МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно- обчислювальних систем і управління

КУЧАР Олександр Михайлович

Програмний модуль навчання гри в покер / Software Module for Poker Game Learning

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
О. М. Кучар

Науковий керівник:
к.т.н., доцент, П. Є. Биковий

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **М. П. Комар**

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
КУЧАР Олександр Михайлович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Програмний модуль навчання гри в покер / Software Module for Poker Game Learning
керівник роботи Биковий Павло Євгенович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. №753.

2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити:
- провести опис предметної області;
- дослідити найкращі методи для навчання;
- зробити аналіз предметної області;
- зробити постановку задачі та описати шляхи її вирішення;
- дослідити теоретичні основи теорії ймовірності та прийняття рішень;
- зробити систему для симуляції ігор;
- зробити систему для визначення найкращого ходу на основі симуляційних даних;
- розробити методи для визначення комбінацій та шансів на перемогу;
- зробити методи для побудови графіків на основі даних про раунди гри.

5. Перелік графічного матеріалу в роботі:
- схема роботи симуляцій ігор;
- схема аналізу карт та оцінки комбінацій для кожного раунду гри;
- загальний алгоритм гри;
- процес побудови графіків та діаграм статистичних даних про раунди.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі «Unicheck».	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.06. 2024 р.	

Студент _____ О.М. Кучар
(підпис)

Керівник проекту _____ П.Є. Биковий
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Програмний модуль навчання гри в покер» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом 55 сторінок і містить 12 ілюстрацій, чотири таблиці, два додатки та 25 використаних джерел.

Метою даної роботи є створення ефективного інструменту, який допоможе гравцям або початківцям здобути основні знання про правила та стратегії гри, зрозуміти або поглибити основні знання в теорії ігор, теорії ймовірності, оцінки шансів та математичних розрахунках.

Для розроблення та ознайомлення з завданням було обрано такі методи, як розрахунок ймовірності покерних рук, обчислення кількості комбінацій, аналіз з обмеженою інформацією, базові алгоритми прийняття та побудови дерева рішень, математичні оцінки ризиків, теорії нешівської рівноваги та змішаних стратегій.

Результати роботи – розроблено програмний модуль навчання гри в покер, який забезпечує користувачу програму для навчання гри в «Техаський холдем».

Результати роботи можна використовувати в особистих та організаційних завданнях та в інших сферах де потрібно аналізувати інформацію та/або проводити розрахунки теорії ймовірності.

Ключові слова: ПРОГРАМНИЙ МОДУЛЬ, НАВЧАННЯ, СТРАТЕГІЇ, МАТЕМАТИЧНИЙ РОЗРАХУНОК.

ANNOTATION

Qualification Work on the topic "Software Module for Learning Poker" for obtaining the educational degree "Bachelor" in the specialty 122 "Computer Science" of the educational program "Computer Science" is written with a volume of 55 pages and contains of 12 illustrations, four tables, two appendices, and 25 used sources.

The aim of this work is to create an effective tool to help players or beginners acquire basic knowledge of game rules and strategies, understand or deepen their understanding of game theory, probability theory, odds evaluation, and mathematical calculations.

To develop and familiarize with the task, methods such as calculating poker hand probabilities, computing the number of combinations, analysis with limited information, basic decision-making algorithms, building decision trees, mathematical risk assessments, Nash equilibrium theory, and mixed strategies were chosen.

The results of the work include the development of a poker training software module that provides a program for learning the game of Texas Hold'em.

The results can be used in personal and organizational tasks and in other areas where information analysis and/or probability theory calculations are required.

Keywords: SOFTWARE MODULE, LEARNING, STRATEGIES, MATHEMATICAL CALCULATION.

ЗМІСТ

ВСТУП

Гра в покер була визначена як корисна область для сучасних досліджень у галузі штучного інтелекту та теорії ймовірності через необхідність роботи з прихованою інформацією та випадковістю подій. Визначення покеру як корисної області досліджень неминуче призвело до підвищеної уваги з боку академічних дослідників, які досліджували багато різних напрямків у галузі комп'ютерного покеру.

Навчання гри в покер є цікавою та програмно мало-розв’язаною задачею у галузі штучного інтелекту та теорії прийняття рішень через складність цієї гри та її реалістичному відображенні реальних ситуацій прийняття рішень, оскільки покер відомий своєю складною стратегією, що включає в себе не лише вивчення правил гри, але й розвиток аналітичних, стратегічних та математичних навичок.

Тому розробка програмного модуля навчання гри в покер є актуальною задачею, що дасть змогу гравцям навчитись враховувати не лише власні карти, але й імовірність та можливі рішення інших учасників. Це дозволить передбачати дії та ходи інших гравців, що в результаті приведе до перемоги та бажаного результату навчитись грати в покер.

Метою даної роботи є створення ефективного інструменту, який допоможе гравцям або початківцям здобути основні знання про правила та стратегії гри, зрозуміти або поглибити основні знання в теорії ігор, теорії ймовірності, оцінки шансів та математичних розрахунках.

Основні завдання які треба вирішити:

- провести опис предметної області;
- дослідити найкращі методи для навчання;
- зробити аналіз предметної області;
- зробити постановку задачі та описати шляхи її вирішення;
- дослідити теоретичні основи теорії ймовірності та прийняття рішень;
- зробити систему для симуляції ігор;
- зробити систему для визначення найкращого ходу;

– розробити методи для визначення комбінацій та шансів на перемогу;

– зробити методи для побудови графіків на основі даних про раунди гри.

Об'єктом для дослідження є програмний модуль навчання для гри в найбільш популярний вид покеру - «Техаський холдем».

Предметом для дослідження постають методи та алгоритми навчання гри в покер, їх ефективність та аналіз та вплив цих методів на покращення ігрових навичок користувачів.

Методи для дослідження включають: розрахунок ймовірності покерних рук, та обчислення кількості комбінацій використовуючи теорію ймовірності, алгоритми прийняття рішень, математичні оцінки ризиків та обчислення сили комбінації.

Функціональність системи полягатиме в симуляції навчальних ігор та навчанню противника на основі зібраних даних для прийняття рішень при грі з користувачем та відображені підказок, таких як – розрахунок шансів на перемогу та поточної комбінації. Візуалізації процесу навчання та даних про ігрові раунди.

Практичне застосування програмного модуля полягає в навчанні користувачів, аналізу ігрових даних, підвищенні ефективності гри та може використовуватись для навчальних цілей або дотичних сферах де використовується аналіз даних і теорії ймовірності.

Результати роботи були апробовані на міжнародній науково-практичній конференції: «Перспективні напрямки розвитку економіки, обліку, управління та права: Теорія і практика» та подані до друку в «Перспективні напрямки розвитку економіки, обліку, управління та права: теорія і практика: збірник тез доповідей міжнародної науково-практичної конференції (Ізмаїл, 18 травня 2024 р.). Ізмаїл: ЦФЕНД, 2024. 63 с.».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис базових правил версії гри в покер «Техаський холдем»

Техаський холдем – це найпоширеніший вид покеру з загальними картами, який можуть грати від двох до десяти гравців одночасно. Для гри використовують колоду із п’ятдесяти двох карт, які розділені на чотири масті: ♠ - піка, ♥ - чирва, ♦ - бубна, ♣ - трефа та дев’ять карт розділених за рангом від двійки до десятки та чотирьох карт із буквеним рангом, а саме: J – валет, Q - дама, K – король, A - туз . У Техаському холдемі кожному гравцеві роздається по дві особисті закриті карти, що означає, що бачити їх може лише гравець, якому вони дісталися [1-3]. Впродовж гри всім гравцям може відкритись ще п’ять загальних карт, які доступні всім гравцям. Метою гри є складання найкращої можливої комбінації із п'яти карт із своїх двох особистих карт і будь-яких із загальних карт. Тривалість гри поділена на чотири основні раунди ставок та останній п’ятий раунд де визначається переможець [4-6]:

1. Префлоп – перший раунд де кожен гравець отримує дві закриті (кишенькові) карти та починається перший раунд ставок. Гравці приймають рішення на основі своїх кишенькових карт.
2. Флоп - на стіл викладаються три загальні (спільні) карти. Починається другий раунд ставок, і гравці приймають рішення, ґрунтуючись на комбінації своїх кишенькових карт і спільних карт на столі.
3. Терн - на стіл додається четверта загальна карта. Починається третій раунд ставок.
4. Рівер - на стіл викладається п’ята і остання загальна карта. Починається четвертий і останній раунд ставок.
5. Шоудаун - гравці показують свої карти, і визначається переможець, що має найкращу комбінацію [7].

