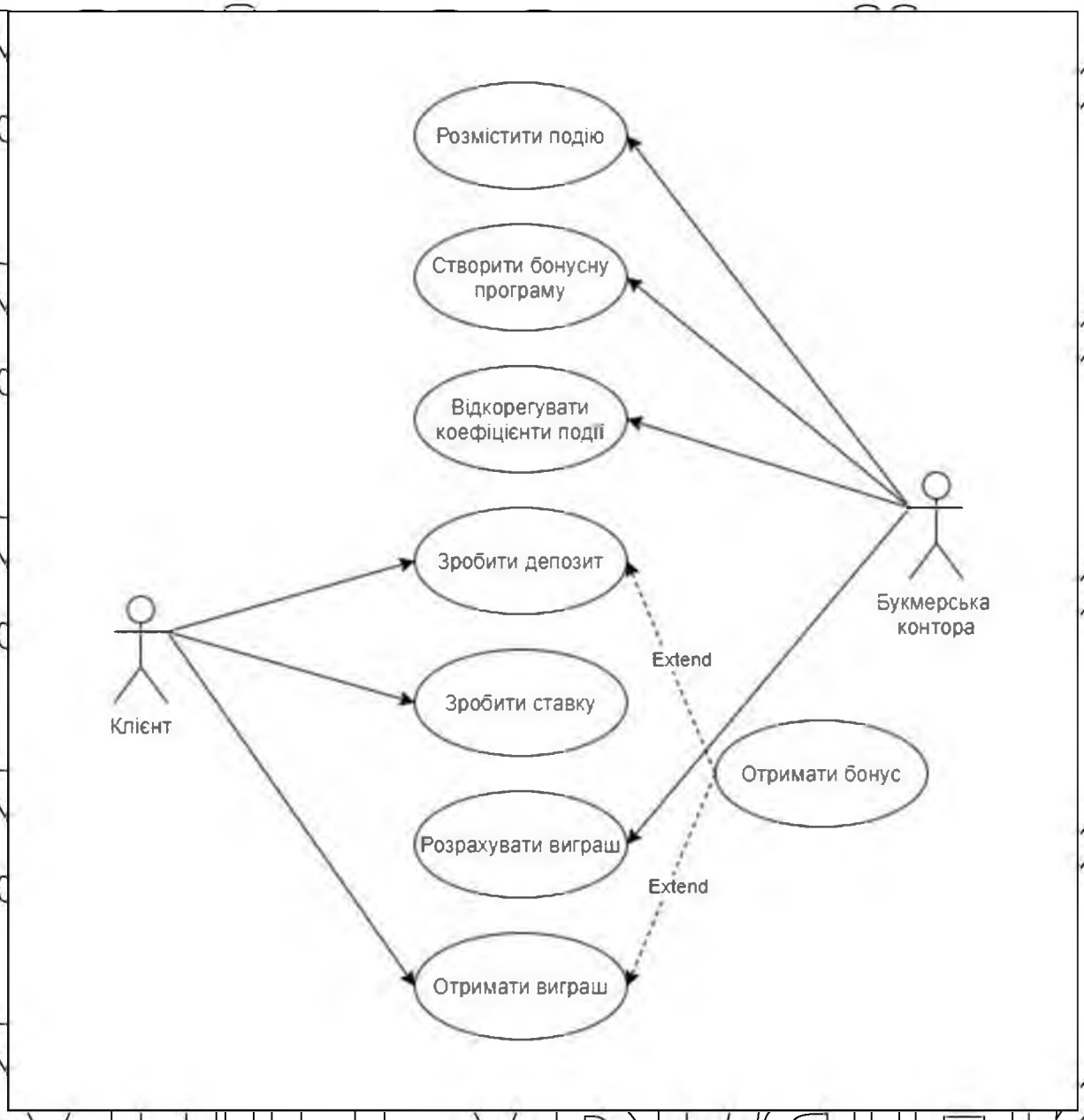

Рис. 1.2 Діаграма прецедентів букмекерської контори

2 ТЕОРЕТИЧНІ ПЕРЕДУМОВИ

2.1 Штучний інтелект як технологічні и наукові рішення

На даний момент, лиш малий відсоток представників сфери інформаційних технологій чітко розуміють різницю між такими речами як штучний інтелект, машинне навчання, нейронні мережі та близькі терміни даного наукового напрямку, у зв'язку з, цим вважаю за потрібно уділити увагу їх пояснення.

Насамперед, штучний інтелект (artificial intelligence) – представляє собою різні технологічні та наукові рішення та методи, які допомагають зробити програми подібні до інтелекту людини. Штучний інтелект включає безліч інструментів, алгоритмів і систем, серед яких також всі складові data science і machine learning.

2.2 Машинне навчання

Машинне навчання (machine learning) ж це один із розділів штучного інтелекту, алгоритми, що дозволяють комп'ютеру робити висновки на підставі даних, не дотримуючись жорстко заданих правил. Тобто машина може знайти закономірність у складних і багато параметричних завданнях (які мозок людини не може вирішити), таким чином знаходячи точніші відповіді. Як результат – правильне прогнозування. Види машинного навчання можна розрізняти за ознакою наявності вчителя та за типом алгоритмів, що застосовуються. За ознакою наявності вчителя, навчання ділиться навчання з вчителем (Supervised Learning), без вчителя (Unsupervised Learning) і з підкріпленням (Reinforcement Learning).

- навчання з учителем застосовують, коли потрібно навчити машину розпізнавати об'єкти чи сигнали. Загальний принцип навчання з учителем це "дивися, ось це двері і це теж двері, і це теж двері".

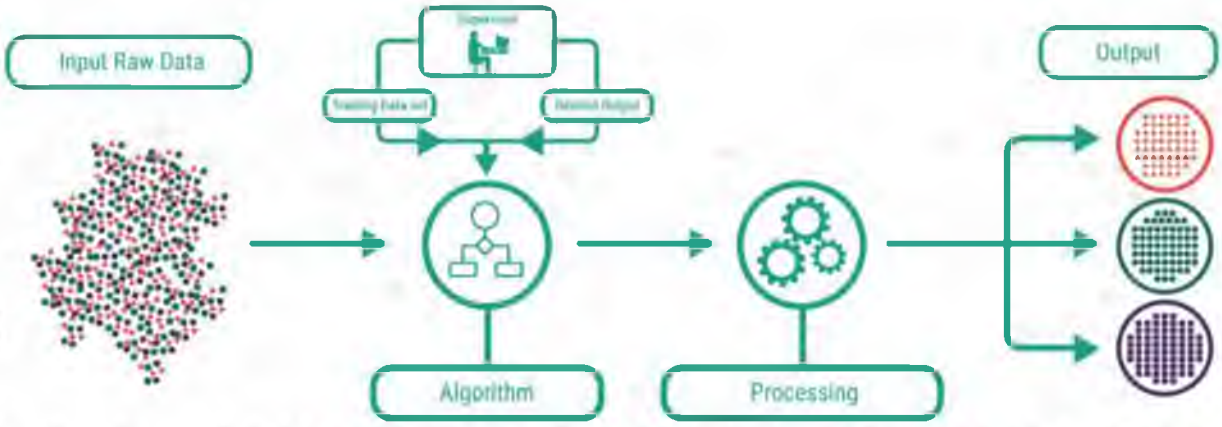

Рис. 2.1 Схематичне зображення процесу навчання з учителем

- навчання без вчителя використовує принцип “ця річ така сама, як інші”. Алгоритми вивчають подібності і можуть виявити відмінність і виявити аномалії, розпізнаючи, що незвичайним чи несхожим.

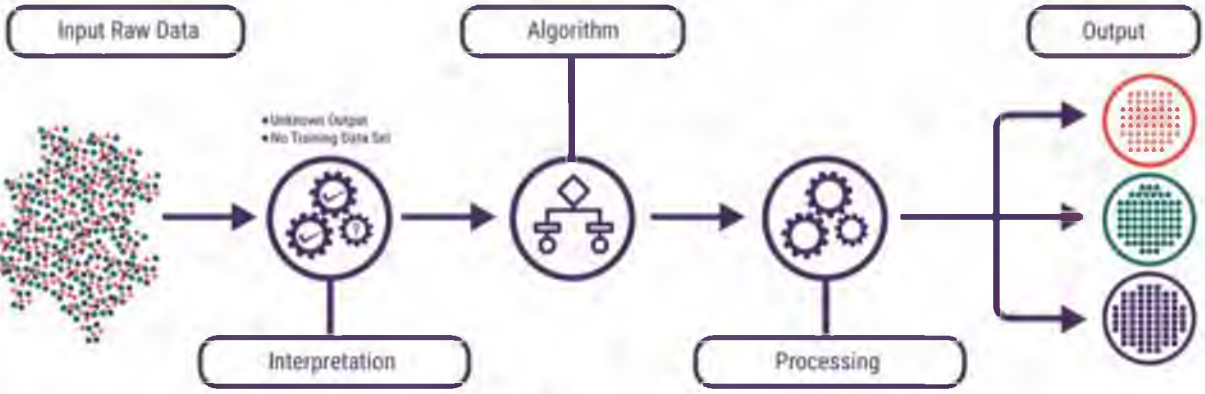

Рис. 2.2 Схематичне зображення процесу навчання без учителя

- навчання з підкріпленням використовують там, де перед машиною стоїть завдання – правильно виконати поставлені завдання у зовнішньому середовищі, маючи безліч можливих варіантів дії. Наприклад, у комп'ютерних іграх, трейдингових операціях для безпілотної техніки.

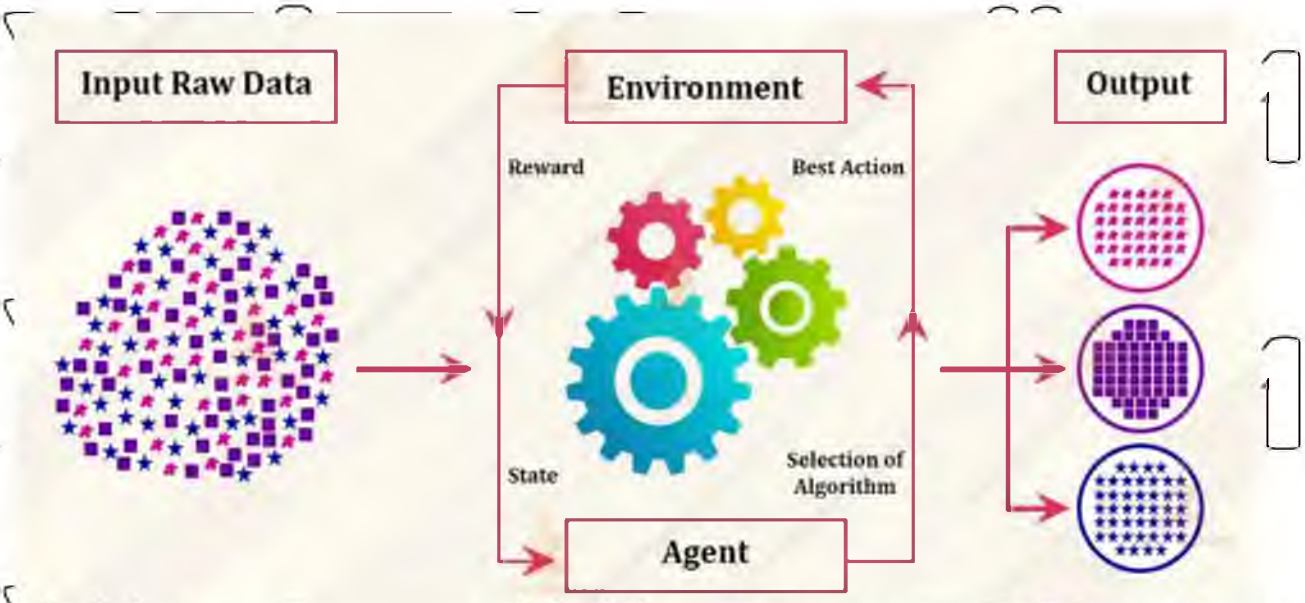

Рис. 2.3 Схематичне зображення процесу навчання з підкріпленням

За типом алгоритмів, що застосовуються, можна виділити два види:

- класичне навчання – відомі та добре вивчені алгоритми навчання, розроблені переважно більш ніж 50 років тому. Основна розділ застосування це завдання з даними: класифікація, кластеризація, регресія тощо. Застосовують для прогнозування, сегментації клієнтів тощо.
- нейронні мережі та глибоке навчання – найсучасніший підхід до машинного навчання. Нейронні мережі застосовуються там, де потрібні розпізнавання чи генерація зображень і відео, складні алгоритми управління чи прийняття рішень, машинний переклад та подібні складні завдання.

2.3 Нейронні мережі

Тепер розглянемо як працюють нейроні мережі. Нейронна мережа (neural network) або штучна нейронна мережа (artificial neural network) влаштована так, що вона за допомогою штучних нейронів моделює роботу людського мозку (нейронів), що вирішує певне завдання, самонавчається з урахуванням

попереднього досвіду. І з кожним разом робить все менше помилок. За конфігураціями штучні нейронні мережі бувають надзвичайно різноманітні. Незважаючи на це, мережеві парадигми мають багато спільного. Нейронні мережі розрізняють за топологічними типами відповідно до структури звязків між нейронами мережі, а також за типом використаних формальних нейронів. Штучні нейронні мережі можуть розглядатися як спрямований граф зі зваженими зв'язками, у якому штучні нейрони є вузлами. По архітектурі зв'язків штучні нейронні мережі можуть бути згруповані в два класи (рис. 2.4):

- мережі прямого поширення, у яких графи не мають петель
- рекурентні мереж, або мережі зі зворотними зв'язками.

Рис. 2.4 Класи нейронних мереж за архітектурою зв'язків

Також важливо пам'ятати, що нейромережі є одним із видів машинного навчання, а не окремим інструментом. Основуючись на цьому ми вже можемо зрозумити ієрархічну залежність між даними термінами представлену на рис. 2.5.

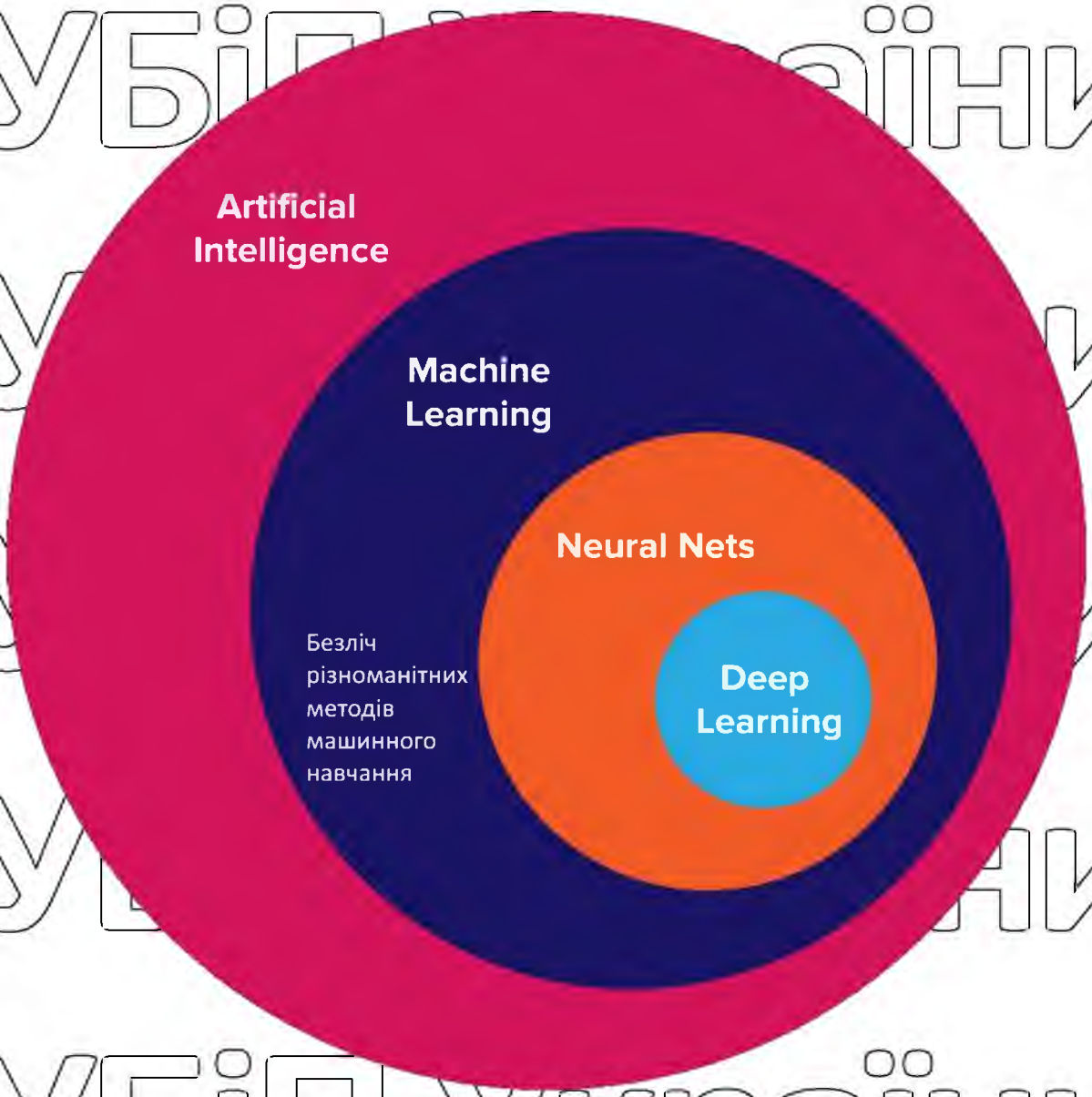

Рис. 2.5 Ієрархія розділів машинного навчання

Наступним кроком, необхідно розглянути такі терміни як логістична регресія, метод опорних векторів та персептрон, оскільки дані моделі будуть використовуватися та порівнюватися в даній науковій роботі.