Гра розпочинається з префлоп раунду де двоє гравців, розташованих ліворуч від дилера роблять перші ставки (блайнди). Такі ставки називаються малий та великий блайнд відповідно, які робляться до роздачі карт. Ці обов’язкові ставки створюють початкову дію та стимулюють інших гравців продовжити гру. Зазвичай

малий блайнд становить половину від великого блайнду. Наприклад для таких блайндів найчастіше використовують номінали один та два або співвідношення один до двох. Інші гравці дивляться на дві свої карти, після чого починається перше коло ставок за годинниковою стрілкою де гравці вирішують чи брати участь в грі. Кожен гравець може зрівняти ставку (прийняти ставку попереднього гравця та поставити стільки ж фішок), підвищити ставку (збільшить поточну ставку на певну кількість, яка була обговорена до початку гри або на довільну кількість для безлімітної гри) або скине карти (відмовиться грати далі та не робитиме ставок)

Як тільки всі ставки зрівнялися розпочинається флоп раунд та дилером відкриваються перші три загальні карти. Після цього починається черговий раунд ставок. Гравцеві стає доступна нова можливість пропустити хід і залишитися в грі не міняючи ставки. Як правило, скидати карти в такій позиції немає сенсу, якщо не потрібно зрівнювати ставку із попереднім гравцем. Всі наступні гравці після попереднього повинні підтримати ставку або підняти її або скинути карти. Гравець також може підняти ставку, яка вже була піднята до нього.

Після того, як всі гравці зрівняють свої ставки на флопі, з'являється четверта загальна карта та розпочинається терн раунд. Механізм ставок повторюється, як на флопі. Коло ставок триває до тих пір, поки всі активні гравці (які не скинули свої карти) не зрівняють ставки.

Рівер це останній раунд та остання можливість зробити ставку починається після того, як відкривається остання п'ята відкрита карта. За його підсумками гравці, що залишилися розкривають карти в шоудаун раунді де переможця визначають за силою комбінації п’яти карт складеної із семи доступних (дві закриті та п’ять загальних) карт або єдиного гравця, який не скинув свої карти та залишився в грі до кінця.

Усього в покері існує десять комбінацій, продемонстровані в таблиці 1.1, які мають власну силу. Існує закономірність – чим більша сила комбінації тим менший шанс її отримання. Чимало професійних гравців за всю свою кар'єру, яка може тривати не одне десятиліття, жодного разу не вдається зібрати хоча б одну із трьох найсильніших комбінацій [8].

Таблиця 1.1 – Покерні комбінації за зростаючою силою

Назва комбінації	Карти для формування комбінації	Приклад
Старша карта	Якщо гравець не склав жодної комбінації оцінюється карта з найбільшим рангом	3♠-J♦-A♥-10♦-K♠
Пара	Дві карти з однаковим рангом	7♠-7♥-5♣-J♦-K♦
Дві пари	Дві пари карт з однаковим рангом	6♥-9♦-6♠-9♠-2♦
Трійка (сет)	Три карти одно рангу	K♠-K♥-K♣-3♦-5♥
Стріт	П’ять карт поспіль згідно рангу любих мастей	6♦-7♠-8♥-9♠-10♥
Флеш	П’ять карт будь-якого рангу однієї масті	6♥-A♥-K♥-9♥-4♥
Фул-хаус	Об’єднані комбінації трійки та пари	6♥-6♦-A♦-A-♥-A♠
Каре	Чотири карти одного рангу	Q♦-Q♠-**Q♥**-Q♣-**3♦**
Стріт-флеш	П’ять карт поспіль згідно рангу та всіма картами однієї масті	2♦-3♦-4♦-5♦-6♦
Флеш рояль	Карти від десятки до туза однієї масті	10♠-**J♠-Q♠-K♠**-A♠

1.2 Аналіз додаткових базових знань для навчання

Окрім базових правил та комбінацій, щоб стати успішним гравцем варто звернути увагу на ознайомлення з додатковими навичками та знаннями, які можуть допомагати аналізувати стан гри.

Простий розрахунок сили карт гравця - це аналіз карт, які є у гравця, та поточних відкритих карт. Краща оцінка враховує кількість гравців, які ще залишилися в грі, відносне положення гравця за столом та історію ставок для поточної гри. Ще точніший розрахунок враховує ймовірності для кожної можливої руки супротивника, засновані на ймовірностях кожної руки, яка може бути зіграна до поточної точки в грі.

Потенціал карт обчислює ймовірність того, що комбінація із можливих наявних карт покращиться для перемоги, або що така комбінація програє після

того, як з'являться майбутні відкриті карти. Наприклад, карти на руках гравця, які містить чотири карти однієї масті, може мати низьку силу, але водночас має доволі хороший потенціал для розвитку комбінації та покращення шансі на перемогу флешем, коли з'являться додаткові відкриті карти. Навпаки, карти з високою парою можуть очікувати зниження сили, якщо для протилежних карт на руках в гравців доступно багато варіантів досягнення потрібної комбінації. Мінімально потенціал це комбінація карт у руці та поточних відкритих карт. Проте кращий розрахунок використовуватиме всі додаткові фактори, що описані в обчисленні сили [9].

Блеф дозволяє гравцеві вигравати зі слабкою рукою та створює сумніви у противників, збільшуючи таким чином шанси на перемогу у наступних роздачах вже з сильною рукою. Блеф є необхідним для успішної гри. Мінімальні знання теорії ігор можуть бути використані для обчислення теоретично оптимальної частоти блефу в певних ситуаціях. Мінімальна система блефу блефувала б цей відсоток без розбору. На практиці слід враховувати й інші фактори (наприклад, потенціал руки). Більш продумана та краща така система виявляє вигідні можливості для блефу, визначаючи приблизну силу руки противника і передбачаючи ймовірність скидання ним карт та виходу з гри.

Непередбачуваність дій ускладнює противникам формування точного уявлення про обрану гравцем стратегію. Змішування стратегій (іноді різне поводження в одній і тій самій ситуації) допомагає приховувати певну інформацію про потенціал та силу поточних наявних карт. Варіюючи стиль гри з часом, можна змусити противників робити хибні розрахунки та робити помилки, ґрунтуючись на неправильних переконаннях.

Моделювання суперника визначає ймовірний стан руки (дві закриті карти гравця) суперника. Мінімальне моделювання такого типу здебільшого може без проблем використовувати єдину типову універсальну модель для всіх суперників. Таке моделювання можна покращити, змінюючи ці ймовірності на основі особистих ігор, ставок, роздач та зібраної статистики для кожного суперника окремо. Існує декілька інших ідентифікованих характеристик, які деколи можуть бути не обов’язковими для гри на досить високому рівні, але можуть бути необхідними для гри світового класу. У сукупності ця концепція моделювання є

частиною однієї загальної стратегії для поведінки під час роздачі та відкритті карт, яка визначає, чи будуть скидатись карти, зрівнювати або піднімати ставки в конкретних ситуаціях [10].

Багато основних покерних стратегій зосереджені на фундаментальних поняттях, таких як позиція та розмір ставки. Усі ці покерні поради корисні, але новачки, як правило, розглядають їх більш ізольовано. Іншими словами, вони дивляться на одну концепцію стратегії для Техаського холдему як на відмінність від іншої. Реальність така, що насправді все взаємопов'язане між собою. Не можна думати про позицію без врахування позицій інших гравців. Не можна думати про розмір ставки без шансів банку та неявних шансів. Тому все має значення. Звичайно, не можна сказати, що вам потрібно знати всі найкращі покерні стратегії просто тут і зараз. Мова йде про потрібність об'єднати окремі поради, щоб розробити загальну оптимальну стратегію.

Одна з найпростіших, але найскладніших у виконанні покерних порад, яку можна дати новачкам, це не допустити тильту. Перехід у тильт означає, що гравець втрачає контроль над своїми емоціями. Це змушує помилятись та приймати необдумані та погані рішення і значно зменшує всі шанси на перемогу. Щоб запобігти таким ситуаціям потрібно завжди зберігати спокій та контролювати емоції, наскільки це можливо. Ніколи не варто забувати, що покер призначений не для азарту, а саме для задоволення та розвитку логічного і математичного способу мислення. Якщо отримати поганий ритм гри, потрібно як найшвидше позбутись його та продовжувати грати. Кожен хід, який робиться покері, повинен бути зосереджений на його довгостроковому потенціалі та систематичному використанні [11].

1.3 Огляд і аналіз існуючих рішень

Існуючі рішення у сфері покерних стратегій та програмного забезпечення для навчання гри в покер охоплюють максимально можливі різноманітні методи моделювання суперників, оптимізації, ймовірного розвитку подій, отримання тієї чи іншою та аналітичного розрахунку кожної гри для порівняльної статистики.

Розглядаючи сучасні підходи, важливо звернути увагу на основні алгоритми, що використовуються для визначення оптимальних ставок та реалізації блефу. Аналіз цих методів дозволяє виявити їхні сильні та слабкі сторони, що, у свою чергу, сприяє визначенню напрямків для подальшого вдосконалення та розвитку нових рішень.

Огляд рішень включає технології, які застосовуються для аналізу дій суперників та оптимізації ставок. Виявлення ефективних методів та вказівка на можливі покращення дозволить досягти більш високого рівня гри. Крім того, аналіз основних підходів та алгоритмів сприяє виявленню сильних і слабких сторін наявних методів, що дозволяє знайти шляхи для подальшого розвитку та вдосконалення покерних стратегій. Цей аналіз стане основою для створення програмного модуля для навчання гри в покер. Модуль буде враховувати найкращі практики та інноваційні підходи у сфері покерних стратегій, що дозволить новим гравцям ефективно вдосконалювати свої навички гри.

1.3.1 Advanced Poker Training

Advanced Poker Training – веб-сайт, де можна активно розпочинати своє знайомство з світом покеру, розкрити свій потенціал і підняти свою гру на новий рівень. Платформа пропонує різноманітні можливості для навчання та покращення навичок у грі в покер, незалежно від попереднього досвіду і рівня майстерності [12].

На сайті є доступ до матеріалів та курсів для навчання грі в покер та його варіаціях. Інтегровано систему інтерактивних симуляцій з опонентами, які допоможуть вам відчути атмосферу реальної гри та вдосконалити свої стратегічні навички та календарний план навчання.

Advanced Poker Training використовує платний план по справжньому хорошого навчання та розуміння всіх тонкощів гри. З безкоштовних варіантів доступні лише деякі навчальні матеріали в текстовому форматі та два інструменти для роботи з картами.

Перший з них це - розрахунок шансів на перемогу, відображений у відсотках, зображений на рисунку 1.1, який дозволяє вручну вибрати карти які є в гравців та

відкриті карти на столі, щоб відображати шанси того чи іншого гравця в одному з раундів гри. Таке рішення зможе допомогти проекспериментувати певні роздачі карт та здобути базові знання розрахунку шансів на перемогу але водночас такий підхід не є ефективним при грі в реальному часі через потребу витрачати обмежений час на заповнення карт, а також потребі знати карти суперниках.

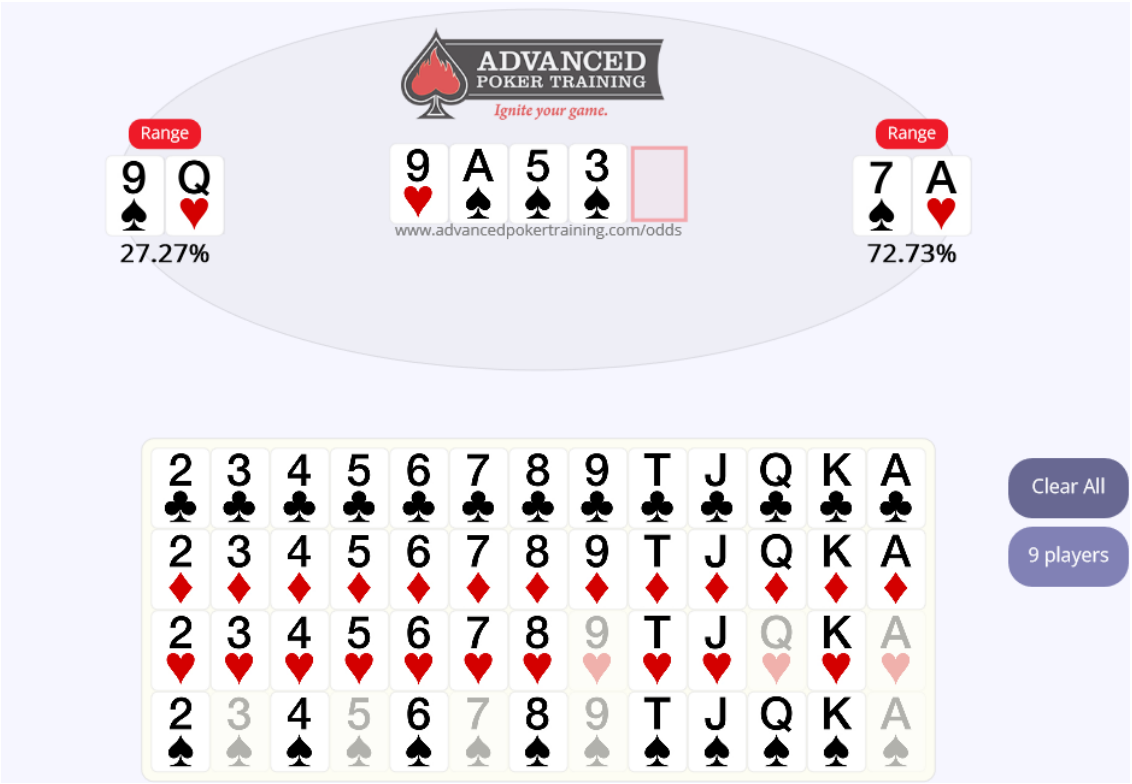


Рисунок 1.1 – Розрахунок шансів на сайті “Advanced Poker Training”

Другий інструмент для роботи з Шов-фолд стартегією, зображеній на рисунку 1.2, яка є дуже заплутаною для початків оскільки базується на етапі гри, при якому як мінімум однин з гравців перебуває на межі вильоту з гри та мусить використати всі доступні йому фішки щоб можливо залишитись в грі або скинути наявні карти. Є можливість вибрати для кого саме із гравців виконувати відсотковий розрахунок продовження гри або скидання карт із врахування вже зробленого ходу іншого гравця. Розрахункова формула також базується на інформації коли відомі карти суперника, додатково робота інструменту працює лише на імітації двох гравців.

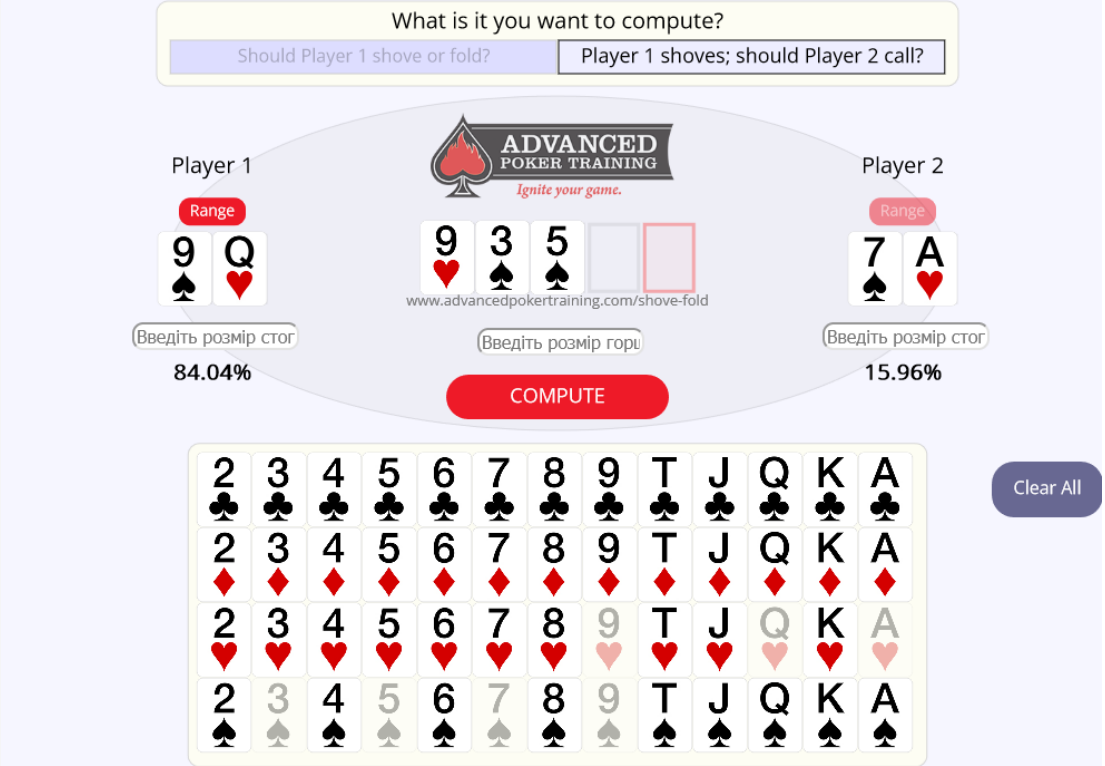


Рисунок 1.2 – Розрахунок Шов-фолд на сайті “Advanced Poker Training”