2.4 Логістична регресія

Логістична регресія - це модель машинного навчання, що використовується для вирішення завдань класифікації. Для прикладу кілька прикладів класифікаційних завдань машинного навчання:

- спам електронної пошти (спам чи спам?);
- претензія щодо страховки автомобіля (виплата компенсації чи ремонт?);
- діагностика хвороби.

Кожне з цих завдань має чітко дві категорії, що робить їх прикладами задач двійкової класифікації. Логістична регресія добре підходить для вирішення задач двійкової класифікації – ми просто призначаємо різним категоріям значення 0 та 1 відповідно. Таким чином, при створенні регресійної моделі, кінцевим продуктом є рівняння, за допомогою якого можна передбачити до якої категорії належить певна точка даних. Схематичне зображення логістичної регресії можна побачити на рис. 2.6.

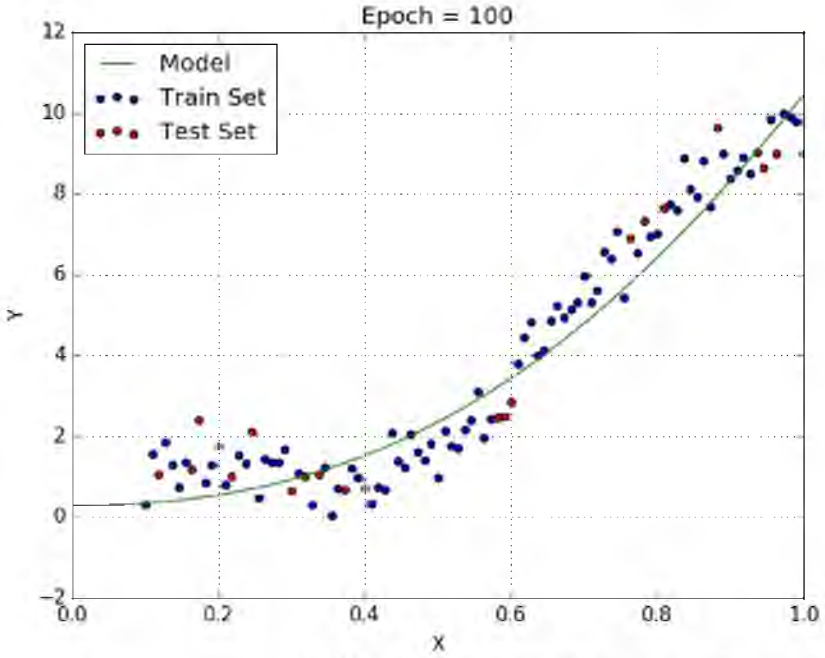

Рис. 2.6 Схематичне зображення логістичної регресії

2.5 Метод опорних векторів

Далі розглянемо метод опорних векторів. Метод опорних векторів - це модель машинного навчання з учителем, з відповідними алгоритмами

навчання, які аналізують дані та розпізнають закономірності. Метод опорних векторів можна використовувати як для завдань класифікації, так і для регресійного аналізу.

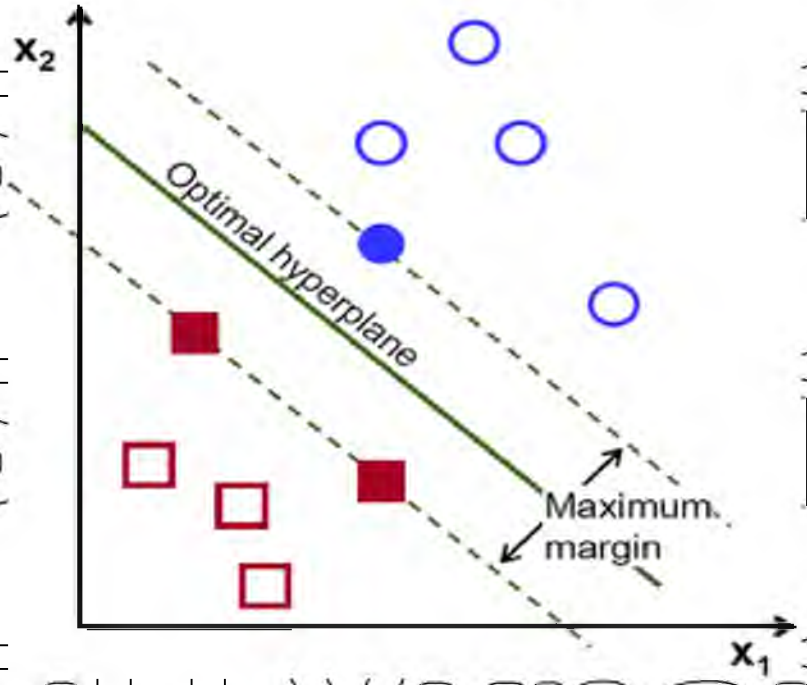

Рис. 2.7 Візуальне представлення методу опорних векторів

На рис. 2.7 Зображена візуальне представлення методу опорних векторів. На цій діаграмі гіперплощина позначена як "оптимальна гіперплощина" (optimal hyperplane). Теорія методу опорних векторів дає таке визначення оптимальної гіперплощини - це гіперплощина, яка максимізує поле між двома найближчими точками даних різних категорій. Можна бачити, що межа поля дійсно зачіпає 3 точки даних — 2 із червоної категорії та 1 із синьої. Ці точки, які стикаються з кордоном поля, і називаються опорними векторами — звідки й пішла назва.

2.6 Багатошаровий персептрон

Останньою моделлю є багатошаровий персептрон. Багатошаровий персептрон - це клас штучних нейронних мереж прямого поширення, що

складаються як мінімум із трьох шарів: вхідного, прихованого та вихідного. Крім вхідних, всі нейрони використовує нелінійну функцію активації. Під час навчання багатошарового персептрона використовується навчання з учителем та алгоритм зворотного розповсюдження помилки. Як активаційні функції нейронів використовуються сигмоїдні: логістична або гіперболічний тангенс. Багатошарові персептрони показали можливість знаходити наближені рішення для надзвичайно складних завдань. Зокрема, вони є універсальним апроксиматором функцій, тому успішно використовуються у побудові регресійних моделей. Оскільки класифікацію можна як окремий випадок регресії, коли вихідна змінна категоріальна, на основі багатошарового персептрона можна будувати класифікатори. Вперше багатошаровий персептрон було запропоновано Ф. Розенлаттом. Однак у тому вигляді, в якому він використовується в даний час, багатошаровий персептрон розроблено Д. Румельхартом. Персептрон Румельхарта відрізняється від персептрона Розенлатта за такими властивостями:

- використання нелінійної активаційної функції;
- число прихованих шарів більше одного (зазвичай трохи більше трьох);
- вхідні сигнали не бінарні, а кодуються десятковими числами, нормованими до інтервалу [0,1];
- навчання проводиться не до мінімізації помилки, а до стабілізації ваги мережі, що дозволяє уникнути перенавчання.

На рис. 2.8 Можна побачити структуру багатошарового персептрона Румельхарта.

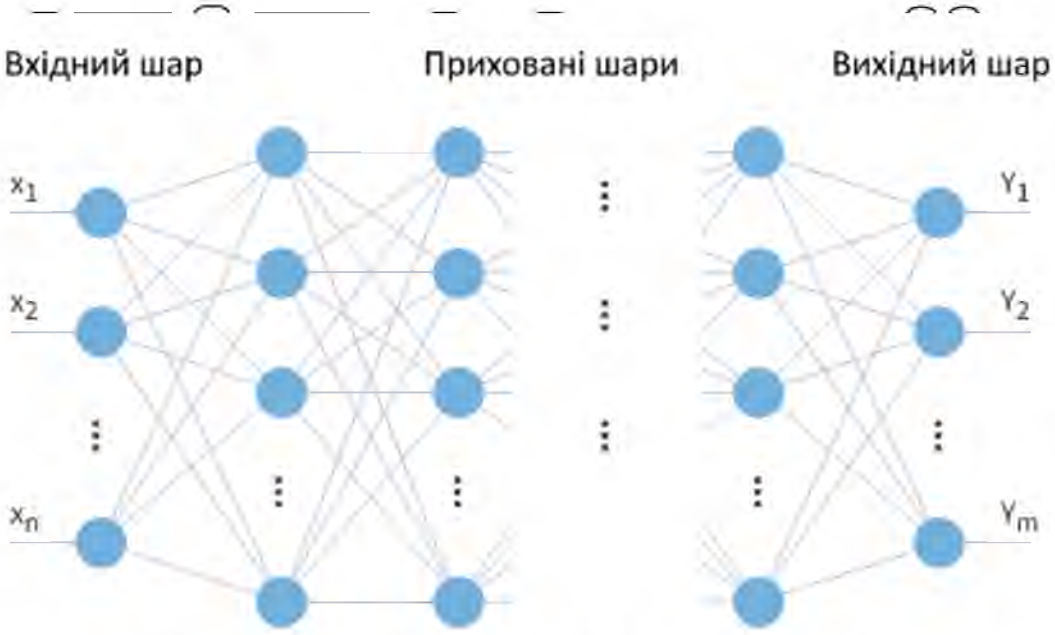

Рис. 2.8 Багатошаровий персептрон Румельхарта

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Проектування загальної архітектури програмної системи

У даному дипломному проекті розроблювалась система за архітектурою найближчою до якої є архітектура клієнт-сервер.

До основних елементів діаграми розгортання відносяться:

- компонент;
- екземпляр компонента;
- інтерфейс;
- вузол;
- екземпляр вузла;
- об’єкт;
- активний об`єкт;
- залежність ;
- зв’язок;
- момент згину зв’язків;
- коментар;
- конектор коментаря.

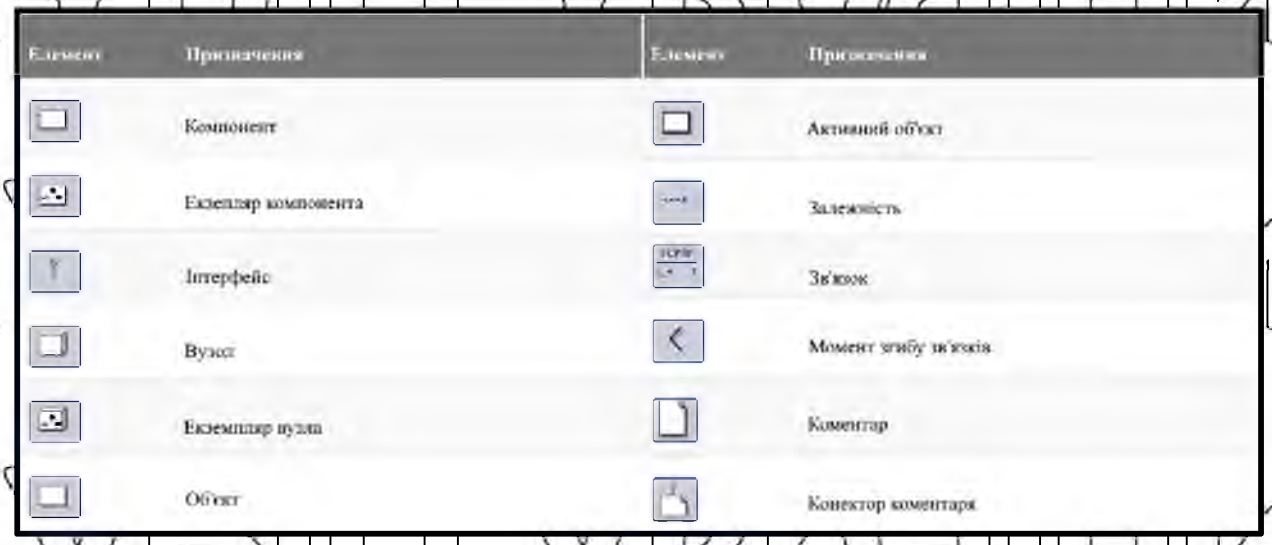

Рис. 3.1 Основні елементи діаграми розгортання

Побачити діаграму розгортання розроблюваної системи можна на рис. 3.2.

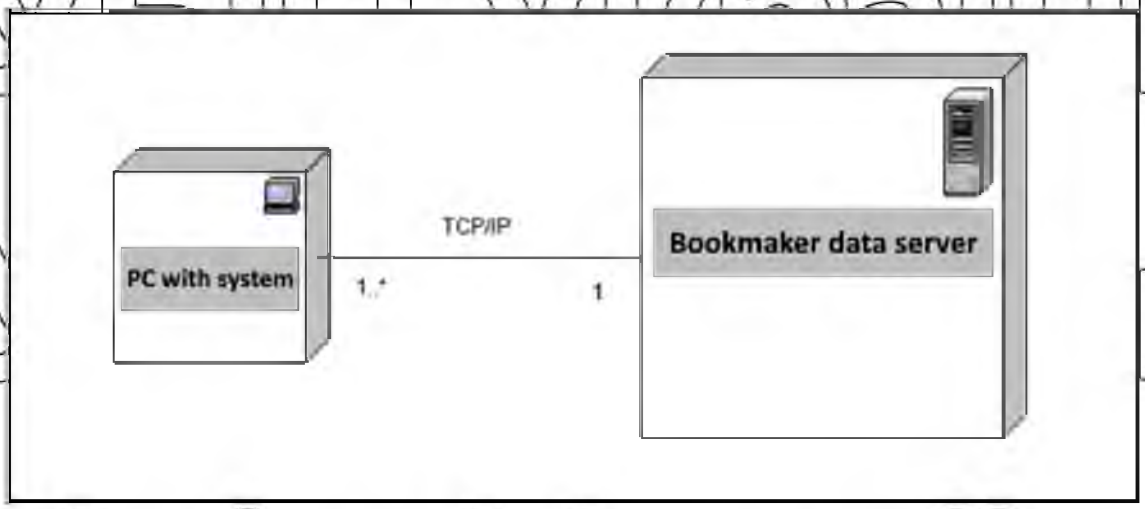

Рис. 3.2 Діаграма розгортання

Даний тип архітектури, а саме клієнт-серверний, набув популярності у зв'язку з стрімким розвитком мережі Інтернет та накопиченням значної частини інформації в базах даних на серверах.

Клієнт-серверну архітектуру можна означити, як концепцію інформаційної мережі в якій переважна частина її ресурсів зосереджена на серверах, які займаються обслуговуванням власних клієнтів. Дана архітектура виділяє такі типи компонентів:

- мережа, яка забезпечує взаємодію клієнтів і серверів;
- клієнти, які користуються сервісами, які надаються серверами;
- сервери, які надають доступ до інформаційного ресурсу та інших послуг програмам, які звертаються до них.

Підсумовуючи, робимо висновок, що розглянута архітектура є найбільш підходящою для розроблюваного проекту, адже переважна частина розроблюваного ПЗ є клієнтом серверу сайту, який надає інформацію, необхідну програмі для коректного функціонування.

3.2 Організаційна структура програмної системи

3.2.1 Інтерфейсна частина програмної системи. Елементи інтерфейсної частини можна побачити в табл. 3.1.

Таблиця 3.1

Елементи інтерфейсної частини додатку

Назва елементу	Опис елементу
Таблиця виводу результатів	Таблиця для візуалізації переліку спрогнозованих результатів live-матчів (рис. 3.3).
Текстове поле інформаційне	Поле відображення виняткових станів роботи додатку (рис. 3.4).
Текстове поле інформаційно-часове	Поле перегляду часу останнього аналізу live-матчів (рис. 3.5).

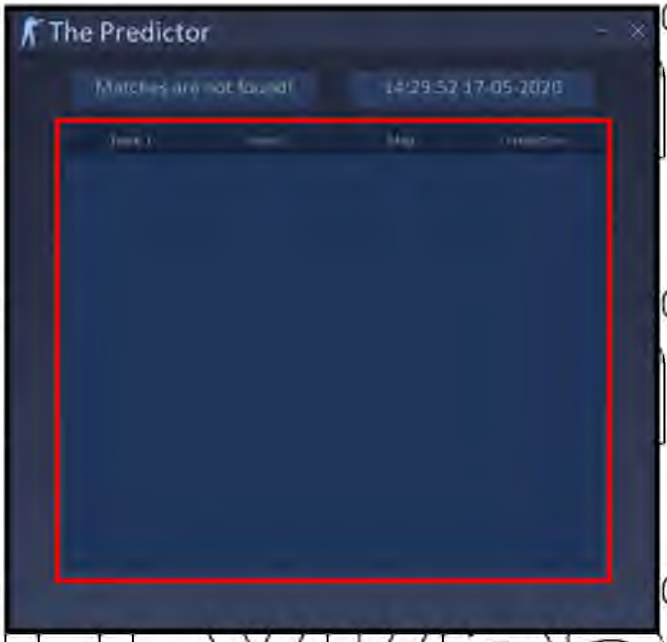

Рис. 3.3 Таблиця результатів

Рис. 3.4 Поле винятковних станів

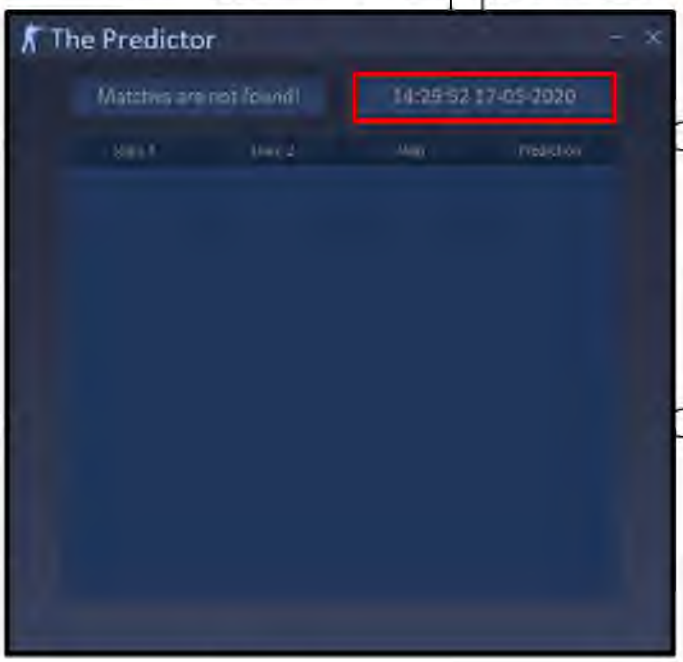

Рис. 3.5 Поле інформаційно-часове

3.2.2 Нейронномережева частина програмної системи. Чи не найважливішу роль у програмному продукті виконує нейронна мережа, даному випадку була використана нейронна мережа за структурою повнозв'язана, за архітектурою чотиришарова мережа прямого поширення, всі зв'язки проективні та аферентні, оскільки нейрони прихованих шарів не пов'язані в межах одного шару та скеровані від вхідних прошарків до вихідних. Нейронна мережа навчалася з вчителем (контрольоване навчання)

методом зворотного поширення помилки. Типом передатної функції була використана сигмоїдальна.

3.3 Розробка та тренування нейронної мережі

Хоча у наш час штучний інтелект широко розвинувся в великій кількості сфер людської діяльності, проте не кожний чітко розуміє різнищя між такими базовими поняттями даної сфери як, наприклад, штучний інтелект та глибоке навчання.

3.3.1 Формування набору даних для навчання та тренування мережі. Однією із проблем із якими зустрічається розробник нейромережі є нестаток даних. Тому першим, що було зроблено було обрано відкритий ресурс представлений у вигляді сайту зберігаючого статистику всіх проведених офіційних матчів із обраної дисципліни.

Наступним кроком було обрано перелік параметрів, які в більшій мірі впливають на результат матчу. В перелік увійшли параметри, які демонструють стан команд-учасників або параметри саме певного матчу (зустрічі).

Далі було розроблено алгоритм парсера, для створення якого було обрано наступні фреймворки мови Python:

- Requests [9] – виконання http-запитів до серверу букмекера

(на рис. 3.6 можна побачити елемент коду, де було використано функціонал даного фреймворку для отримання результату http-запиту);

- BeatifulSoup [10] – семантичний аналіз html

(на рис. 3.7 можна побачити функцію, де було використано функціонал даного фреймворку для виокремлення параметрів із отриманого в запиті html);

- SQLite – збереження зібраних параметрів до таблиці бази даних

(на рис. 3.8 можна побачити елемент коду, де продемонстровано SQL-запит на додавання записів до таблиці).

```
# Http-request for getting page with parameters
request = requests.get(matches_link).content
```

Рис. 3.6 Ділянка коду для HTTP-запитів

```
def get_active_map_pool():
    map_pool = []

    queue = requests.get('https://liquipedia.net/counterstrike/Portal:Maps').content
    page = BeautifulSoup(str(queue), 'html.parser')

    for m in page.find('table', {'class': 'navbox navigation-not-searchable'}). \
            findAll('table', {'class': 'nowraplinks navbox-subgroup wiki-backgroundcolor-light'}):
        if 'Active Duty' in m.text:
            for tr in m.find('tbody').findAll('tr'):
                if 'Active Duty' in tr.text:
                    for a in tr.find('td', {'class': 'navbox-list navbox-odd'}).findAll('a'):
                        if a.text == 'Dust II':
                            map_pool.append('Dust2')
                        else:
                            map_pool.append(a.text)
    return map_pool
```

Рис. 3.7 Елемент коду семантичного аналізу html

```
conn = sqlite3.connect(project_dir + '\\databases\\' + yesterday + '.db')
cursor = conn.cursor()
cursor.executemany("INSERT INTO matches(team1,team1world_rank,team1in_top,team1average_age,"
                   "                    team1players_stats_1,team1players_stats_2,team1players_stats_3,"
                   "                    team1players_stats_4,team1players_stats_5,team1players_stats_6,"
                   "                    team1players_stats_7,team1players_stats_8,team1players_stats_9,"
                   "                    team1players_stats_10,"
                   "                    team2,team2world_rank,team2in_top,team2average_age,"
                   "                    team2players_stats_1, team2players_stats_2,team2players_stats_3,"
                   "                    team2players_stats_4,team2players_stats_5, team2players_stats_6,"
                   "                    team2players_stats_7,team2players_stats_8,team2players_stats_9,"
                   "                    team2players_stats_10, "
                   "                    map_name, "
                   "                    team1_score, team1_rates,"
                   "                    team2_score, team2_rates,"
                   "                    stars) "
                   "             VALUES (?,?,?,?,?,?,?,?,?,?,"
                   "                     ?,?,?,?,?,?,?,?,?,?,"
                   "                     ?,?,?,?,?,?,?,?,?,?,?,?,?)", matches)
conn.commit()
conn.close()
```

Рис. 3.8 Елемент додавання запису до таблиці

3.3.2 Створення структури нейронної мережі. Відповідно до обраної, вході аналізу існуючих рішень, архітектури було розроблено клас мовою Python. Розроблений клас містить в собі:

- __init__ - метод-конструктор, ініціалізуючий структуру основні поля класу, а саме кількості нейронів у відповідних шарах (вхідному, прихованих, вихідному), коефіцієнт швидкості навчання мережі, функція

активації нейрону та ваги відповідних з'єднань між нейронами (реалізацію функції можна побачити на рис. 3.9);

- train_net – метод, який відповідає за навчання нейронної мережі або, інакше кажучи, поширення сигналу через шари нейронної мережі (реалізацію функції можна побачити на рис. 3.10);
- ask_net – метод, який відповідає за прогнозування результату (реалізацію функції можна побачити на рис. 3.11);

```
def __init__(self, inputnodes, hiddennodes1, hiddennodes2, outputnodes, learningrate):
    self.inodes = inputnodes
    self.hnodes1 = hiddennodes1
    self.hnodes2 = hiddennodes2
    self.onodes = outputnodes
    self.lr = learningrate
    self.activation_func = lambda x: scipy.special.expit(x)

    self.wih1 = numpy.random.normal(0.0, pow(self.hnodes1, -0.5), (self.hnodes1, self.inodes))
    self.wh1h2 = numpy.random.normal(0.0, pow(self.hnodes2, -0.5), (self.hnodes2, self.hnodes1))
    self.wh2o = numpy.random.normal(0.0, pow(self.onodes, -0.5), (self.onodes, self.hnodes2))
```

Рис. 3.9 Код init-методу класу нейронної мережі

```
def train_net(self, inputs_list, targets_list):
    inputs = numpy.array(inputs_list, ndmin=2).T
    targets = numpy.array(targets_list, ndmin=2).T

    hidden_inputs1 = numpy.dot(self.wih1, inputs)
    hidden_outputs1 = self.activation_func(hidden_inputs1)

    hidden_inputs2 = numpy.dot(self.wh1h2, hidden_outputs1)
    hidden_outputs2 = self.activation_func(hidden_inputs2)

    final_inputs = numpy.dot(self.wh2o, hidden_outputs2)
    final_outputs = self.activation_func(final_inputs)

    output_errors = targets - final_outputs
    hidden_errors2 = numpy.dot(self.wh2o.T, output_errors)
    hidden_errors1 = numpy.dot(self.wh1h2.T, hidden_errors2)

    self.wh2o += self.lr * numpy.dot((output_errors * final_outputs * (1.0 - final_outputs)),
                                     numpy.transpose(hidden_outputs2))
    self.wh1h2 += self.lr * numpy.dot((hidden_errors2 * hidden_outputs2 * (1.0 - hidden_outputs2)),
                                      numpy.transpose(hidden_outputs1))
    self.wih1 += self.lr * numpy.dot((hidden_errors1 * hidden_outputs1 * (1.0 - hidden_outputs1)),
                                     numpy.transpose(inputs))
```

Рис. 3.10 Код train_net-метод класу нейронної мережі

```
def ask_net(self, inputs_list):
    inputs = numpy.array(inputs_list, ndmin=2).T

    hidden_inputs1 = numpy.dot(self.wih1, inputs)
    hidden_outputs1 = self.activation_func(hidden_inputs1)

    hidden_inputs2 = numpy.dot(self.wh1h2, hidden_outputs1)
    hidden_outputs2 = self.activation_func(hidden_inputs2)

    final_inputs = numpy.dot(self.wh2o, hidden_outputs2)
    final_outputs = self.activation_func(final_inputs)

    return final_outputs
```

Рис. 3.11 Код ask_net-методу класу нейронної мережі

3.3.3 Тренування та тестування нейронної мережі. Тренування нейронної мережі це по суті процес вибору конфігурації (моделі) нейронної мережі із переліку можливих, що зводить похибку до мінімуму.

На рис. 3.12 можна побачити представлення напрямів діяльності людини, для яких використовується кожна з парадигм навчання нейронних мереж.

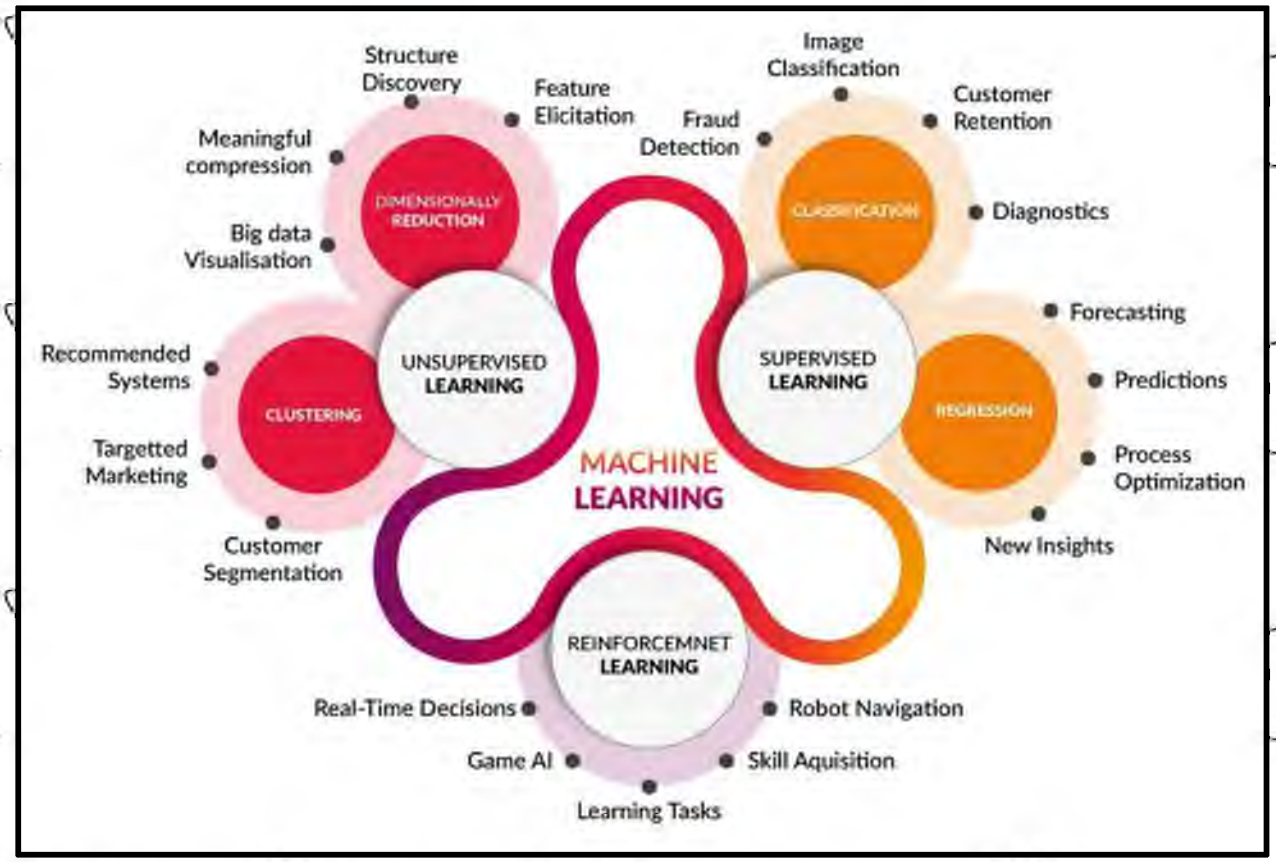

Рис. 3.12 Напрями використання машинного навчання

Проте для зведення значення хибності прогнозу з використанням нейронної мережі з архітектурою багатошарового перцептрону за допомогою

градієнтного спуску алгоритмом зворотного поширення використовується парадигма керованого навчання.

Відповідно до парадигми навчання кожний запис як тренувальному, так і у тестовому наборах даних мають два поля різниця між якими надає змогу їх маркувати. Таким чином нейронна мережа має класифікувати всі запис лише в два підкласи «переможе перший команда» або «переможе друга команда».

В процесі тренування було випробувано різноманітні конфігурації мережі:

- із різною кількістю прихованих шарів;
- із різною кількістю нейронів на кожному з прихованих шарів;
- із різною кількістю епох, інакше кажучи, кількістю циклу проходження навчання на навчальному наборі даних;
- із різною кількістю вхідних нейронів (параметрів);
- із різним коефіцієнтом швидкості навчання.

На рис. 3.13 можна побачити перелік основних параметрів від яких залежить вихідна точність нейронної мережі.

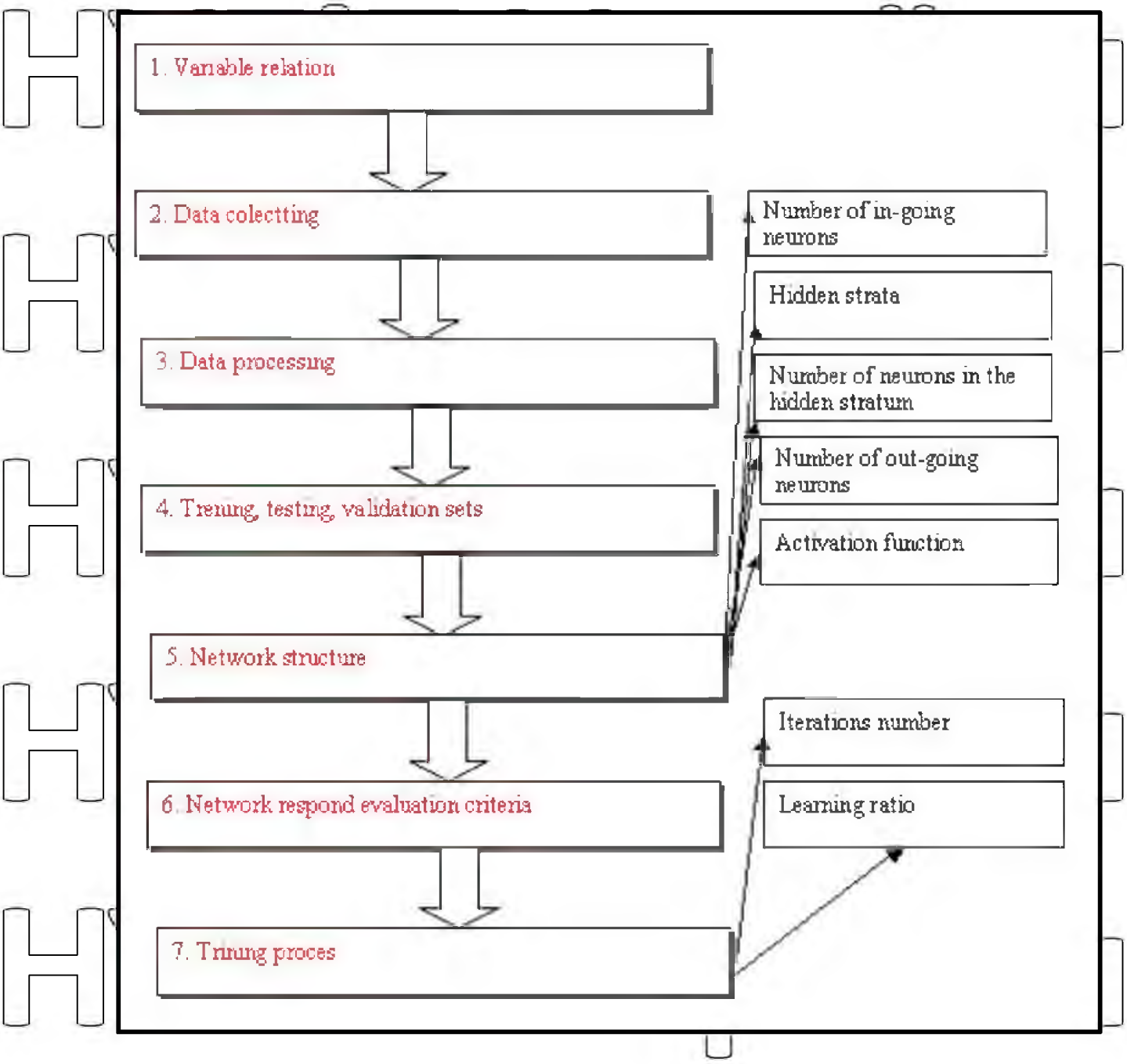

Рис. 3.13 Параметри, від яких залежить точність нейронної мережі

В результаті навчання було досягнуто відсотку точного прогнозування зі значенням близьким до 77% на навчальному наборі та 76,9% на тестовому наборі. Результати випробувань на навчальному наборі відповідного до різної кількості епох можна побачити на рис. 3.14.