1.3.2 PokerSnowie

PokerSnowie — це платне програмне забезпечення, зображене на риснку 1.3 на основі штучного інтелекту для безлімітного холдему. Програма містить навчальні елементи та добре навчена грати в безлімітний покер і може ефективно грати як з короткими стеками, так і з дуже глибокими. PokerSnowie базується на штучних нейронних мережах, які оцінюють оптимальні дії для конкретних покерних ситуацій. Нейронні мережі добре інтерполюють дані, тому якщо вони навчилися грати у двох різних ситуаціях, вони зазвичай дають хороші результати в третій, схожій на вивчені [13].

Однак, нейронні мережі не є досконалими і іноді можуть давати неправильні результати, як у вивчених ситуаціях, так і в інтерпольованих. Таким чином, PokerSnowie не є ідеальним і теоретично може бути переможений ідеальною стратегією. Наприклад використовувати лише один розмір ставки для всього діапазону карт у певній ситуації, як робить більшість професійних гравців. Це означає, що розмір ставки буде однаковим, незалежно від того, які карти має гравець. Це перевага, оскільки розмір ставки не вказує на силу рук опонентам.

Гравець, який завжди ставить багато і мало на однакові типи карт програє, як тільки опоненти дізнаються про це.

Рисунок 1.3 – Інтерфейс програми “ PokerSnowie ”

Теоретично, можна побудувати кілька діапазонів для різних розмірів ставок, збалансованих між собою, що буде трохи кращим, ніж використання одного розміру. Але цей підхід дуже складний як для PokerSnowie, так і для людей через вимагання постійних математичних обчислень та потребу пам’ятати всі відкриті під час гри карти.

Помилки в оцінках при діях опонентів які не відповідають оптимальній теорії ігор теж один із недоліків даної програми. Коли гравець явно грає не за стратегією даної теорії, використовуючи ходи, які зазвичай математично вважаються неправильними, оцінки PokerSnowie стають доволі хибними.

1.3.3 DriverHUD 2

DriverHUD 2 — це потужна програма для аналізу покеру, яка допомагає

гравцям оптимізувати свою гру завдяки глибокому аналізу ігрових даних. Вона підходить для використання лише на найпопулярніших покерних онлайн платформах і забезпечує гравців цінною інформацією в режимі реального часу. Програма відображає статистику та тенденції суперників прямо на ігровому столі за допомогою налаштування системи HUD (Heads-Up Display), зображеної на рисунку 1.4 [14].

Рисунок 1.4 – Налаштування HUD в “DriverHUD 2”

Основні функції програми: користувацькі вікторини, де грвець створює свої власні сценарії гри на будь-який аспект гри: префлоп, гра на терні тощо. Звіт про неправильне використання карт, де DriverHUD 2 покаже будь-які карти на руках гравця, які були зіграли неправильно, базуючись на ваших оптимальній теорії ігор.

DriverHUD 2 не є безкоштовною програмою. Вона пропонує різні плани підписки, які залежать від обсягу функцій та тривалості використання. Існують

місячні, річні та довгострокові підписки, що дозволяють користувачам обрати найзручніший для них варіант. Для нових користувачів зазвичай доступна демонстраційна версія, яка дає можливість протестувати програму перед купівлею.

1.4 Постановка задачі

Після аналізу існуючих аналогів, їх рішень та певних обмежень під час використання можна сформувати рішення для вирішення поставленого завдання, а саме - програмний модуль, що розроблятиметься, має використовувати для навчання штучний інтелект та грати в покер, використовуючи велику кількість симуляційних ігор. Це дозволить оптимально розраховувати математичні аспекти гри та приймати стратегічні рішення на основі отриманих даних. Здатність зробити симуляцію більш ніж для тисячі ігор дозволить програмі отримати широкий спектр ситуацій та можливостей, що сприятиме покращенню навичок штучного інтелекту в грі та процесі навчання гравця.

Окрім того, модуль має надавати можливість гравцям грати проти комп'ютера, який використовує навчені дані для прийняття рішень. Це створить можливість для користувачів випробувати свої навички в грі зі суперником у вигляді комп’ютера.

Під час кожного раунду програма аналізує карти на руках гравців та на столі, оцінює поточні комбінації гравців та їхні шанси на перемогу. Ця інформація використовується для прийняття оптимальних рішень щодо наступних ходів. В кінці кожної гри модуль відображає два графіки: перший показує зміну комбінацій гравців протягом раундів, а другий - зміну їхніх шансів на перемогу. Це допоможе користувачам отримати уявлення про те, як розвивається гра та які стратегії можуть бути найбільш ефективними в різних ситуаціях.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

2.1 Загальна структура проектованого модуля

Основний компонент програм - це класи для представлення гравців та гри. Там буде зберігатись інформація про гравців, включаючи їхні карти, поточний стан гри, раунд та стратегію. А також буде містити основну логіку гри, включаючи створення та роздачу карт, раунди ходів, визначення переможця та інші ключові аспекти гри в покер [16].

В першу чергу необхідно забезпечити симуляцію ігор по якій можна навчити логіку противника, в контексті програмного модуля це буде комп'ютер. Під час симуляцій безліч ігор зібрані дані можна ретельно проаналізувати та завдяки математичним методам та їх аналізу в результаті отримати власний штучний інтелект, який буде правильно реагувати на різні ігрові ситуації. В процесі навчання це дозволяє гравцю початківцю краще орієнтуватись при тій чи іншій ситуації та допомагає йому приймати вірні рішення для досягнення перемоги. Для цього використовується зв'язок між роздачею карт, раундами гри, математичними розрахунками та файлами де зберігатимуться дані. На рисунку 2.1 зображено принцип роботи симуляцій ігор програмою та їх використання.

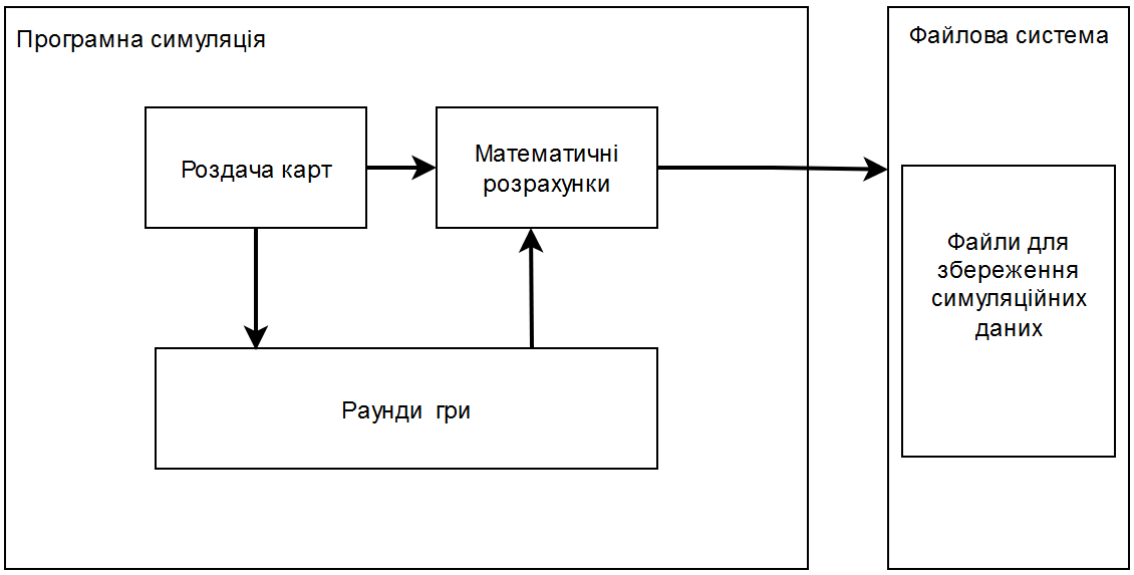


Рисунок 2.1 – Принцип роботи програмних симуляцій

Одним із найважливіших компонентів є система оцінки покерних комбінацій. Спеціальна частина модуля буде відповідальною за визначення сили карт гравців, порівняння їх між собою та визначення переможця раунду. Цей клас повинен враховувати всі можливі покерні комбінації та їхню силу в математичному еквіваленті та можливість їх покращення при відкритті наступних карт. На рисунку 2.2 продемонстровано схему аналізу карт та оцінки комбінацій для кожного раунду гри.