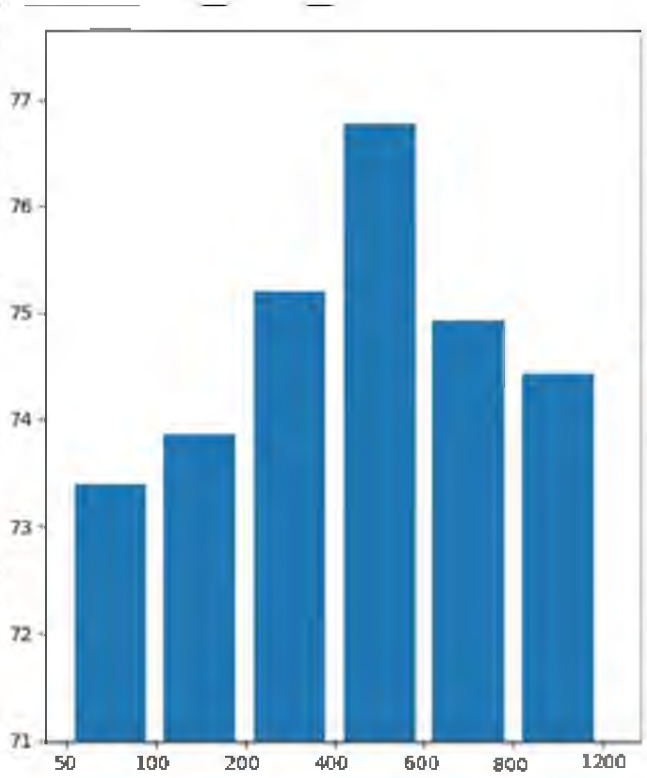

Рис. 3.14 Результати випробувань мережі

3.4. Ініціалізація та тренування моделі логістичної регресії

3.4.1 Створення структури моделі. В даному розділі буде розглянуто ініціалізацію моделі логістичної регресії, одну з моделей обраних у вході аналізу існуючих рішень. Для ініціалізації моделі було використано клас LogisticRegression з бібліотеки scikit-learn [17] мови Python. Були використані такі методи класу як:

- __init__ - метод-конструктор, ініціалізуючий структуру моделі на основі переданих параметрів або ж їх значень за замовченням
- dual (подвійне або первинне формулювання);
- tol (допуск критеріїв зупинки);
- C (зворотна сила регуляризації);
- fit_intercept (визначає, чи слід додати константу (яка же зміщення або перехоплення) до функції прийняття рішення);

- class_weight (вагові показники);
- solver (алгоритм для використання в задачі оптимізації);
- та деякі інші.

- fit– метод, який відповідає за навчання нейронної мережі;
- score – метод, який відповідає за прогнозування результатів.

3.4.2 Тренування та тестування нейронної мережі. Тренування нейронної мережі це по суті процес вибору конфігурації (моделі) нейронної мережі із переліку можливих, що зводить похибку до мінімуму. Тому в процесі тренування мережі було випробувано та оцінено безліч варіантів налаштувань мережі та обрано найкращий. Далі на рис. 3.15 можна побачити функцію в якій ініціалізується модель (рядок 225), отримуються вибірка даних (рядки 227-229), проводиться розподіл даних 9 до 1 для тренувального та випробувальних наборів відповідно (рядки 231-237), тренування мережі (рядок 239) та випробування мережі на навчальному та випробувальному наборах даних.

```
224  def main():
225      lr = LogisticRegression(C=0.1)
226
227      data = get_data_from_dir()
228      data = remove_stars_feature_from_dataset(data)
229      data = create_new_features_from_old_features_2(data)
230
231      df = pd.DataFrame(data)
232      train_data, test_data = train_test_split(df, test_size=0.1)
233
234      train_features = train_data.iloc[:, :-1]
235      train_targets = train_data.iloc[:, -1:].values.ravel()
236      test_features = test_data.iloc[:, :-1]
237      test_targets = test_data.iloc[:, -1:].values.ravel()
238
239      lr.fit(train_features, train_targets)
240
241      print('Score (train):', lr.score(train_features, train_targets))
242      print('Score (test):', lr.score(test_features, test_targets))
```

Рис. 3.15 Код функції тренування та тестування моделі логістичної регресії

В результаті навчання було досягнуто відсотку точного прогнозування зі значення близьким до 76,8% на навчальному наборі та 76,4% на тестовому наборі. Результати випробувань даної моделі можна побачити на рис. 3.16.

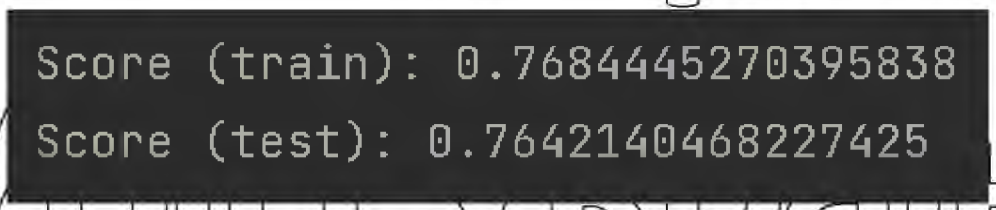

Рис. 3.16 Результати випробувань модель логістичної регресії

3.5 Ініціалізація та тренування моделі методу опорних векторів

3.5.1 Створення структури моделі. В даному розділі буде розглянуто ініціалізацію моделі методу опорних векторів, одну з моделей обраних у вході аналізу існуючих рішень. Для ініціалізації моделі було використано клас LinearSVC з бібліотеки scikit-learn [17] мови Python. Були використані такі методи класу як:

- __init__ - метод-конструктор, ініціалізуючий структуру моделі на основі переданих параметрів або ж їх значень за замовченням:
  - lose (визначає функцію втрат);
  - penalty (визначає норму, яка використовується в «покаранні»);
  - dual (подвійне або первинне формулювання);
  - tol (допуск критеріїв зупинки);
  - C (зворотна сила регуляризації);
  - fit_intercept (визначає, чи слід додати константу (вона же зміщення або перехоплення) до функції прийняття рішення);
  - class_weight (вагові показники);
  - solver (алгоритм для використання в задачі оптимізації);
  - та деякі інші.
- fit– метод, який відповідає за навчання нейронної мережі;

- score – метод, який відповідає за прогнозування результатів.

3.5.2 Тренування та тестування нейронної мережі

Тренування нейронної мережі це по суті процес вибору конфігурації (моделі) нейронної мережі із переліку можливих, що зводить похибку до мінімуму. Тому в процесі тренування мережі було випробувано та оцінено безліч варіантів налаштувань мережі та обрано найкращий. Далі на рис. 3.17 можна побачити функцію в якій ініціалізується модель (рядок 225), отримуються вибірка даних (рядки 227-229), проводиться розподіл даних 9 до 1 для тренувального та випробувальних наборів відповідно (рядки 231-237), тренування мережі (рядок 239) та випробування мережі на навчальному та випробувальному наборах даних.

```
def main():
    svc = LinearSVC(C=0.1)

    data = get_data_from_dir()
    data = remove_stars_feature_from_dataset(data)
    data = create_new_features_from_old_features_2(data)

    df = pd.DataFrame(data)
    train_data, test_data = train_test_split(df, test_size=0.1)

    train_features = train_data.iloc[:, :-1]
    train_targets = train_data.iloc[:, -1:].values.ravel()
    test_features = test_data.iloc[:, :-1]
    test_targets = test_data.iloc[:, -1:].values.ravel()

    svc.fit(train_features, train_targets)
    print('Score (train):', svc.score(train_features, train_targets))
    print('Score (test):', svc.score(test_features, test_targets))
```

Рис. 3.17 Код функції тренування та тестування моделі логістичної регресії

В результаті навчання було досягнуто відсотку точного прогнозування зі значення близьким до 77.0% на навчальному наборі та 76,6% на тестовому наборі. Результати випробувань даної моделі можна побачити на рис. 3.18.

```
Score (train): 0.769745004831815
Score (test): 0.765862876254181
```

Рис. 3.18 Результати випробувань модель логістичної регресії

3.6 Ініціалізація та тренування моделі багатошарового персептрону

3.6.1 Створення структури моделі. В даному розділі буде розглянуто ініціалізацію моделі методу опорних векторів, одну з моделей обраних у вході аналізу існуючих рішень. Для ініціалізації моделі було використано клас MLPClassifier з бібліотеки scikit-learn [17] мови Python. Були використані такі методи класу як:

- __init__ - метод-конструктор, ініціалізуючий структуру моделі на основі переданих параметрів або ж їх значень за замовченням
  - hidden_layer_sizes (i-й елемент представляє кількість нейронів в і-му прихованому шарі);
  - alpha (параметр штрафу L2 (термін регулювання));
  - learning_rate_init (використовується для визначення швидкості навчання та контролює розмір кроку при оновленні вагів);
  - tol (допуск критеріїв зупинки);
  - class_weight (вагові показники);
  - solver (алгоритм для використання в задачі оптимізації);
  - та деякі інші.
- fit– метод, який відповідає за навчання нейронної мережі;
- score – метод, який відповідає за прогнозування результатів.

3.6.2 Тренування та тестування нейронної мережі. Тренування нейронної мережі це по суті процес вибору конфігурації (моделі) нейронної мережі із переліку можливих, що зводить похибку до мінімуму. Тому в процесі

тренування мережі було випробувано та оцінено безліч варіантів налаштувань мережі та обрано найкращий. Далі на рис. 3.19 можна побачити функцію в якій ініціалізується модель (рядок 225), отримуються вибірка даних (рядки 227-229), проводиться розподіл даних 9 до 1 для тренувального та випробувальних наборів відповідно (рядки 231-237), тренування мережі (рядок 239) та випробування мережі на навчальному та випробувальному наборах даних.

```
def main():
    mlp = MLPClassifier(hidden_layer_sizes=(18, 6), learning_rate_init=0.5, activation='logistic')

    data = get_data_from_dir()
    data = remove_stars_feature_from_dataset(data)
    data = create_new_features_from_old_features_2(data)

    df = pd.DataFrame(data)
    train_data, test_data = train_test_split(df, test_size=0.1)

    train_features = train_data.iloc[:, :-1]
    train_targets = train_data.iloc[:, -1:].values.ravel()
    test_features = test_data.iloc[:, :-1]
    test_targets = test_data.iloc[:, -1:].values.ravel()

    mlp.fit(train_features, train_targets)
    print('Score (train):', mlp.score(train_features, train_targets))
    print('Score (test):', mlp.score(test_features, test_targets))
```

Рис. 3.19 Код функції тренування та тестування моделі логістичної регресії

В результаті навчання було досягнуто відсотку точного прогнозування зі значенням близьким до 77,3% на навчальному наборі та 76,1% на тестовому наборі. Результати випробувань даної моделі можна побачити на рис. 3.20.

```
Score (train): 0.7732763426872329
Score (test): 0.7608695652173914
```

Рис. 3.20 Результати випробувань моделі багатошарового перцептрону

3.7 Функція для побудови графічного відображення матриці невідповідностей

В даному розділі буде розглянуто реалізацію алгоритму який дозволяє створити та відобразити графічне відображення матриці невідповідностей

певної моделі. На рис. 3.21 можна бачити реалізацію даного алгоритму мовою Python, який виконує наступні інструкції:

- Отримує значення для заповнення матриці (рядки 2-18);
- За потреби переводить кількісні значення до відсоткових (рядки 27-29);
- Ініціалізує та наповнює значення графічне відображення матриці (рядки 30-45);
- Відображає матрицю (рядок 46);

```
def show_confusion_matrix(model, features, targets, show_as_percents=False):
    correct_guesses, wrong_guesses = 0, 0
    confusion_matrix = defaultdict(int)

    for feature, target in zip(features, targets):
        guess = numpy.argmax(model.predict(feature))

        correct_guesses = correct_guesses + 1 if guess == target else correct_guesses
        wrong_guesses = wrong_guesses + 1 if guess != target else wrong_guesses
        if show_confusion_matrix:
            confusion_matrix['tp'] = confusion_matrix['tp'] + 1 \
                if guess and target else confusion_matrix['tp']
            confusion_matrix['fp'] = confusion_matrix['fp'] + 1 \
                if guess and not target else confusion_matrix['fp']
            confusion_matrix['fn'] = confusion_matrix['fn'] + 1 \
                if not guess and target else confusion_matrix['fn']
            confusion_matrix['tn'] = confusion_matrix['tn'] + 1 \
                if not guess and not target else confusion_matrix['tn']

    plt.figure(figsize=(10, 2))
    if show_as_percents:
        cm = numpy.array([[round(confusion_matrix['tp']/len(targets)*100, 1),
                           round(confusion_matrix['fp']/len(targets)*100, 1)],
                          [round(confusion_matrix['fn']/len(targets)*100, 1),
                           round(confusion_matrix['tn']/len(targets)*100, 1)]])
    else:
        plt.figure(figsize=(10, 2))
        cm = numpy.array([[confusion_matrix['tp'], confusion_matrix['fp']],
                          [confusion_matrix['fn'], confusion_matrix['tn']]])
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title('Confusion matrix')
    plt.colorbar()
    classes = [1, 0]
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()
```

Рис.3.21 Код функції для побудови графічного відображення матриці невідповідностей

3.8 Розробка десктопного додатку

Десктопний додаток із архітектурою клієнт-сервер – це додаток, клієнтом якого є користувач ПК, сервером – веб-сервер. Логіка даного додатку зосереджена на ПК користувача, а веб-сервер відповідає лише за обробку запитів, ціллю яких є отримання невеликого об'єму інформації, яка стосується live-матчів. Недоліком такої системи є залежність клієнта від конкретної операційної системи, оскільки додаток розроблявся лише під ОС Windows 10. Проте чи не найважливішою перевагою є те, що дякуючи оптимізаційним мірам, прийнятим під час розробки, розроблене ПЗ використовує об'єм оперативної пам'яті в межах 80-120 мегабайт, що дозволяє запустити додаток на переважаючій більшості комп'ютерів.

3.8.1 Структура інтерфейсного модулю системи. Даний модуль відповідає за роботу та візуалізацію віджетів GUI розроблюваної системи. GUI системи представлено у вигляді одновіконного додатку. GUI розроблено за допомогою PyQt5 [8] – реалізація фреймворку для розробки графічних інтерфейсів для мови Python.

Основна частина інтерфейсу представлена в трьох елементах. Перші два являють собою текстові поля реалізовані за допомогою віджетів QTextEdit, перше з яких несе інформаційну роль та виводить до GUI інформацію про стан системи, друге має інформативний характер і виконує інформаційну функцію, сповіщаючи користувача останнім часом вдалої спроби запиту до сервера букмекера на отримання інформації про live-матчі. Третій же елемент слугує для візуалізації результатів, спрогнозованих live-матчів.

3.8.2 Структура модулю відповідаючого за прогнозування. В даному модулі реалізовано основний алгоритм збору даних щодо триваючих live-матчів та подальшого прогнозування їх завершення.