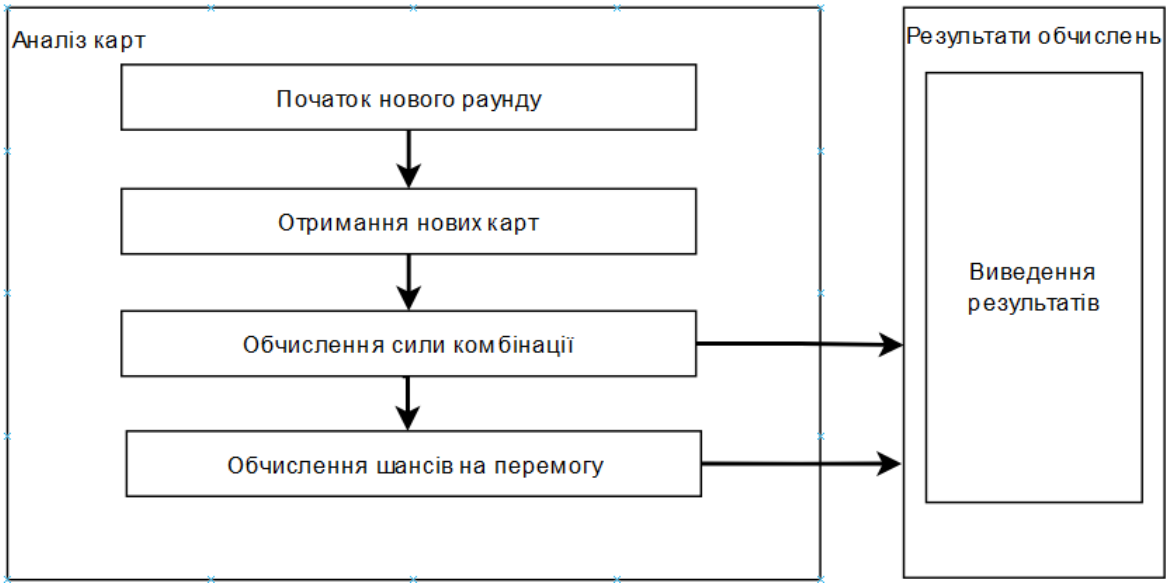


Рисунок 2.2 – Принцип роботи аналізу карт під час раунду

Інтерактивність та тренування є ключовим елементом для навчального модуля. Тому програма використовує режим гри з комп'ютером та надає зручний доступ до інформації. Програма відображає поточні карти гравців, шанси на перемогу, інформацію про можливі комбінації та іншу важливу інформацію для процесу навчання, а також дозволяти гравцям робити ходи та приймати рішення протягом раундів. На рисунку 2.3 зображено загальний алгоритм роботи симуляції гри користувача з комп'ютером [17].

Для надання користувачу детального аналізу його ігор гри та задля допомоги у вдосконаленні стратегій, необхідний процес для візуалізації даних. Ця частина програми буде відповідальною за створення графіків, які показують як мінялись шанси на перемогу під час раундів та які максимальні можливі комбінації можна

Було утворити під час гри, та інші корисні статистичні дані. Ці графіки допоможуть краще розуміти свої сильні та слабкі сторони, а також побачити прогрес у навчанні. На рисунку 2.4 зображено процес побудови діаграми на основі даних по раундах.

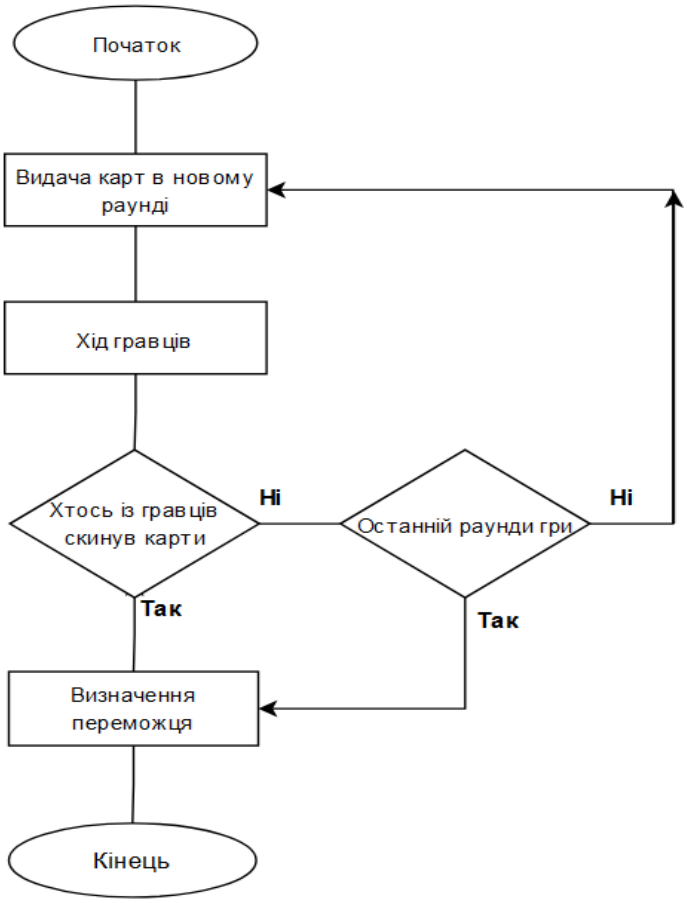


Рисунок 2.3 – Загальний алгоритм гри

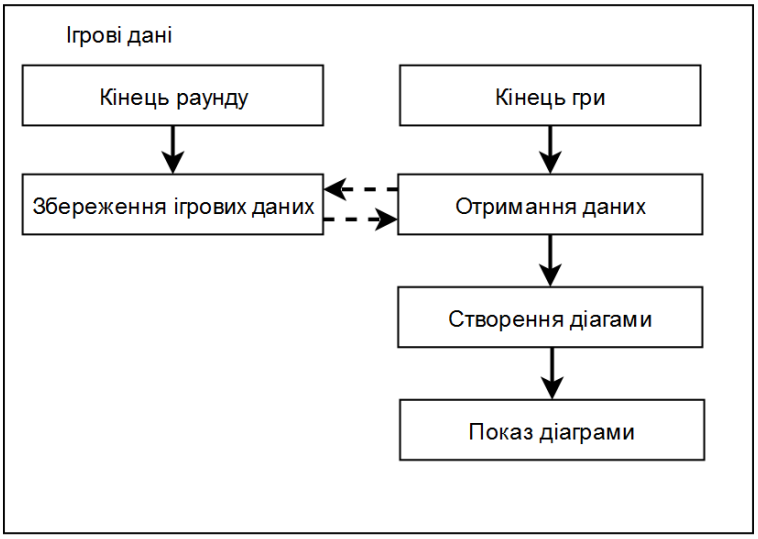


Рисунок 2.4 – Процес побудови діаграм статистичних даних про раунди

2.2 Математичне забезпечення

Покер це не історія про везіння, а про безліч як складних так і простих математичних обчислень. Основним завданням математичної частини гри є процес звикнення до обчислення чисел в реальному часі, що потребує потребує практики. Для гравця початківця не є обов'язковим одразу розглядати низку складних термнів та формул із вищої математики, теорії ймовірності, оптимальної теорії ігор тощо, а розуміння базових визначень та основних формул із кожного згаданого розділу математки та їх розрахунок в голові, поки це не стане для нього звичайною справою під час гри, адже це є важливим для навчання та успіху за покерним столом [18].

Комбінаторика в покері - це практика розбиття діапазонів і підрахунку індивідуальних комбінацій для прийняття кращих рішень [19]. Формула для знаходження кількості комбінацій k об'єктів, які ви можна вибрати з набору n об'єктів, виглядає так:

$$Number\ Combinations = \frac{n!}{k!(n-k)!}$$

де n - загальна кількість елементів у множині, k - кількість елементів, які обираються з цієї множини.

Перед флопом існує сто шістдесят дев'ять різних покерних рук, але, з урахуванням мастей, кількість конкретних двокарткових покерних комбінацій у колоді з п'ятдесяти двох карт становить одна тисяча триста двадцять шість, які обчислюються за формулою:

$$Hand\ Combinations = \frac{52!}{2!(52-2)!}$$

Формула для обчислення кількості комбінацій пар кишенькових карт в діапазоні гравця виглядає так:

$$Combination\ of\ PP = \frac{x * (x-1)}{2}$$

де x - це кількість доступних карт, які залишилися в колоді.

Для будь-якої пари кишенькових карт це обчислення виглядає наступним чином:

$$Combination\ of\ PP = \frac{4 * (4-1)}{2} = \frac{12}{2} = 6$$

Існує тринадцять різних рангів пар і шість різних способів скласти пару кишенькових карт будь-якого рангу, отже, всього є 13 * 6 = 78 комбінацій кишенькових пар [20].

Формула для обчислення кількості комбінацій для непарних рук:

$$Combination\ of\ Unpaired\ Hands = (x * y)$$

де x і y - кількість доступних карт, що залишилися в колоді для кожного рангу. Візьмемо для прикладу карти x рангом шість та дев'ять. У колоді є чотири шістки та чотири дев'ятки, 4 * 4 = 16, отже, є 16 комбінацій для непарної руки із картами 6 та 9 (і будь-якої іншої непарної руки) з доступних.

Тепер розглянемо, як розподіляється діапазон карт суперника на конкретному флопі A♣J♥9♥. В цьому випадку в суперника є всього сто п'ятдесят сім комбінацій сформованих рук та сім комбінацій для можливих пар, як зазначено нижче.

Пара AA (два тузи): A♣ є на флопі, і ми блокуємо A♦ (це означає, що вона в нашій руці), тому залишається лише одна комбінація A♠A♥.

Пара JJ (два валета) або 99 (дві дев'ятки): можна порахувати використовуючи формулу для кишенькових пар для JJ:

$$\text{Combinations of JJ} = \frac{3 * (3-1)}{2} = 3$$

J♥ є на флопі, тому залишилося лише три можливі варіанти вилучення дам у діапазоні суперника: J♠, J♥ та J♣. Той самий процес використовується для 99, отже, у нас є загалом одна комбінація AA, три комбінації JJ та три комбінації 99, що в сумі становить сім комбінацій можливих пар.

Дві пари (десять комбінацій) AJ (туз та валет): A♣ та J♥ є на флопі, ми блокуємо A♦, тому залишилося два тузи та три валети які обчислюють за формулою вказаною вище:

$$\text{Combinations of AJ} = (2 * 3) = 6$$

A9 (туз та дев’ятка): A♣ та 9♥ є на флопі, тому A♣9♣ та A♥9♥ неможливі. Ми блокуємо A♦, тому суперник не може мати A♦9♦, що робить A♠9♠ єдиною комбінацією A9, залишеною у діапазоні суперника.

J9 (валет та дев’ятка): J♥ та 9♥ є на флопі, тому в діапазоні суперника є три комбінації J9s (валет та дев’ятка) де s означає однакову масть для обох карт.

Топ пара (тридцять вісім комбінацій) AK (туз та король): залишилося два тузи та чотири короля:

$$\text{Combinations of AK} = (2 * 4) = 8$$

Такий самий процес розрахунку можна зробити і для AQ (туз та дама) та A10 (туз та десятка), що в загальної кількості дає двадцять чотири комбінації. A8s (тузи та вісімки однакових мастей) A2s (туз та двійка однакових мастей): A♣ є на флопі, і ми блокуємо A♦, тому залишилося лише дві масті для кожного з семи різних рангів тузів: 7 * 2 = 14 комбінацій.

Середні Пари (тридцять комбінацій) KJ (король та валет), QJ (дама та валет): двадцять чотири комбінації (повторюється той самий процес, що використовується для топ пар). J10s (валет та десятка однієї масті), J8s (валет та вісімка однієї масті) аналогічно отримує шість можливих комбінацій.

Нижня пара (п’ятнадцять комбінацій) формується перебором K9s (король та дев’ятка однієї масті), Q9s (дама та дев’ятка однієї масті), 109s(десятка та дев’ятка однієї масті), 98s(дев’ятка та вісімка однієї масті), 97s (дев’ятка та сімка однієї масті) та такою ж формулою.

Кишенькова пара 1/2 (дванадцять комбінацій) KK (два королі), QQ (дві дами): кишенькова пара 1/2 означає кишенькові пари між верхньою і середньою картою, які є на флоп раунді.

Кишенькові пари 2/3 (шість комбінацій) 10 (дві десятки): кишенькова пара 2/3 означає кишенькову пару між середньою та нижньою картою, які є на флоп раунді.

Непарні пари (тридцять дев’ять комбінацій) 33 (дві трійки): У нас є 3♦ у руці, тому залишилося лише три комбінації для пари трійок. Інші пари (тридцять шість

комбінацій) – де ранги - 88 (дві вісімки), 77 (дві сімки), 66 (дві шістки), 55 (дві п’ятірки), 44 (дві четвірки), 22 (дві двійки).

Дроу (покращення комбінації). У суперника є сімдесят дві руки для покращення, які складаються такими можливими комбінаціями як дроу на флеш (вісім комбінацій). Дроу на стріт-флеш: шість комбінацій: K♥Q♥, K♥T♥, Q♥T♥, Q♥8♥, T♥8♥, 8♥7♥. Нат флеш дроу (префікс нат використовується коли гравець має чотири карти однієї масті) також сюди входить пара, яка вже була врахована при підрахунку топ пар: A♥K♥, A♥Q♥, A♥T♥, A♥8♥, A♥7♥, A♥6♥, A♥5♥, A♥4♥, A♥3♥, A♥2♥. Другий нат флеш дроу: одна комбінація: K♥8♥ та інші доступні флеш дроу теж одна комбінація: 7♥6♥.

Вісім аутів (карта, яка покращує руку гравця до потенційно виграшної комбінації.) на стріт (вісімнадцять комбінацій) Q10 (дама та десятка): є шістнадцять комбінацій, але Q♥T♥ вже була врахована, тому залишилося лише п’ятнадцять комбінацій та 108s (десятка та вісімка однієї масті), де вже враховано 10♥8♥, тому залишилося лише три комбінації з іншими мастями.

Чотири аути на стріт (тридцять шість комбінацій). Для KQ (король та дама), K10 (король та дестка) є шістнадцять комбінацій кожної з них, але вже було враховано K♥Q♥ і K♥T♥ отже загалом залишилось лише тридцять комбінацій. Q8s (дама та вісімка однієї масті), 87s (вісімка та сімка однієї масті) де вже враховано Q♥8♥ і 8♥7♥, тому залишилося лише три масті для кожної з пари, загалом це шість комбінацій.

Порожні руки (шість комбінацій) Враховуючи відсутність карт на флопі (так як A♦, 3♦, A♣, J♥ і 9♥ недоступні), діапазон суперника скорочується із триста шести до двісті двадцять п’ять комбінацій, де двісті дев’ятнадцять комбінацій вже є сформованими руками або дроу покращеннями, тому в діапазоні суперника залишилося лише шість комбінацій для порожніх рук: K♣8♣, K♦8♦, K♠8♠, 7♣6♣, 7♦6♦, 7♠6♠ [21].

2.3 Інформаційне забезпечення

Невід’ємною частиною для навчання гри в покер є правильна оцінка інформації, яку треба використати та отримати. В цьому підрозділі розглядатиметься основна вхідна та вихідна інформації, потрібні для роботи програми та правильного її використання в контексті поставленого завдання. Вхідна та вихідна інформації міститься в таблицях 2.1 та 2.2 відповідно.

Таблиця 2.1 – Вхідна інформація програмного модуля

Категорія	Параметр
Початкові налаштування гри	Кількість симуляцій
Початкові налаштування гри	Кількість гравців
Інформація для кожної гри	Дії гравця
Інформація для кожної гри	Карти гравців
Інформація для кожної гри	Карти на столі
Навчальні дані	Результати симуляцій

Опис параметрів вхідної інформації.