Порядок алгоритму наступний:

1) Перевірка активного Інтернет-з'єднання;

2) Ініціалізація нейронної мережі у вигляді її «натренованої» частини, а саме синапсів (вагів) збережених в файлі з розширенням JSON, за допомогою однойменного вбудованого в Python модуля (рис. 3.22);

```
# Read weights from file
wih1 = []
wh1h2 = []
wh2o = []

try:
    with open('D:/_NeuroBet_/weights/weights.json') as f:
        data = json.load(f)
        for item in data["wih1"]:
            mass = []
            for cell in item.values():
                mass.append(cell)
            wih1.append(mass)

        for item in data["wh1h2"]:
            mass = []
            for cell in item.values():
                mass.append(cell)
            wh1h2.append(mass)

        for item in data["wh2o"]:
            mass = []
            for cell in item.values():
                mass.append(cell)
            wh2o.append(mass)
except FileNotFoundError:
    print("Cannot read static_weights.json!")
    error_code = 3  # EC = 3 Could not find weights.txt file
    return predicts, matches_found, error_code  # EC = 1 driver problems
```

Рис. 3.22 Код алгоритму завантаження вагів натренованої мережі

3) Парсинг необхідних параметрів для триваючих матчів;

4) Прогнозування результатів матчів (рис. 3.24 можна побачити схематичне представлення алгоритму прогнозування;

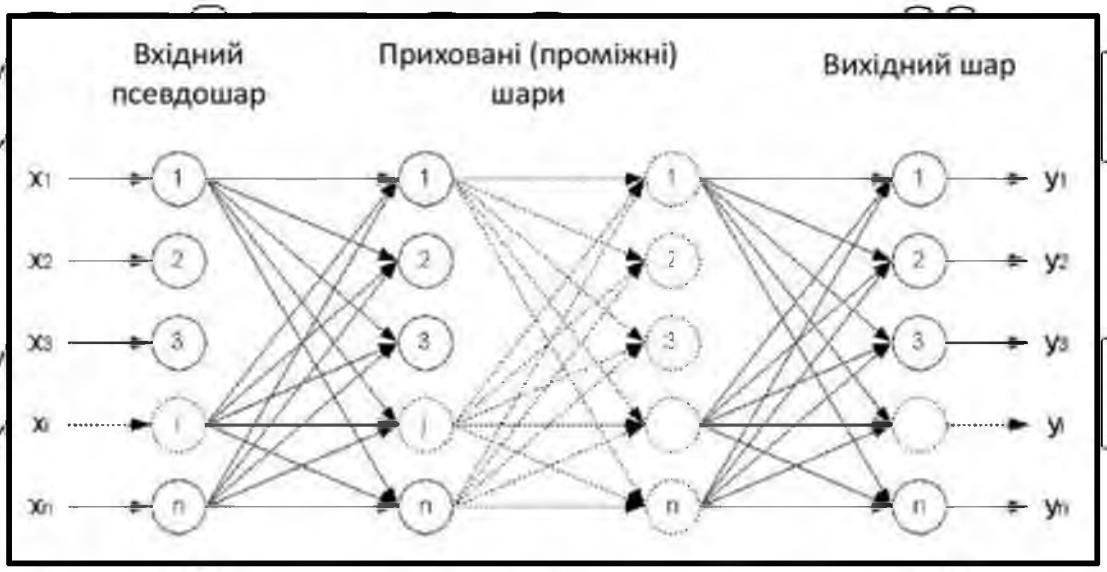

Рис. 3.23 Схематичне зображення процесу поширення сигналу через багатошарову нейронну мережу прямого поширення

```
# Predicting
row = map_info
data_set = row[1:14] + row[15:27] + row[27:28] + row[29:]
data_set = list(data_set)

data_set[0] = data_set[0] * 0.01
data_set[0] = float(str(data_set[0])[:5])
data_set[1] = data_set[1] * 0.001
data_set[1] = float(str(data_set[1])[:5])
data_set[2] = data_set[2] * 0.01
data_set[2] = float(str(data_set[2])[:5])

data_set[13] = data_set[13] * 0.01
data_set[13] = float(str(data_set[13])[:5])
data_set[14] = data_set[14] * 0.001
data_set[14] = float(str(data_set[14])[:5])
data_set[15] = data_set[15] * 0.01
data_set[15] = float(str(data_set[15])[:5])

data_set[26] = float(data_set[26][:-1]) * 0.01
if data_set[26] > 0:
    data_set[26] = data_set[26] - 0.0001
elif data_set[26] == 0:
    data_set[26] = data_set[26] + 0.0001
data_set[26] = float(str(data_set[26])[:16])
data_set[27] = float(data_set[27][:-1]) * 0.01
if data_set[27] > 0:
    data_set[27] = data_set[27] - 0.0001
elif data_set[27] == 0:
    data_set[27] = data_set[27] + 0.0001
data_set[27] = float(str(data_set[27])[:16])

inputs = numpy.array(data_set, ndmin=2).T
hidden_inputs1 = numpy.dot(wih1, inputs)
hidden_outputs1 = scipy.special.expit(hidden_inputs1)
hidden_inputs2 = numpy.dot(whlh2, hidden_outputs1)
hidden_outputs2 = scipy.special.expit(hidden_inputs2)
final_inputs = numpy.dot(wh2o, hidden_outputs2)
final_outputs = scipy.special.expit(final_inputs)

matches.append(team1 + ':' + team2 + '(' + map_names[game] + ')-' +
               str(numpy.argmax(final_outputs) + 1))

predicts.append(str(team1) + ' : ' + str(team2) + ':' + str(map_names[game]) + ': ' +
                'Team ' + str(numpy.argmax(final_outputs) + 1) + ' will win.')
```

Рис. 3.24 Код алгоритму прогнозування матчів

5) Пересилання результатів до основного потоку додатку, з метою подальшого їх відображення в таблиці результатів, на рис 3.25 можна побачити код реалізації класу відповідаючого за прогнозування матчів в другому потоці системи та надсилання результатів до першого потому системи для подальшого заповнення ними таблиці результатів:

```
class Threader(QThread):
    authResult = pyqtSignal(object)
    updateStart = pyqtSignal(object)

    def __init__(self, main_window):
        super().__init__()
        self.main_window = main_window
        self.predictor_result = []

    def run(self):
        while True:
            self.updateStart.emit('Updating the results table...')
            predictions, matches_found, error_code = live_predictor.get_predicts()

            if matches_found is False:
                if error_code == 1:
                    self.authResult.emit(['Problems with webdriver!', 1])
                elif error_code == 2:
                    self.authResult.emit(['Network currently down!', 2])
                elif error_code == 3:
                    self.authResult.emit(['Weights.json not found!', 3])
                elif error_code == 4:
                    self.authResult.emit(['Can\'t get active duty maps!', 4])
                else:
                    self.authResult.emit(['Matches are not found!', 0])
            else:
                self.authResult.emit(predictions)

            self.sleep(10)
```

Рис.3.25 Код класу відповідаючого за отримання прогнозів із модуля з

4 АНАЛІЗ ТОЧНОСТІ МОДЕЛЕЙ

Для того, щоб проаналізувати та порівняти досліджувані моделі треба оцініти їх за кількома метрика. Проте перед переходом до самих метрик необхідно ввести важливу концепцію для опису цих метрик у термінах класифікації помилок — confusion matrix (матриця невідповідностей).

4.1 Матриця невідповідностей

В галузі машинного навчання, й зокрема в задачі статистичної класифікації, матриця невідповідностей, також відома як матриця помилок, - це таблиця особливого компонування, що дає можливість унаочнювати продуктивність алгоритму, зазвичай керованого навчання. Кожен з рядків цієї матриці представляє зразки прогнозованого класу, тоді як кожен зі стовпців представляє зразки справжнього класу. Її назва походить від того факту, що вона дає можливість просто бачити, чи допускає система невідповідності між цими двома класами. Вона є особливим видом таблиці спряженості з двома вимірами "справжній" та "прогнозований" та ідентичними наборами "класів" в обох вимірах кожна з комбінацій виміру та класу є змінною цієї таблиці спряженості.

Тепер для кращого розуміння розглянемо приклад. Нехай задано вибірку з 13 зображень — 8 котів та 5 псів, де коти належать до класу 1, а пси належать до класу 0, справжній = [1,1,1,1,1,1,1,1,0,0,0,0,0], припустімо, що ми перевіряємо класифікатор, який розрізняє котів та псів. Для цього ці 13 зображень передаємо до класифікатора, і, нехай, класифікатор зробив 8 точних прогнозів, та 5 помилок: для 3 котів було помилково зроблено прогноз, що це пси (перші три прогнози), й для 2 псів було зроблено помилковий прогноз, що це коти. Прогнозований = [0,0,0,1,1,1,1,1,0,0,0,1,1]. Маючи ці два мічені набори (справжній та прогнозований), ми можемо створити матрицю невідповідностейт (рис. 4.1), що узагальнюватиме ці результати перевірки

класифікатора.

		Справжній клас	
		Кіт	Пес
Прогнозований клас	Кіт	5	2
	Пес	3	3

Рис. 4.1 Абстрактне представлення матриці невідповідностей

В цій матриці невідповідностей система порахувала, що із 8 зображень котів 3 були псами, а для 2 з 5 зображень псів було зроблено прогноз, що це коти. Всі правильні прогнози розміщено на діагоналі таблиці (виділеній жирним), тож цю таблицю легко візуально перевіряти на помилки прогнозування, оскільки їх представлено значеннями поза цією діагоналлю. Абстрактно, матриця невідповідностей є такою, як зображено на рис. 4.2.

		Справжній клас	
		П	Н
Прогнозований клас	П	ІП	ХП
	Н	ХН	ІН

Рис. 4.2 Абстрактне представлення матриці невідповідностей

де П = Позитивний, Н = Негативний, ІП = Істинно Позитивний, ХП = Хибно Позитивний, ІН = Істинно Негативний, ХН = Хибно Негативний.В прогнозній аналітиці, таблиця невідповідностей (англ. table of confusion, іноді також звана матрицею невідповідностей), — це таблиця з двома рядками та двома стовпцями, що повідомляє число хибно позитивних (англ. false positives), хибно негативних (англ. false negatives), істинно позитивних (англ. true positives) та істинно негативних (англ. true negatives) результатів. Це уможливлює аналіз, докладніший за просту пропорцію правильних класифікацій (точність). Точність видаватиме оманливі результати, якщо набір

даних є незбалансованим, тобто коли число спостережень в різних класах сильно різниться. Наприклад, якби в цих даних було 95 котів і лише 5 псів, певний класифікатор міг би класифікувати всі спостереження як котів. Загальна точність становила би 95 %, але, докладніше, класифікатор мав би 100 %-вий рівень розпізнавання (чутливість) для класу котів, але 0 %-вий рівень розпізнавання для класу псів. Міра F1 є ще ненадійнішою в таких випадках, і тут видавала би понад 97,4 %, тоді як поінформованість усуває це упередження, й видає 0 як імовірність поінформованого рішення для будь-якого виду гадання навмання (в даному випадку завжди гадання, що це є коти).

Виходячи з наведеної вище матриці невідповідностей, відповідною їй таблицею невідповідностей для котів буде матриця зображена на рис. 4.3.

		Справжній клас	
		Кіт	Не-кіт
Прогнозований клас	Кіт	5 істинно позитивних	2 хибно позитивних
	Не-кіт	3 хибно негативних	3 істинно негативних

Рис. 4.3 Матриця невідповідностей

Повертаючись до досліджуваних матриць, з використанням раніше реалізованої мовою Python функції для побудови графічного відображення досліджуваних моделей ми маємо наступні матриці невідповідностей:

- матриця невідповідностей для моделі логістичної регресії (рис. 4.4);
- матриця невідповідностей для моделі методу опорних векторів (рис. 4.5);
- матриця невідповідностей для моделі багатошарового персептрону (рис. 4.6);
- матриця невідповідностей для власної моделі багатошарового персептрону (рис. 4.7);

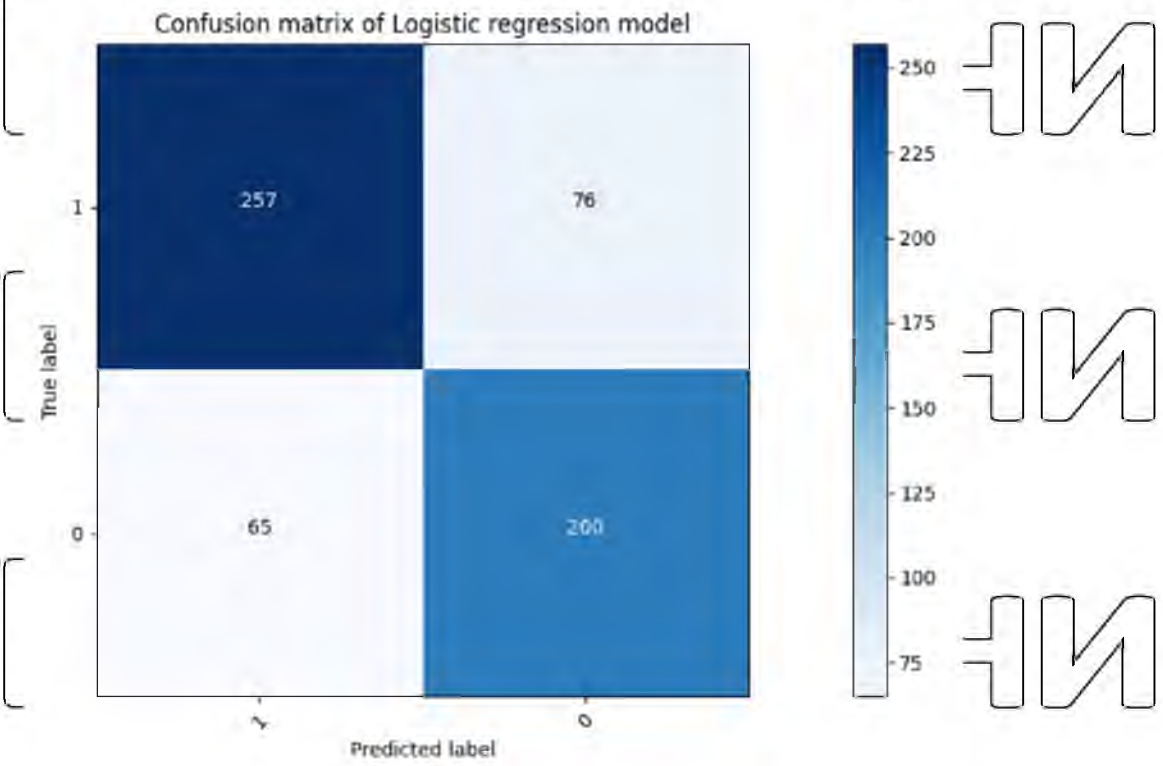

Рис. 4.4 Матриці невідповідностей моделі логістичної регресії

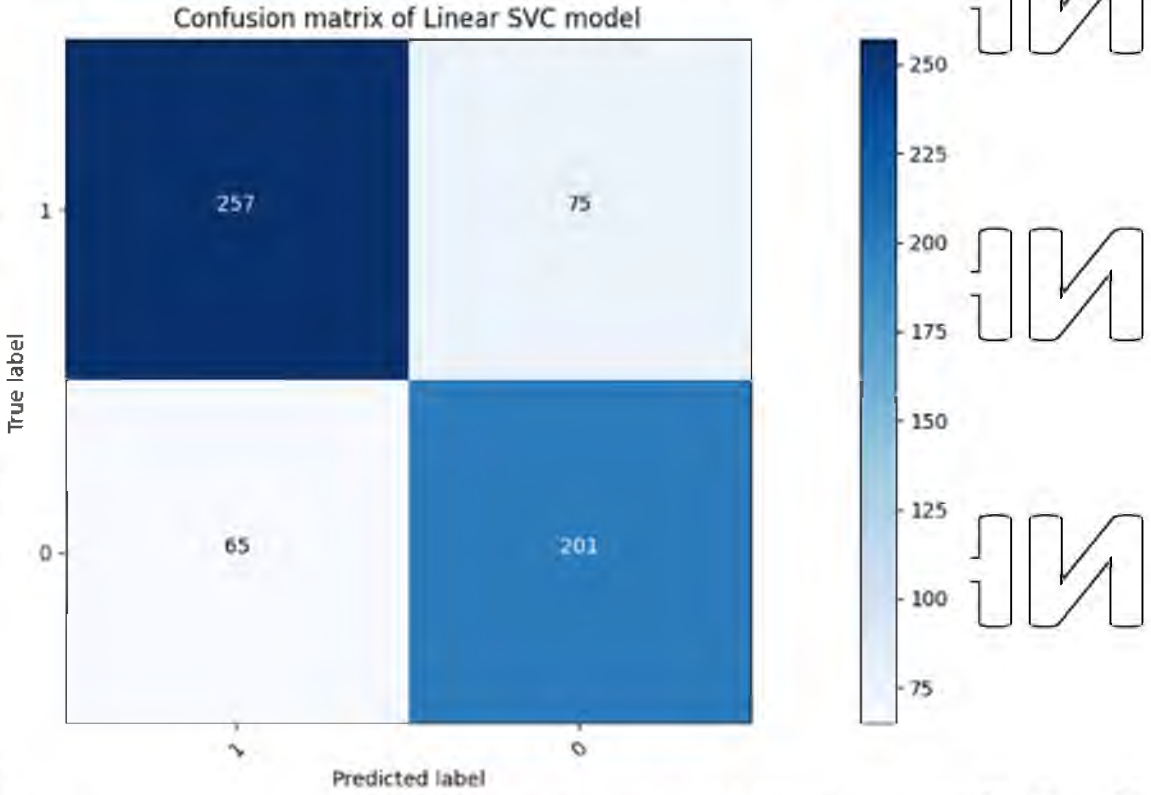

Рис. 4.5 Матриці невідповідностей моделі методу опорних векторів

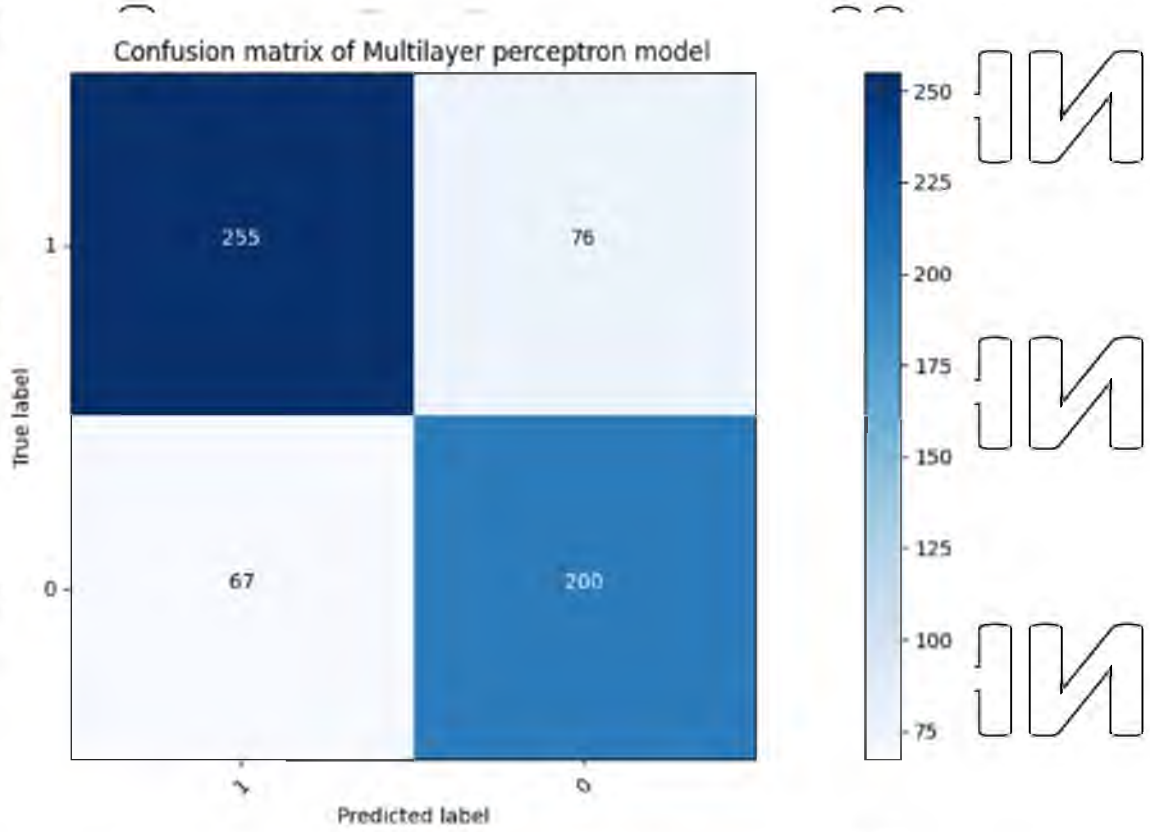

Рис. 4.6 Матриці невідповідностей моделі багатошарового персептрону

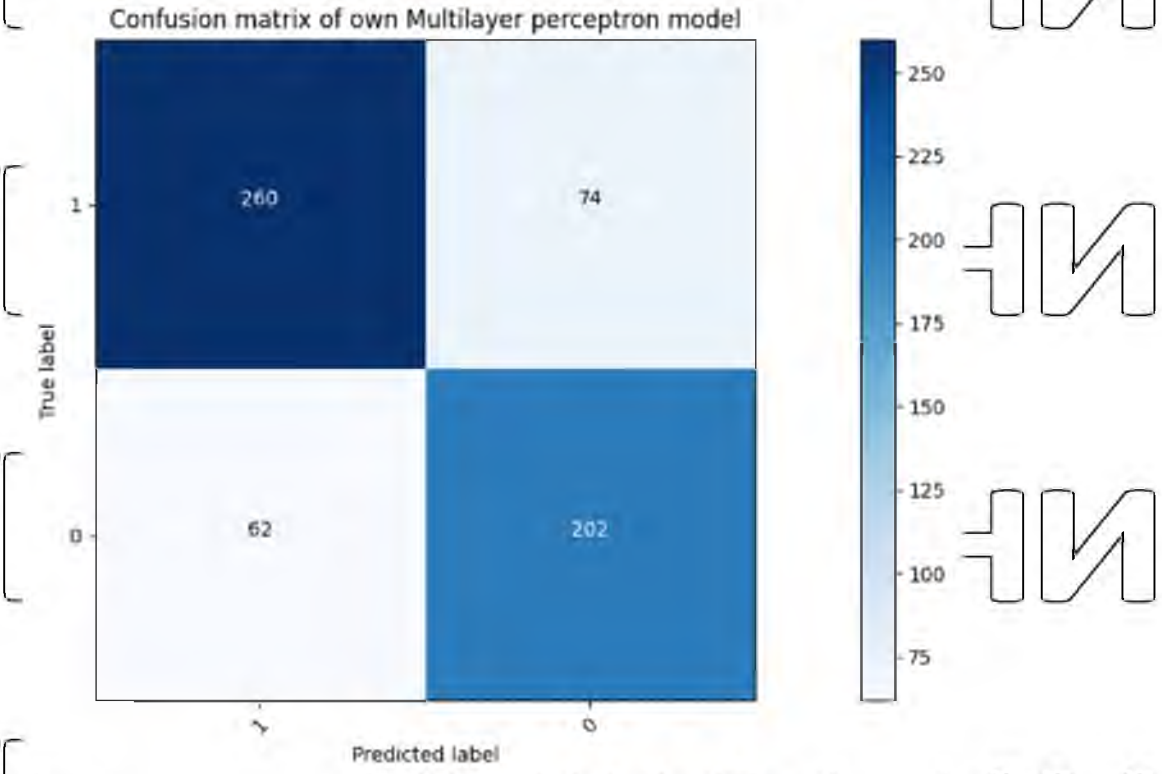

Рис. 4.7 Матриці невідповідностей моделі власного багатошарового персептрону

4.2 Аналіз метрики моделей

В даному розділі будуть порівняні наступні метрики.

- *Accuracy* - частка правильних відповідей алгоритму, в даному випадку є корисною через те, що класи є переважно рівними; розрахунок цієї величини представлений в (4.1):

- $$accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \quad (4.1)$$

де TP – кількість вірно спрогнозованих цілей першого класу;

TN – кількість вірно спрогнозованих цілей другого класу;

FP – кількість невірно спрогнозованих цілей першого класу;

FN – кількість невірно спрогнозованих цілей другого класу.

- *Precision* - частка об'єктів, названих класифікатором позитивними і при цьому дійсно є позитивними; розрахунок цієї величини представлений в (4.2):

$$precision = \frac{TP}{TP+FP}, \quad (4.2)$$

де TP – кількість вірно спрогнозованих цілей певного класу;

FP – кількість невірно спрогнозованих цілей певного класу.

Спочатку вирахуємо *accuracy* для кожної моделі:

- логістичної регресії (значення узяті з результатів представлених на рис. 4.4 та розраховані за формулою 4.1):
(257+200) / (257+200+76+65) = 0.7642;
- методу опорних векторів (значення узяті з результатів представлених на рис. 4.5 та розраховані за формулою 4.1):
(257+201) / (257+200+75+65) = 0.7671;
- багатошарового персептрону (значення узяті з результатів предетавлених на рис. 4.6 та розраховані за формулою 4.1):
(255+200) / (255+200+76+67) = 0.7609;

- власного багатошарового персептрону (значення узяті з результатів представлених на рис. 4.7 та розраховані за формулою 4.1):
(260+202) / (260+202+74+62) = 0.7726.

Наступним кроком вирахуємо *precision* для кожної моделі:

- логістичної регресії (значення узяті з результатів представлених на рис. 4.4 та розраховані за формулою 4.2):
((257 / (257+76)) + (200 / (200+65))) / 2 = 0.7632;
- методу опорних векторів (значення узяті з результатів представлених на рис. 4.5 та розраховані за формулою 4.2):
((257 / (257+75)) + (201 / (201+65))) / 2 = 0.7649;
- багатошарового персептрону (значення узяті з результатів представлених на рис. 4.6 та розраховані за формулою 4.2):
((255 / (255+76)) + (200 / (200+67))) / 2 = 0.7598;
- власного багатошарового персептрону (значення узяті з результатів представлених на рис. 4.7 та розраховані за формулою 4.2):
((260 / (260+74)) + (202 / (202+62))) / 2 = 0.7718.

Основуючись на отриманих результатах, можна прийти до висновку, що найкращих результатів було досягнуто з використанням власної моделі багатошарового персептрону. Тому в досліджуваній системі найдоцільніше використовувати власноруч розроблену модель багатошарового персептрону.

4.3 Інтерфейс програми

На рис. 4.10 можна побачити, вигляд вікна додатку якщо ПК не має активного Інтернет-з’єднання.

Рис. 4.10 Відсутнє інтернет-з'єднання

На рис. 4.11 можна побачити, що система сигналізує про процес перевірки наявності змін у переліку live-матчів.

Рис. 4.11 Оновлення вмісту таблиці результатів

На рис. 4.12 можна побачити, вигляд інтерфейсу додатку у випадку, якщо у момент перевірки не триває не одного підходящого матча.

Рис. 4.12 Підходящі live-матчів відсутні

На рис. 4.13 можна побачити, вигляд інтерфейсу додатку у випадку, якщо було проаналізовано та прогнозовано результати двох live-матчів.

Рис. 4.13 Прогнозовані результати для двох live-подій

На рис. 4.14 можна побачити, передбачену для користувача можливість виділити кілька рядків у таблиці результаті за необхідністю.

Рис. 4.14 Виділення кількох записів із таблиці результатів

4.4 Розгортання програмної системи

Повна діаграма розгортання розроблюваної системи слугує для того, щоб створити уявлення того, де знаходяться програмні вузли системи та як вони пов'язані між собою.

На рис. 4.15 можна побачити, що програмна система розміщується на двох вузлах. Перший вузол являє собою ПК клієнта та в ньому розміщуються 3 компоненти. Другий вузол знаходиться на сервері букмекера.

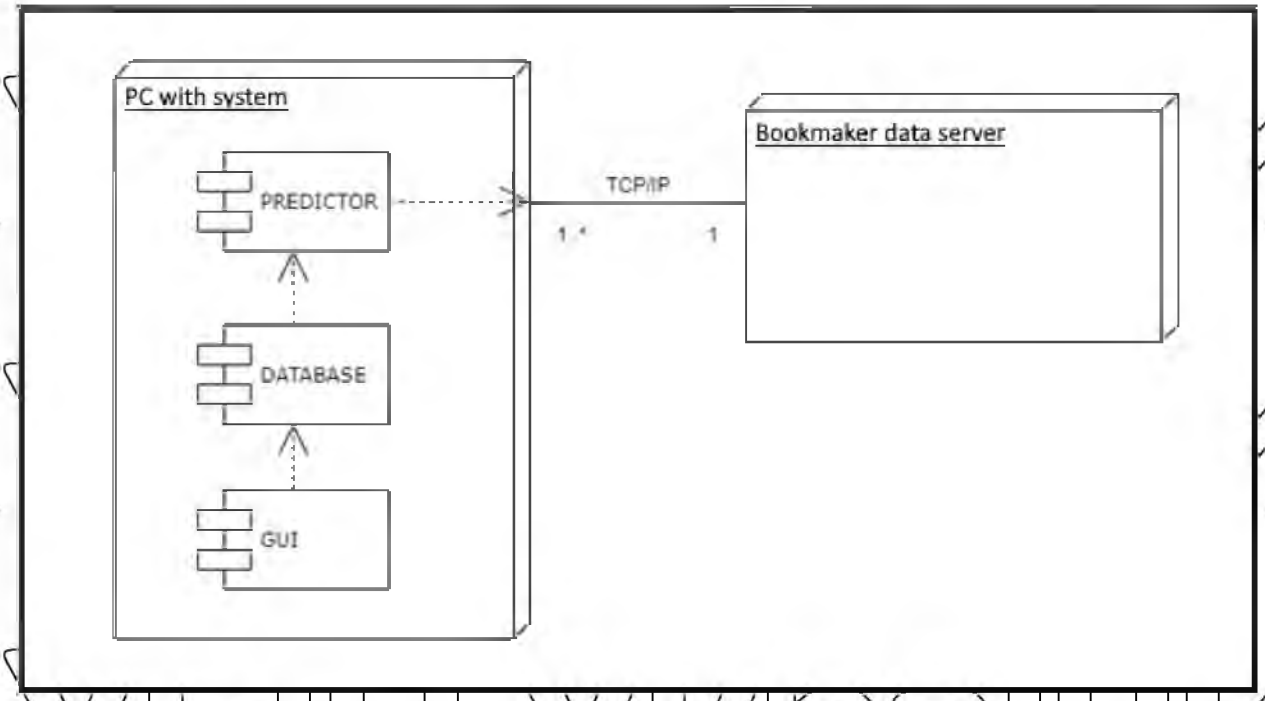

Рис. 4.15 Повна діаграма розгортання розробленої системи

ВИСНОВКИ

У даній роботі було досліджено та розроблено інтелектуальну систему для букмекерської контори, а також було описано моделювання роботи системи в основу якої було покладено використання інтелектуальної моделі. Було чітко охарактеризовано архітектуру, представлено змодельований опис предметної області, досліджено існуючі рішення, розібрані їх переваги та недоліки. Також з метою досягнення найвищої точності прогнозування даною системою, було досліджено та порівняно чотири різних інтелектуальних моделі. Для порівняння для кожної з моделей були побудовані матриці невідповідностей та на їх основі були розраховані метрики їх порівняння, що дозволили більш точно порівняти їх між собою. З урахуванням отриманих результатів точності близьких до 77%, результат можна вважати гарним.

Також було продумано та описано ціль та концепцію досліджуваного проекту, сформовано функціональні і нефункціональні вимоги до системи та визначено параметри якості для даної системи. Все було зображено у вигляді UML-діаграм та описано.

Було прокоментовано те, як працюють окремі модулі розробленою в проекті системи, те за що відповідає той, чи інший, фрагмент коду, та те, до яких рішень прийшли в процесі розробки.

Система була спроектована на основі вже існуючих рішень, було створено прототип, випробувано, після було об'єднано працююче програмне забезпечення з двох програмних модулів. Результати роботи було апробовано на конференції [42].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя. – М.: ДМК,2000.

2. Somboonphokkaphan A., Phimoltares S., Lursinsap C. Tennis Winner Prediction based on Time-Series History with Neural Modeling. URL: http://www.iaeng.org/publication/IMECS2009/IMECS2009_pp127-132.pdf (дата звернення: 29.06.2021).

3. 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 (дата звернення: 07.7.2021).

4. Wagner A., Narayanan. Using Machine Learning to predict tennis match outcomes. URL: http://deepakn94.github.io/assets/papers/6.867.pdf (дата звернення: 20.7.2021).

5. Panjan A., Sarabon N., Filipcic A. Prediction of the successfulness of tennis players with machine learning methods. URL: https://www.academia.edu/15216537/PREDVI%C4%90ANJE_NATJECATELJSKE_USPJE%C5%A0NOSTI_TENISA%C4%8CA_KORI%C5%A0TENJEM_METODA_STROJNOG_U%C4%8CENJA (дата звернення: 20.12.2020).

6. Bucquet A., Sarukkai V. The Bank is Ope: AI in Sports Gambling. URL: http://cs229.stanford.edu/proj2018/report/3.pdf (дата звернення: 10.6.2021).

7. Stubinger J., Mangold B., Knoll J. Machine Learning in Football Betting: Prediction of Match Results Based on Player Characteristics.

URL: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=10&ved=2ahUKEwj87tLbh8XpAhUwposKHdqQAPgQFjAJegQIARAB&url=https%3A%2F%2Fwww.mdpi.com%2F2076-3417%2F10%2F1%2F46%2Fpdf&usg=AOvVaw0cbn7JhgxvyOqrujf0jIU (дата звернення: 15.10.2020).

8. Інформація про Python фреймворк PyQt5 [Електронний ресурс]. – Режим доступу: https://pypi.org/project/PyQt5/

9. Інформація про Python фреймворк requests [Електронний ресурс]. – Режим доступу: https://pypi.org/project/requests/

10. Інформація про Python фреймворк bs4 [Електронний ресурс]. – Режим доступу: https://pypi.org/project/beautifulsoup4/

11. Інформація про Python фреймворк scipy [Електронний ресурс]. – Режим доступу: https://pypi.org/project/scipy/

12. Інформація про Python фреймворк numpy [Електронний ресурс]. – Режим доступу: https://pypi.org/project/numpy/

13. Інформація про Python фреймворк urllib [Електронний ресурс]. – Режим доступу: https://pypi.org/project/urllib4/

14. Інформація про Python фреймворк json [Електронний ресурс]. – Режим доступу: https://pypi.org/project/json5/

15. Інформація про Python фреймворк sys [Електронний ресурс]. – Режим доступу: https://pypi.org/project/os-sys/

16. Інформація про Python фреймворк datetime [Електронний ресурс]. – Режим доступу: https://pypi.org/project/daetime2/

17. Інформація про Python фреймворк scikit-learn [Електронний ресурс]. – Режим доступу: https://pypi.org/project/scikit-learn/

18. Python 3.6 documentation. [Електронний ресурс]. – Режим доступу: https://docs.python.org/fr/3.6/

19. Python 3.7 documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/fr/3.7/

20. Python 3.8 documentation. [Електронний ресурс]. – Режим доступу: https://docs.python.org/fr/3.8/

21. Марк Лутц, Изучаем Python, 4-е издание, Символ-Плюс, 2010. – 1280с.

22. Matt Telles, Python Power! The Comprehensive Guide, Thomson Course Technology, 2012. – 528с.

23. Aurélien Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems, O'Reilly UK Ltd., 2017 – 543с.

24. Ian H. Witten, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann; 4th edition, 2016 - 654с.

25. Peter Harrington, Machine Learning in Action, Manning; 1st edition, 2012 - 384с.

26. John Paul Mueller, Machine Learning For Dummies, For Dummies; 1. edition, 2016 - 432с.

27. Oliver Theobald, Machine Learning for Absolute Beginners: A Plain English Introduction, Scatterplot Press; 1st edition, 2017 - 169с.