1. Початкові налаштування гри:
   - Кількість симуляцій: число, що вказує на кількість ігор для яких потрібно симулювати для навчання (програма використовує п’ять тисяч таких ігор).
   - Кількість гравців: ціле число, що визначає кількість гравців у грі (програма використовує користувача та комп’ютерний бот, тобто два гравці).
2. Інформація для кожної гри:
   - Дії гравця: масив дій (ходів), які можуть виконувати гравці.
   - Карти на руках гравців: масив карт, які отримують гравці на початку гри.
   - Карти на столі: масив карт, які розкриваються на столі в різних раундах.
3. Навчальні дані:
   - Результати попередніх симуляцій: Дані, отримані під час симуляцій, які використовуються для навчання бота (наприклад, ймовірності комбінацій, статистики виграшів)

Таблиця 2.1 – Вихідна інформація програмного модуля

Категорія	Параметр
Під час гри	Карти на руках гравців
Під час гри	Карти на столі
Під час гри	Шанси кожного гравця на перемогу
Під час гри	Комбінації карт гравців
Після гри	Результати гри
Після гри	Діаграми зміни шансів на перемогу
Після гри	Діаграми зміни максимальних комбінацій

Опис параметрів вихідної інформації

1. Під час гри:
    - Карти на руках гравців: виводиться для кожного гравця його карти.
    - Карти на столі: виводяться карти, які поступово відкриваються на столі під час раундів.
    - Шанси кожного гравця на перемогу: відсоткове значення обчислене математичною формулою, що вказує ймовірність перемоги кожного гравця в поточному раунді.
    - Комбінації карт гравців: математично розраховане числове значення комбінації карт гравця.
2. Після гри:
    - Результати гри: інформація про те, хто виграв та які комбінації карт у кожного гравця.
    - Діаграми зміни шансів на перемогу: діаграми, що показують, як змінювались шанси на перемогу кожного гравця протягом гри.
    - Діаграми зміни максимальних комбінацій: діаграми, що показують, як змінювались максимальні комбінації карт гравців на кожному етапі гри.

2.3 Використання модулів та бібліотек

Модулі та бібліотеки є важливою складовою в програмуванні. Вони є збірками функцій, методів та класів, які можна використовувати для виконання різноманітних завдань без необхідності писати код з нуля. До головних ключових переваг модулів та бібліотек належать:

1. Повторне використання коду: модулі та бібліотеки дозволяють використовувати функції та класи, які вже написані та протестовані іншими розробниками. Це зменшує кількість коду, який потрібно писати з нуля, та сприяє швидкому розвитку програм.
2. Розділення коду на логічні модулі: модулі дозволяють розділити програму на менші логічні частини, що полегшує розуміння та підтримку коду. Кожен модуль може виконувати конкретну функцію або реалізовувати певний функціонал.
3. Розширення можливостей мови програмування: завдяки бібліотекам, розробники можуть використовувати практично любу мову для різноманітних завдань, від аналізу даних до веб-розробки та машинного навчання. Це розширює потенційні можливості мови програмування та полегшує розвиток різноманітних програм [22].

Модуль для випадкових подій в контексті програмного модуля для навчання гри в покер є надзвичайно важливим компонентом. Він забезпечує можливість генерувати випадкові роздачі карт, випадкові дії гравців та інші випадкові події, які відтворюють реальність гри. Це дозволяє навчальній системі отримувати різноманітні сценарії гри для покращення ефективності навчання та розвитку стратегій гри.

Модуль для візуалізації даних є важливим інструментом для аналізу та відображення результатів гри в покер. Візуалізація може включати створення графіків розподілу карт, гістограм дій гравців, графіків зміни стратегій гри відносно різних умов гри тощо. Це допомагає аналізувати та зрозуміти динаміку гри, виявляти помилки та вдосконалювати стратегії.

Модуль для роботи з низькорівневим введенням є важливим для забезпечення інтерактивності гри в покер. Він дозволяє обробляти введення користувача в реальному часі, такі як вибір дії гравця, без очікування натискання клавіші Enter. Це дозволяє гравцям швидко реагувати на гру та покращує взаємодію з ігровим середовищем. В контексті поставленого завдання такий модуль чудово підійде для симулялцій ігор та навчання штучного інтелекту в більш швидкому темпі.

Узагальнюючи, разом ці модулі утворюють потужну основу для розробки програмного модуля навчання гри в покер. Вони допомагають створювати реалістичне інтерактивне середовище для навчання та ефективної аналітики гри.

3 ПРОГРАМНО-ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ

3.1 Програмні засоби реалізації модуля

Для реалізації програмно модуля навчання гри в покер була використана мова програмування Python. Python — це високорівнева мова програмування загального призначення, відома своєю простотою для написання програм та легкою читабельністю коду. Вона була створена Гвідо ван Россумом і вперше випущена в тисяча дев’ятсот дев’яносто першому році. Python є інтерпретованою мовою, що означає, що її код виконується безпосередньо інтерпретатором, а не компілюється у машинний код перед виконанням. Це робить процес розробки зручнішим, оскільки можна безпосередньо бачити результати змін у коді [23].

Python використовує динамічну типізацію і автоматичне управління пам'яттю, що значно спрощує розробку програм і дозволяє зосередитися на логіці, а не на технічних деталях. Бібліотеки та фреймворки надають потужні інструменти для роботи з даними, машинного навчання та візуалізації. Версія 3.12, яка є найновішою та стабільною версією буде використовуватись для розробки програми, включає в собі покращення продуктивності, нові функції та вдосконалення попередніх версій.

Python ідеально підходить для розробки програми для навчання гри в покер з кількох причин. По-перше, його простий і зрозумілий синтаксис дозволяє швидко розробляти та тестувати алгоритми, що особливо важливо в процесі створення інтелектуальних систем. По-друге, наявність потужних бібліотек для обробки даних аби створювати та налаштовувати моделі, які можуть аналізувати та передбачати ігрові ситуації.

Крім того, Python має велику спільноту розробників та багатий набір ресурсів, що спрощує пошук допомоги та рішень для різноманітних проблем, що можуть виникнути під час розробки. Це дозволяє швидше знаходити оптимальні методи та практики для реалізації складних функцій, таких як розрахунок ймовірностей, оцінка ризиків або оптимізація стратегій гри.

Для написання та редагування коду обрано Sublime Text 3. Це один із найвідомиших текстових редакторів із відкритим вихідним кодом, який відомий

своєю швидкістю, простотою використання та багатофункціональністю. Він підтримує роботу з багатьма мовами програмування, включаючи Python, та має велику кількість розширень і плагінів, які дозволяють розширити його функціонал для потреб конкретного проекту або розробника. Наприклад, розширення SublimeREPL дозволяє виконувати код Python безпосередньо в середовищі Sublime Text 3, що робить процес розробки та експериментування з кодом. Крім того, підтримка різних режимів роботи, таких як режим редагування та режим перегляду, сприяє зручності роботи з різними типами файлів та завдань [24].

Одна з основних переваг Sublime Text 3 - це його легкість налаштування та гнучкість. Користувач може налаштувати редактор згідно своїх власних потреб і використовувати різні теми оформлення та схеми виділення синтаксису для комфортної роботи. Крім того, він підтримує роботу з великими проектами завдяки можливості швидко шукати, замінювати та в цілому неймовірно зручно та швидко навігуватись по коду за допомогою гарячих клавіш.

Ще одна важлива перевага Sublime Text 3 - це його багатоплатформеність. Редактор доступний для операційних систем Windows, macOS та Linux, що дозволяє розробникам працювати на своїй улюбленій операційній системі

3.2 Реалізація програмного модуля

Першим етапом при розробці програмного модуля навчання гри в покер є забезпечення створення та перемішування колоди карт та іншими важливими маніпуляціями для робити.

Метод create_deck() створює стандартну колоду із п’ятдесяти двох карт для гри в покер. Він ініціалізує порожній список deck і потім додає до нього карти зі всіх чотирьох мастей (піки, черви, трефи, бубни) та всіх рангів (від 2 до 14, де 11-валет, 12 - дама, 13 - король, 14 - туз). Для кожної масті цикл for перебирає значення від 2 до 14 і додає об'єкт карту до колоди, використовуючи клас Card, передаючи в нього масті та значення.