28. Shai Shalev-Shwartz, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press; 1st edition, 2014 - 415с.

29. David Barber, Bayesian Reasoning and Machine Learning, Cambridge University Press, 2012 - 735с.

30. Trevor Hastie, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition, Springer, 2009 - 767с.

31. Tom M. Mitchell, Machine Learning, McGraw-Hill Education Ltd., 1997 - 352с.

32. Drew Conway, Machine Learning for Hackers: Case Studies and Algorithms to Get You Started, O'Reilly and Associates, 2012 - 329с.

33. Toby Segaran, Programming Collective Intelligence: Building Smart Web 2.0 Applications, O'Reilly and Associates, 2007 - 334c.

34. Andriy Burkov, The Hundred-Page Machine Learning Book, Andriy Burkov, 2019 - 160c.

35. Leonard Eddison, Python Machine Learning: A Technical Approach To Python Machine Learning For Beginners, CreateSpace Independent Publishing Platform, 2018 - 292c.

36. Sarah Guido, Introduction to Machine Learning with Python: A Guide for Data Scientists, O'Reilly UK Ltd., 2016 - 400c.

37. Bernd Klein, Python Course [Електронний ресурс]. – Режим доступу: https://www.python-course.eu/index.php

38. Зуев В.Н., Кемайкин В.К. Модифицированный алгоритм обучения нейронных сетей.

URL: https://cyberleninka.ru/article/n/modifitsirovannyy-algoritm-obucheniya-neyronnyh-setey (дата звернення: 20.01.2021).

39. Collins M. Feedforward Neural Networks.

URL: http://www.cs.columbia.edu/~mcollins/ff.pdf (дата звернення: 25.01.2021).

40. Sathya R., Annamma A. Comrarison of Supervise and Unsupervised Learning Algorithms for Pattern Classification.

URL: https://thesai.org/Downloads/IJARAI/Volume2No2/Paper_6-Comparison_of_Supervised_and_Unsupervised_Learning_Algorithms_for_Pattern_Classification.pdf (дата звернення: 15.08.2021).

41. Інформація про штучні нейронні мережі [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/5740123/.

42. Басов Г. ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПРОГНОЗУВАННЯ РЕЗУЛЬТАТІВ ДЛЯ БУКМЕКЕРСЬКОЇ КОНТОРИ / Збірник матеріалів конференції «Теоретичні та прикладні аспекти розробки комп'ютерних систем 2021», 11 листопада 2021 року, НУБіП України, Київ. – 2021.

ДОДАТОК А
Сторінок 10

Лістинг створення програмного забезпечення модуля нейронної мережі

Київ - 2021

```
import json

import numpy

import scipy.special

import sqlite3

import os

import matplotlib.pyplot as plt

import shutil


project_dir = os.path.dirname(__file__)


class NumpyEncoder(json.JSONEncoder):

    def default(self, obj):

        if isinstance(obj, numpy.ndarray):

            return obj.tolist()

        return json.JSONEncoder.default(self, obj)


def get_data(db_name):

    data = []

    conn = sqlite3.connect(project_dir + '\\databases\\' + db_name)

    cursor = conn.cursor()

    cursor.execute('''SELECT * FROM matches''')

    query_data = cursor.fetchall()

    for row in query_data:

        data_set = row[2:15] + row[16:29] + row[30:]

        data_set = list(data_set)
```

```
data_set[0] = data_set[0] * 0.01

data_set[0] = float(str(data_set[0])[:5])

data_set[1] = data_set[1] * 0.001

data_set[1] = float(str(data_set[1])[:5])

data_set[2] = data_set[2] * 0.01

data_set[2] = float(str(data_set[2])[:5])

data_set[13] = data_set[13] * 0.01

data_set[13] = float(str(data_set[13])[:5])

data_set[14] = data_set[14] * 0.001

data_set[14] = float(str(data_set[14])[:5])

data_set[15] = data_set[15] * 0.01

data_set[15] = float(str(data_set[15])[:5])

data_set[26] = data_set[26] * 0.01

data_set[26] = float(str(data_set[26])[:5])

data_set[27] = float(data_set[27][:-1]) * 0.01
if data_set[27] > 0:

    data_set[27] = data_set[27] - 0.0001

elif data_set[27] == 0:

    data_set[27] = data_set[27] + 0.0001

data_set[27] = float(str(data_set[27])[:6])

data_set[28] = data_set[28] * 0.01

data_set[28] = float(str(data_set[28])[:5])

data_set[29] = float(data_set[29][:-1]) * 0.01
if data_set[29] > 0:

    data_set[29] = data_set[29] - 0.0001

elif data_set[29] == 0:

    data_set[29] = data_set[29] + 0.0001
```

```
        data.append(data_set)
    conn.close()
    return data
```



```
class NeuralNet:
    def __init__(self, inputnodes, hiddennodes1, hiddennodes2, outputnodes, learningrate):
        self.inodes = inputnodes
        self.hnodes1 = hiddennodes1
        self.hnodes2 = hiddennodes2
        self.onodes = outputnodes
        self.lr = learningrate
        self.activation_func = lambda x: scipy.special.expit(x)
        self.wih1 = numpy.random.normal(0.0, pow(self.hnodes1, -0.5), (self.hnodes1, self.inodes))
        self.wh1h2 = numpy.random.normal(0.0, pow(self.hnodes2, -0.5), (self.hnodes2, self.hnodes1))
        self.wh2o = numpy.random.normal(0.0, pow(self.onodes, -0.5), (self.onodes, self.hnodes2))
```



```
    def train_net(self, inputs_list, targets_list):
        inputs = numpy.array(inputs_list, ndmin=2).T
        targets = numpy.array(targets_list, ndmin=2).T
```



```
        hidden_inputs1 = numpy.dot(self.wih1, inputs)
        hidden_outputs1 = self.activation_func(hidden_inputs1)
        hidden_inputs2 = numpy.dot(self.wh1h2, hidden_outputs1)
        hidden_outputs2 = self.activation_func(hidden_inputs2)
```

```
        final_inputs = numpy.dot(self.wh2o, hidden_outputs2)
        final_outputs = self.activation_func(final_inputs)

        output_errors = targets - final_outputs
        hidden_errors2 = numpy.dot(self.wh2o.T, output_errors)
        hidden_errors1 = numpy.dot(self.wh1h2.T, hidden_errors2)

        self.wh2o += self.lr * numpy.dot((output_errors * final_outputs * (1.0 - final_outputs)),
                                         numpy.transpose(hidden_outputs2))
        self.wh1h2 += self.lr * numpy.dot((hidden_errors2 * hidden_outputs2 * (1.0 - hidden_outputs2)),
                                          numpy.transpose(hidden_outputs1))
        self.wih1 += self.lr * numpy.dot((hidden_errors1 * hidden_outputs1 * (1.0 - hidden_outputs1)),
                                         numpy.transpose(inputs))
    def ask_net(self, inputs_list):
        inputs = numpy.array(inputs_list, ndmin=2).T
        hidden_inputs1 = numpy.dot(self.wih1, inputs)
        hidden_outputs1 = self.activation_func(hidden_inputs1)

        hidden_inputs2 = numpy.dot(self.wh1h2, hidden_outputs1)
        hidden_outputs2 = self.activation_func(hidden_inputs2)

        final_inputs = numpy.dot(self.wh2o, hidden_outputs2)
        final_outputs = self.activation_func(final_inputs)
        return final_outputs
```

```
if __name__ == '__main__':

    # Getting training data from db

    training_data = get_data('learning_set.db')

    # Getting training data from db

    testing_data = get_data('test_set.db')

    print(len(testing_data))

    # Initialise NeuralNet object

    input_nodes = 28

    hidden_nodes1 = 26

    hidden_nodes2 = 16

    output_nodes = 2


    learning_rate = 0.035

    avg_success = 0

    successes = []

    lowest_success = 1


    highest_success = 0

    highest_success_wih1 = []

    highest_success_wh1h2 = []

    highest_success_wh2o = []

    times = 1

    for r in range(times):
```

```
nn = NeuralNet(input_nodes, hidden_nodes1, hidden_nodes2, output_nodes, learning_rate)
# Train NeuralNet object
epochs = 25

for e in range(epochs):
    for record in training_data:
        e1 = record[:26]
        e1.append(record[27])
        e1.append(record[29])
        inputs_set = numpy.asfarray(e1)
        targets_set = numpy.zeros(output_nodes) + 0.0001
        if record[26] > record[28]:
            targets_set[0] = 0.9999
        else:
            targets_set[1] = 0.9999
        nn.train_net(inputs_set, targets_set)
scorecard = []

for record in testing_data:
    e1 = record[:26]
    e1.append(record[27])
    e1.append(record[29])
    inputs_set = numpy.asfarray(e1)
    if record[26] > record[28]:
        correct_label = 0
    else:
        correct_label = 1
    outputs = nn.ask_net(inputs_set)
```

```
        label = numpy.argmax(outputs)
        if label == correct_label:
            scorecard.append(1)
        else:
            scorecard.append(0)
    scorecard_array = numpy.asarray(scorecard)
    success = scorecard_array.sum() / scorecard_array.size
    print("Success = " + str(success)[:6])
    if highest_success < success:
        highest_success = success
        highest_success_wih1 = nn.wih1
        highest_success_wh1h2 = nn.wh1h2
        highest_success_wh2o = nn.wh2o
    avg_success += success
    sucesses.append(success)
    if lowest_success > success:
        lowest_success = success
print()
print('lowest_success = ' + str(lowest_success))
print('avg_success = ' + str(avg_success / times))
print('highest_success = ' + str(highest_success))
```

```
bins = [x for x in range(int(lowest_success * 100) - 1, int(highest_success * 100) + 1)]
percent_successes = [x * 100 for x in successes]
plt.hist(percent_successes, bins, histtype='bar', rwidth=0.8)
plt.show()

print('Do you want to rewrite weights of net?(y/n)')
answer = input()
if answer == 'y':
    with open('D:/_NeuroBet_/weights/test_weights.json', 'w') as f:
        weights_dict = {"wih1": list(), "wh1h2": list(), "wh2o": list()}
        for i in highest_success_wih1:
            local_dict = {}
            for counter, i_ in enumerate(i, 1):
                local_dict[str(counter)] = i_
            weights_dict["wih1"].append(local_dict)

        for i in highest_success_wh1h2:
            local_dict = {}
            for counter, i_ in enumerate(i, 1):
                local_dict[str(counter)] = i_
            weights_dict["wh1h2"].append(local_dict)

        for i in highest_success_wh2o:
            local_dict = {}
            for counter, i_ in enumerate(i, 1):
                local_dict[str(counter)] = i_
            weights_dict["wh2o"].append(local_dict)
```

```
json.dump(weights_dict, f, indent=2, ensure_ascii=False, cls=NumpyEncoder)

shutil.copy('D:/_NeuroBet_/weights/test_weights.json', 'D:/_NeuroBet_/weights/test_d_weights.json')
```

ДОДАТОК Б

Сторінок 12

Лістинг створення програмного забезпечення модуля графічного інтерфейсу

Київ – 2021

```
## -*- coding: utf-8 -*-
import datetime
import sys

from PyQt5 import QtCore, QtGui
from PyQt5.QtCore import QThread, pyqtSignal
from PyQt5.QtGui import QFont, QIcon, QPixmap
from PyQt5.QtWidgets import QApplication, QGridLayout, QWidget, QTextEdit, QTableWidget,
QTableWidgetItem, \
    QTableView, QAbstractItemView, QHeaderView, QLabel, QPushButton
#import tracemalloc

import opt_live_predictor as live_predictor

class Threader(QThread):
    authResult = pyqtSignal(object)
    updateStart = pyqtSignal(object)

    def __init__(self, main_window):
        super().__init__()
        self.main_window = main_window
        self.predictor_result = []

    def run(self):
        while True:
            self.updateStart.emit('Updating the results table...')
```

```
# current, peak = tracemalloc.get_traced_memory()
# print(f"Current memory usage is {current / 10 ** 6}MB; Peak was {peak / 10 ** 6}MB")
predictions, matches_found, error_code = live_predictor.get_predicts()

# - False case -
# self.sleep(2)
# error_code = 0
# matches_found = False

# - True case -
# self.sleep(2)
# error_code = 0
# matches_found = True
# predictions = ['Astralis:fnatic:Mirage:W2', 'Astralis:fnatic:Dust2:W2', 'Astralis:fnatic:Inferno:W2',
#                'TeamEmpire:NaVi:Dust2:W1',
'TeamEmpire:NaVi:Vertigo:W1', 'TeamEmpire:NaVi:Train:W1',
#                'TeamEmpire:NaVi:Inferno:W1',
'TeamEmpire:NaVi:Cobblestone:W1']

if matches_found is False:
    if error_code == 1:
        self.authResult.emit(['Problems with webdriver!', 1])
    elif error_code == 2:
        self.authResult.emit(['Network currently down!', 2])
    elif error_code == 3:
        self.authResult.emit(['Weights.json not found!', 3])
    elif error_code == 4:
        self.authResult.emit(['Can\'t get active duty maps!', 4])
```

```
            else:
                self.authResult.emit(['Matches are not found!', 0])
            else:
                self.authResult.emit(predictions)

            # for d in drivers_list:
            #     d.__exit__()
            # del drivers_list

            self.sleep(10)


class Window(QWidget):

    def __init__(self):
        super().__init__(flags=QtCore.Qt.FramelessWindowHint)

        self.offset = None
        self.predictions = []
        self.push_button_quit = QPushButton(self)
        self.push_button_minimize = QPushButton(self)
        self.text_edit_title = QTextEdit(self)
        self.text_edit_matches_found = QTextEdit(self)
        self.text_edit_last_update_time = QTextEdit(self)
        self.table_widget = QTableWidget(0, 4, self)

        self.label = QLabel(self)
        self.label.setPixmap(QPixmap('D:/_NeuroBet_/title_bar_icon.png'))
```

```
self.label.resize(720, 50)

self.label.move(0, -5)

self.label.stackUnder(self.table_widget)


self.label = QLabel(self)

self.label.setPixmap(QPixmap('D:/_NeuroBet_/title_icon.png'))

self.label.resize(35, 35)

self.label.move(5, 5)

self.label.stackUnder(self.table_widget)


self.bg_label = QLabel(self)

self.bg_label.setPixmap(QPixmap('D:/_NeuroBet_/bg_image.png'))

self.bg_label.resize(720, 690)

self.bg_label.move(0, 0)

self.bg_label.stackUnder(self.label)

self.bg_label.stackUnder(self.text_edit_last_update_time)

self.bg_label.stackUnder(self.text_edit_matches_found)


self.thread = Threader(self)

self.thread.authResult.connect(self.handleAuthResult)

self.thread.updateStart.connect(self.handleStartUpdate)

self.thread.start()


self.initUi()


self.setWindowIcon(QIcon('D:/_NeuroBet_/taskbar_icon.png'))

self.setWindowTitle('The Predictor')
```

```
        self.setMinimumSize(QtCore.QSize(720, 690))
        self.setMaximumSize(QtCore.QSize(720, 690))

    def initUi(self):
        font = QFont('Calibri', 14)
        stylesheet = """
            QTextEdit#matches_found, QTextEdit#update_time {
                color: rgb(154,157,161);
                background-color: rgb(42,62,97);
                border: none;
            }
            QTextEdit#title {
                color: white;
            }
            QPushButton#quit {
                border: none;
            }
            QPushButton#quit:hover {
                background-image: url("D:/_NeuroBet_/red-x.png");
                background-repeat: no-repeat;
            }
            QPushButton#minimize {
                border: none;
            }
            QPushButton#minimize:hover {
                background-color: rgb(40, 67, 115);
            }
```

```
QTableView::item
{
background: black;
}
"""
layout = QGridLayout()
layout.setSpacing(50)
self.push_button_quit.setObjectName('quit')
self.push_button_quit.move(678, 3)
self.push_button_quit.resize(40, 40)
self.push_button_quit.setStyleSheet(stylesheet)
self.push_button_quit.stackUnder(self.table_widget)
self.push_button_quit.setIcon(QIcon('D:/_NeuroBet_/x.png'))
self.push_button_quit.clicked.connect(self.button_quit)
self.push_button_quit.setToolTip('Close app')
self.push_button_quit.toolTip()
self.push_button_minimize.setObjectName('minimize')
self.push_button_minimize.move(640, 5)
self.push_button_minimize.resize(34, 34)
self.push_button_minimize.setStyleSheet(stylesheet)
self.push_button_minimize.stackUnder(self.table_widget)
self.push_button_minimize.setIcon(QIcon('D:/_NeuroBet_/-.png'))
self.push_button_minimize.clicked.connect(self.button_minimize)
self.push_button_minimize.setToolTip('Minimize app\'s window')
```

```
self.text_edit_title.setObjectName('title')

self.text_edit_title.setText("The Predictor")

self.text_edit_title.setFont(QFont('Calibri', 20))

self.text_edit_title.setDisabled(True)

self.text_edit_title.setStyleSheet('color:rgb(183, 206, 247);background: transparent;border: none;')

self.text_edit_title.move(40, 0)

self.text_edit_title.resize(270, 60)

self.text_edit_title.stackUnder(self.table_widget)

self.text_edit_title.show()

self.text_edit_matches_found.setObjectName('matches_found')

self.text_edit_matches_found.setFont(font)

self.text_edit_matches_found.setDisabled(True)

self.text_edit_matches_found.setStyleSheet(stylesheet)

self.text_edit_matches_found.move(70, 68)

self.text_edit_matches_found.resize(270, 40)

self.text_edit_matches_found.setToolTip('Message box, which signalising about updating \n'
                                        'of the results table or showing reason \n'
                                        'why a results table is empty.')

self.text_edit_matches_found.setToolTipDuration(5000)

self.text_edit_matches_found.hide()

self.text_edit_last_update_time.setObjectName('update_time')

self.text_edit_last_update_time.setFont(font)

self.text_edit_last_update_time.setDisabled(True)

self.text_edit_last_update_time.setStyleSheet(stylesheet)
```

```
self.text_edit_last_update_time.move(380, 68)

self.text_edit_last_update_time.resize(270, 40)

self.text_edit_last_update_time.setToolTip('Message box, showing a timestamp of last check or \n'

                                           'update of a results table.')

self.text_edit_last_update_time.setToolTipDuration(5000)

self.text_edit_last_update_time.hide()

self.setLayout(layout)

layout.setContentsMargins(60, 130, 60, 60)

self.table_widget.setHorizontalHeaderLabels(('Team 1', 'Team 2', 'Map', 'Prediction'))

self.table_widget.horizontalHeader().setSectionResizeMode(QHeaderView.Fixed)

self.table_widget.horizontalHeader().setHighlightSections(False)

self.table_widget.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)

self.table_widget.horizontalHeader().setStyleSheet('::section{color:rgb(154,157,161);'

                                                   'background: rgb(24, 39, 66);}')

self.table_widget.setStyleSheet('QTableView{background: rgb(31, 53, 89);border: none;}')

self.table_widget.setColumnWidth(0, 150)

self.table_widget.setColumnWidth(1, 150)

self.table_widget.setColumnWidth(2, 150)

self.table_widget.setColumnWidth(3, 151)

self.table_widget.setSelectionBehavior(QTableView.SelectRows)

self.table_widget.setEditTriggers(QAbstractItemView.NoEditTriggers)

self.table_widget.verticalHeader().setVisible(False)
```

```
        layout.addWidget(self.table_widget)

    def button_minimize(self):
        self.showMinimized()

    def button_quit(self):
        self.close()

    def handleAuthResult(self, result):
        """Function receives signals from predicting thread and
        renew result`s table
        """
        if len(result) == 2 and 'int' in str(type(result[1])):
            self.table_widget.setRowCount(0)
            self.text_edit_last_update_time.hide()
            if result[1] == 0:
                self.text_edit_last_update_time.setText(datetime.datetime.now().strftime("%H:%M:%S %d-%m-
%Y'))
                self.text_edit_last_update_time.setAlignment(QtCore.Qt.AlignCenter)
                self.text_edit_last_update_time.show()
            self.text_edit_matches_found.setText(result[0])
            self.text_edit_matches_found.setAlignment(QtCore.Qt.AlignCenter)
            self.text_edit_matches_found.show()
        else:
```

```
self.table_widget.setRowCount(0)

self.text_edit_last_update_time.setText(datetime.datetime.now().strftime('%H:%M:%S %d-%m-
%Y'))

self.text_edit_last_update_time.setAlignment(QtCore.Qt.AlignCenter)

self.text_edit_last_update_time.show()

self.text_edit_matches_found.hide()

for n, item in enumerate(result):

    splitted_item = item.split(':')

    self.table_widget.insertRow(n)

    self.table_widget.setItem(n, 0, QTableWidgetItem(str(splitted_item[0])))

    self.table_widget.setItem(n, 1, QTableWidgetItem(str(splitted_item[1])))

    self.table_widget.setItem(n, 2, QTableWidgetItem(str(splitted_item[2])))

    self.table_widget.setItem(n, 3, QTableWidgetItem(str(splitted_item[3])))

    if n % 2 == 0:

        r, g, b = 40, 90, 150

    else:

        r, g, b = 65, 85, 145

    for cell in range(4):

        self.table_widget.item(n, cell).setBackground(QtGui.QColor(r, g, b))

        self.table_widget.item(n, cell).setTextAlignment(QtCore.Qt.AlignCenter)

    self.table_widget.item(n, 3).setToolTip('W1 means The Predictor predicts that "Team 1" will win,
\n'                                         'otherwise W2 means The Predictor predicts \n'
```

```
                'that "Team 2" will win.')

    def handleStartUpdate(self, result):
        self.text_edit_matches_found.setText(result)
        self.text_edit_matches_found.show()

    # Next mouse events are used to customize window
    def mousePressEvent(self, event):
        if event.button() == QtCore.Qt.LeftButton:
            self.offset = event.pos()
        else:
            super().mousePressEvent(event)

    def mouseMoveEvent(self, event):
        if self.offset is not None and event.buttons() == QtCore.Qt.LeftButton:
            self.move(self.pos() + event.pos() - self.offset)
        else:
            super().mouseMoveEvent(event)
    def mouseReleaseEvent(self, event):
        super().mouseReleaseEvent(event)
if __name__ == '__main__':
    # tracemalloc.start()
    app = QApplication(sys.argv)
    screen = Window()
    screen.show()
    sys.exit(app.exec_())
```

ДОДАТОК В

Сторінок 14

Лістинг створення програмного забезпечення модуля прогнозуючого результат спортивної події

Київ – 2021

```
# import time
import json
import requests
from urllib.request import urlopen
from urllib.error import URLError
import numpy
import scipy.special
from bs4 import BeautifulSoup

def get_active_map_pool():
    map_pool = []
    queue = requests.get('https://liquipedia.net/counterstrike/Portal:Maps').content
    page = BeautifulSoup(str(queue), 'html.parser')

    for m in page.find('table', {'class': 'navbox navigation-not-searchable'}). \
        findAll('table', {'class': 'nowraplinks navbox-subgroup wiki-backgroundcolor-light'}):
        if 'Active Duty' in m.text:
            for tr in m.find('tbody').findAll('tr'):
                if 'Active Duty' in tr.text:
                    for a in tr.find('td', {'class': 'navbox-list navbox-odd'}).findAll('a'):
                        if a.text == 'Dust II':
                            map_pool.append('Dust2')
                        else:
                            map_pool.append(a.text)
```

```
    return map_pool

def get_predicts():

    predicts = []

    matches_found = False

    error_code = 0


    try:

        urlopen("http://google.com")

    except URLError:

        print("Network currently down!")

        error_code = 2  # EC = 2 none internet connection detected

        return predicts, matches_found, error_code

    try:

        active_duty_map_pool = get_active_map_pool()

    except Exception:

        error_code = 4  # EC = 4 cannot scrap active duty maps

        return predicts, matches_found, error_code


    # Read weights from file

    wih1 = []

    wh1h2 = []

    wh2o = []


    try
```

```
        with open('D:/_NeuroBet_/weights/weights.json') as f:
            data = json.load(f)
            for item in data["wih1"]:
                mass = []
                for cell in item.values():
                    mass.append(cell)
                wih1.append(mass)

            for item in data["wh1h2"]:
                mass = []
                for cell in item.values():
                    mass.append(cell)
                wh1h2.append(mass)

            for item in data["wh2o"]:
                mass = []
                for cell in item.values():
                    mass.append(cell)
                wh2o.append(mass)
    except FileNotFoundError:
        print("Cannot read static_weights.json!")
        error_code = 3  # EC = 3 Could not find weights.txt file
        return predicts, matches_found, error_code  # EC = 1 driver problems

    matches_link = "https://www.hltv.org/matches"
    hltv_link = 'https://www.hltv.org'
    matches = []
```

```
# print('Finding live-matches...')
request = requests.get(matches_link).content
matches_page = BeautifulSoup(str(request), 'html.parser')
live_matches_arr = []
if len(matches_page.findAll('div', {'class': 'live-matches'})) >= 1:
    live_matches_section = matches_page.find('div', {'class': 'live-matches'}). \
        findAll('div', {'class': 'live-match'})
    for match in live_matches_section:
        if len(match.findAll('a')) > 0:
            live_matches_arr.append(match)
# print('Have found ' + str(len(live_matches_arr)) + ' live-matches.\n')
if live_matches_arr is not []:
    links = [x.findAll('a') for x in live_matches_arr]
    for link in links:
        link = link[0].get('href')
        request = requests.get(hltv_link + link).content
        matches_page = BeautifulSoup(str(request), 'html.parser')
        # Check are playing maps are already chosen
        maps = matches_page.find('div', {'class': 'flexbox-column'}).findAll('div', {'class': 'mapname'})
        maps_are_chosen = True
```

```
for Map in maps:
    if Map.text not in active_duty_map_pool:
        maps_are_chosen = False
if maps_are_chosen is False:
    continue
map_names = []
team1_rates = []
team2_rates = []
team1players_stats = []
team2players_stats = []

# Getting of team names
teams = matches_page.findAll('div',{'class': 'teamName'})
team1 = teams[0].text
team2 = teams[1].text
# The original  event_time = driver.find_element_by_class_name('time').text

# Active matches check
# if team1 + team2 + event_time in matches:
#     continue
maps = matches_page.findAll('div', {'class': 'mapholder'})
for Map in maps:
```

```
        map_name = Map.find('div', {'class': 'mapname'}).text
        map_names.append(map_name)

    if 'TBA' in maps:
        print('Picking maps...')
        continue

    if len(map_names) < 1:
        continue

    team1_link = matches_page.find('div', {'class': 'team1-gradient'})
    team1_link = team1_link.find('a').get('href')
    team1_link = hltv_link + team1_link

    team2_link = matches_page.find('div', {'class': 'team2-gradient'})
    team2_link = team2_link.find('a').get('href')
    team2_link = hltv_link + team2_link

    request = requests.get(team1_link + '#tab-statsBox').content
    team1_page = BeautifulSoup(str(request), 'html.parser')

    all_maps_container = team1_page.findAll('div', {'class': 'map-statistics-container'})
    for iterator in range(len(map_names)):
        for map_container in all_maps_container:
            processing_map = map_container.find('div', {'class': 'map-statistics-row-map-mapname'})
            map_rate = map_container.find('div', {'class': 'map-statistics-row-win-percentage'})
```

```
            if map_names[iterator] == processing_map.text:
                team1_rates.append(map_rate.text)

    if len(team1_rates) != len(map_names):
        continue

    # 4 Getting of team1world_rank
    team1world_rank = team1_page.findAll('div', {'class': 'profile-team-stat'})[0]
    team1world_rank = team1world_rank.findAll('a')
    if len(team1world_rank) < 1:
        continue

    team1world_rank = team1world_rank[0].text
    team1world_rank = int(team1world_rank[1:])

    # 5 Getting of team1_in_top
    team1_in_top = team1_page.findAll('div', {'class': 'profile-team-stat'})[1]
    team1_in_top = team1_in_top.find('span').text
    team1_in_top = int(team1_in_top)

    # 7 Getting of team1average_age
    if len(team1_page.findAll('div', {'class': 'profile-team-stat'})) < 3:
        continue

    team1average_age = team1_page.findAll('div', {'class': 'profile-team-stat'})[2]
    team1average_age = team1average_age.find('span').text
```

```
team1average_age = float(team1average_age)

# 9 Getting of team1players_stats

team1players_links = team1_page.find('div', {'class': 'bodyshot-team-bg'})

team1players_links = team1players_links.findAll('a', {'class': 'col-custom'})

team1_links = []

for team1_link in team1players_links:

    team1_links.append(hltv_link + team1_link.get('href'))

for team1_link in team1_links:

    request = requests.get(team1_link).content

    team1_player_page = BeautifulSoup(str(request), 'html.parser')

    cols = team1_player_page.find('div', {'id': 'infoBox'}).find('div', {'class': 'two-col'}). \
        findAll('div', {'class': 'col'})

    first_value = float(cols[0].findAll('span', {'class': 'statsVal'})[1].text)

    second_value = float(cols[1].findAll('span', {'class': 'statsVal'})[1].text)

    team1players_stats.append([first_value, second_value])


request = requests.get(team2_link + '#tab-statsBox').content

team2_page = BeautifulSoup(str(request), 'html.parser')

all_maps_container = team2_page.findAll('div', {'class': 'map-statistics-container'})

for iterator in range(len(map_names)):

    for map_container in all_maps_container:
```

```
            processing_map = map_container.find('div', {'class': 'map-statistics-row-map-mapname'})
            map_rate = map_container.find('div', {'class': 'map-statistics-row-win-percentage'})
            if map_names[iterator] == processing_map.text:
                team2_rates.append(map_rate.text)
    if len(team2_rates) != len(map_names):
        continue
    # 4 Getting of team2world_rank
    team2world_rank = team2_page.findAll('div', {'class': 'profile-team-stat'})[0]
    team2world_rank = team2world_rank.findAll('a')
    if len(team2world_rank) < 1:
        continue
    team2world_rank = team2world_rank[0].text
    team2world_rank = int(team2world_rank[1:])
    # 5 Getting of team2_in_top
    team2_in_top = team2_page.findAll('div', {'class': 'profile-team-stat'})[1]
    team2_in_top = team2_in_top.find('span').text
    team2_in_top = int(team2_in_top)
    # 7 Getting of team2average_age
    if len(team2_page.findAll('div', {'class': 'profile-team-stat'})) < 3:
        continue
```

```
team2average_age = team2_page.findAll('div', {'class': 'profile-team-stat'})[2]

team2average_age = team2average_age.find('span').text

team2average_age = float(team2average_age)

# 9 Getting of team2players_stats

team2players_links = team2_page.find('div', {'class': 'bodyshot-team-bg'})

team2players_links = team2players_links.findAll('a', {'class': 'col-custom'})

team2_links = []

for team2_link in team2players_links:
    team2_links.append(hltv_link + team2_link.get('href'))

for team2_link in team2_links:
    request = requests.get(team2_link).content
    team2_player_page = BeautifulSoup(str(request), 'html.parser')

    cols = team2_player_page.find('div', {'id':'infoBox'}).find('div', {'class': 'two-col'}). \
        findAll('div', {'class': 'col'})
    first_value = float(cols[0].findAll('span', {'class': 'statsVal'})[1].text)

    second_value = float(cols[1].findAll('span', {'class': 'statsVal'})[1].text)

    team2players_stats.append([first_value, second_value])

# Live matches check

# matches.append(team1 + team2 + event_time)
# The original print('Predict: \n ')
```

```
for game in range(len(map_names)):
    map_info = [team1, team1world_rank, team1_in_top, team1average_age,
                team1players_stats[0][0], team1players_stats[0][1],
                team1players_stats[1][0], team1players_stats[1][1],
                team1players_stats[2][0], team1players_stats[2][1],
                team1players_stats[3][0], team1players_stats[3][1],
                team1players_stats[4][0], team1players_stats[4][1],
                team2, team2world_rank, team2_in_top, team2average_age,
                team2players_stats[0][0], team2players_stats[0][1],
                team2players_stats[1][0], team2players_stats[1][1],
                team2players_stats[2][0], team2players_stats[2][1],
                team2players_stats[3][0], team2players_stats[3][1],
                team2players_stats[4][0], team2players_stats[4][1],
                map_names[game],
                team1_rates[game],
                team2_rates[game],
                ]
    # Predicting
    row = map_info
    data_set = row[1:14] + row[15:27] + row[27:28] + row[29:]
    data_set = list(data_set)
    data_set[0] = data_set[0] * 0.01
    data_set[0] = float(str(data_set[0])[:5])
    data_set[1] = data_set[1] * 0.001
```

```
data_set[1] = float(str(data_set[1])[:5])

data_set[2] = data_set[2] * 0.01

data_set[2] = float(str(data_set[2])[:5])


data_set[13] = data_set[13] * 0.01

data_set[13] = float(str(data_set[13])[:5])

data_set[14] = data_set[14] * 0.001

data_set[14] = float(str(data_set[14])[:5])

data_set[15] = data_set[15] * 0.01

data_set[15] = float(str(data_set[15])[:5])


data_set[26] = float(data_set[26][:-1]) * 0.01

if data_set[26] > 0:

    data_set[26] = data_set[26] - 0.0001

elif data_set[26] == 0:

    data_set[26] = data_set[26] + 0.0001

data_set[26] = float(str(data_set[26])[:6])

data_set[27] = float(data_set[27][:-1]) * 0.01

if data_set[27] > 0:

    data_set[27] = data_set[27] - 0.0001

elif data_set[27] == 0:

    data_set[27] = data_set[27] + 0.0001

data_set[27] = float(str(data_set[27])[:6])


inputs = numpy.array(data_set, ndmin=2).T

hidden_inputs1 = numpy.dot(wih1, inputs)

hidden_outputs1 = scipy.special.expit(hidden_inputs1)
```

```
        hidden_inputs2 = numpy.dot(wh1h2, hidden_outputs1)
        hidden_outputs2 = scipy.special.expit(hidden_inputs2)
        final_inputs = numpy.dot(wh2o, hidden_outputs2)
        final_outputs = scipy.special.expit(final_inputs)
        matches.append(team1 + ':' + team2 + '(' + map_names[game] + ')- ' +
                       str(numpy.argmax(final_outputs) + 1))
        predicts.append(str(team1) + ' : ' + str(team2) + ':' + str(map_names[game]) + ': ' +
                        'Team ' + str(numpy.argmax(final_outputs) + 1) + ' will win.')
        matches_found = True
    return predicts, matches_found, error_code
```