Метод shuffle() перемішує колоду карт. Він використовує алгоритм перетасовки (простого обміну). Для цього виконується вкладений цикл for, який проходить через кожну карту у колоді. На кожній ітерації внутрішнього циклу генерується випадкове ціле число rand від 0 до 51, і карти на позиціях j і rand обмінюються місцями. Цей процес повторюється тричі (для більшої випадковості перемішування). Після цього перемішана колода повертається як результат методу.

```
def create_deck(self):
    deck = []
    suits = ['♠', '♥', '♣', '♦']
    count = 0
    for suit in suits:
        for i in range(2, 15):
            deck.append(Card(suit, i))
    return deck

def shuffle(self, deck):
    for i in range(0, 3):
        for j in range(0, 52):
            rand = int(random.uniform(0, 52))
            temp = deck[j]
            deck[j] = deck[rand]
            deck[rand] = temp
    return deck
```

Метод convert_to_display приймає числове значення карти (number) і перетворює його у відповідне відображення для візуалізації:

1. Якщо number менше 11, повертається саме число (це відповідає картам від 2 до 10).
2. Якщо number дорівнює 11, повертається символ "J" (валет).
3. Якщо number дорівнює 12, повертається символ "Q" (дама).
4. Якщо number дорівнює 13, повертається символ "K" (король).
5. Якщо number дорівнює 14, повертається символ "A" (туз).

Метод get_card_display повертає зручне для відображення представлення карти. Він створює список, що складається з двох елементів:

1. self.suit — масть карти (наприклад, ♠, ♥, ♣, ♦).
2. Виклик методу __convert_to_display для self.number — числового значення карти, перетвореного у відповідний символ (2-10, J, Q, K, A).

Таким чином, метод get_card_display повертає карту у форматі, що легко читається та відображається, наприклад, ["♠", "K"] для короля піків.

```
def get_card_display(self):
    return [self.suit, self.__convert_to_display(self.number)]


def __convert_to_display(self, number):
    if number < 11:
        return number
    elif number == 11:
        return "J"
    elif number == 12:
        return "Q"
    elif number == 13:
        return "K"
    elif number == 14:
        return "A"
```

Метод get_suit_number призначений для повернення числового значення, яке відповідає масті карти. Метод використовує значення масті (збережене в self.suit) та повертає відповідне числове значення від 0 до 3. Метою використання методу є отримання числового хначення із яким можна проводити математичні розрахунки для визначення комбінацій.

```
def get_suit_number(self):
    if self.suit == "♣":
        return 0
    elif self.suit == "♦":
        return 1
    elif self.suit == "♥":
        return 2
    elif self.suit == "♠":
        return 3
```

Коли створено методи для роботи з картами настає час створити програмну симуляцію. В такій симуляції буде відтворено п’ять тисяч ігор де комп’ютер обчислюватиме сили комбінацій та можливі варіанти для подальшого ходу. Зараз розглядатимуться осовні методи коду які відповідають за збирання на аналіз даних під час комп’ютерної симуляції.

Метод update оновлює ймовірності для різних станів рук (карт гравців) на етапах покет, флоп, терн і рівер. Перший кроком є визначення переможця.

```
if player == 2:
    local = winner * -1
else:
    local = winner
```

Якщо гравець 2 виграє, змінна local дорівнює -1 (припускаючи, що winner - це 1 або -1 в залежності від того, хто виграв). Якщо гравець 1 виграє, local дорівнює значенню winner.

Після визначення переможця дані зібрані протягом всіх раундів гри аналізуються та обчислюють середні значення на основі нових спостережень.

```
probPocket[pocket] = probPocket[pocket] + (local - probPocket[pocket]) / (encounteredPocket[pocket] + 1)
probFlop[flop] = probFlop[flop] + (local - probFlop[flop]) / (encounteredFlop[flop] + 1)
probTurn[turn] = probTurn[turn] + (local - probTurn[turn]) / (encounteredTurn[turn] + 1)
probRiver[river] = probRiver[river] + (local - probRiver[river]) / (encounteredRiver[river] + 1)
```

Метод save_results зберігає результати симуляції в текстові файли та виводить ці результати на екран.

```
def save_results():
    print("Pocket data...")
    for i in range(len(statesPocket)):
        p.write(str(statesPocket[i]) + ":" + str(probPocket[i]) + ":" + str(encounteredPocket[i]) + "\n")
        print(str(statesPocket[i]) + ": " + str(probPocket[i]) + " - " + str(encounteredPocket[i]))
    for i in range(len(statesFlop)):
        f.write(str(statesFlop[i]) + ":" + str(probFlop[i]) + ":" + str(encounteredFlop[i]) + "\n")

        print(str(statesFlop[i]) + ": " + str(probFlop[i]) + " - " + str(encounteredFlop[i]))
    for i in range(len(statesTurn)):
        t.write(str(statesTurn[i]) + ":" + str(probTurn[i]) + ":" + str(encounteredTurn[i]) + "\n")
        print(str(statesTurn[i]) + ": " + str(probTurn[i]) + " - " + str(encounteredTurn[i]))
    for i in range(len(statesRiver)):
        r.write(str(statesRiver[i]) + ":" + str(probRiver[i]) + ":" + str(encounteredRiver[i]) + "\n")
        print(str(statesRiver[i]) + ": " + str(probRiver[i]) + " - " + str(encounteredRiver[i]))
```

На рисунку 3.1 зображено як відбувається процес симуляції ігор та як після математчиних операцій комю’тер буде бачити інформацію для аналізу, що буде використанна для навчання комп’ютера в іграх з реальною людиною.

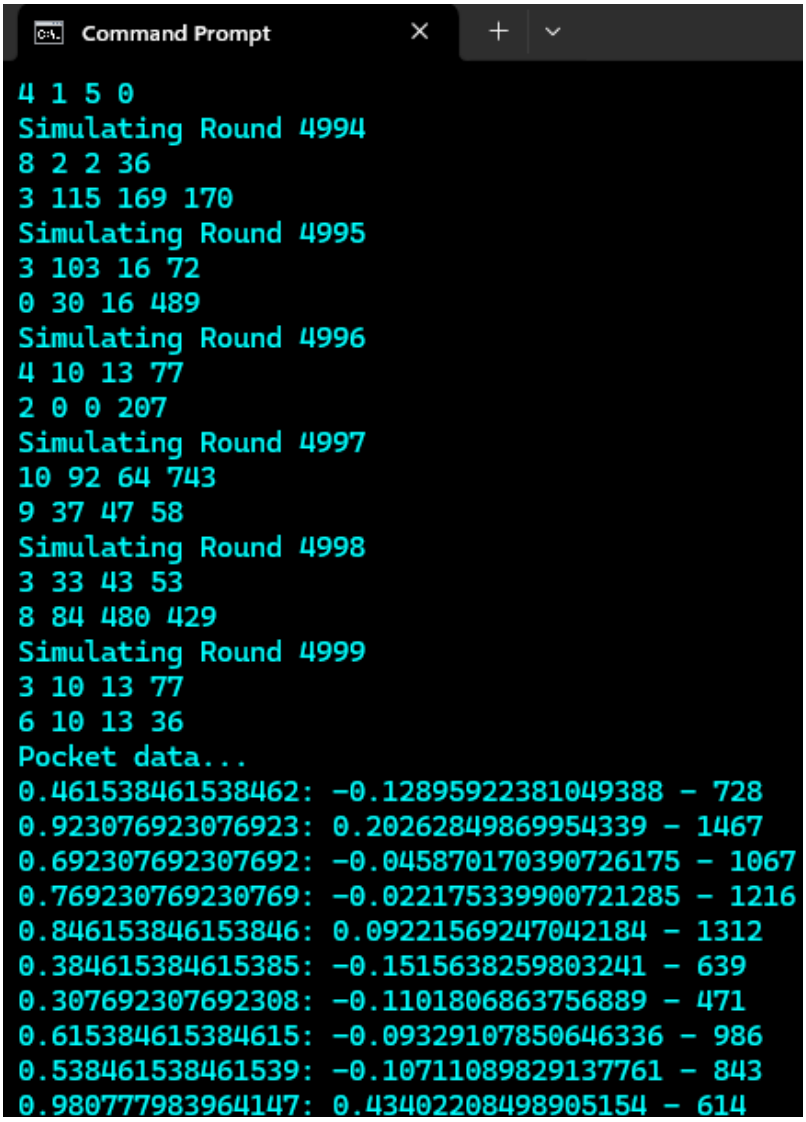


Рисунок 3.1 – Процес симуляції ігор

На рисунку 3.2 представлено діаграми різних етапів гри. Кожна діаграма відображає залежність між рівнем вмінь (позначено на вісі x) та ймовірність виграшу, пов'язану з кожним станом гри (позначено на вісі y). на різних етапах гри. Кожна точка на графіку позначає рівень для конкретної ситуації. Для візуального представлення обрано лінійний графік з точками (scatter plot).

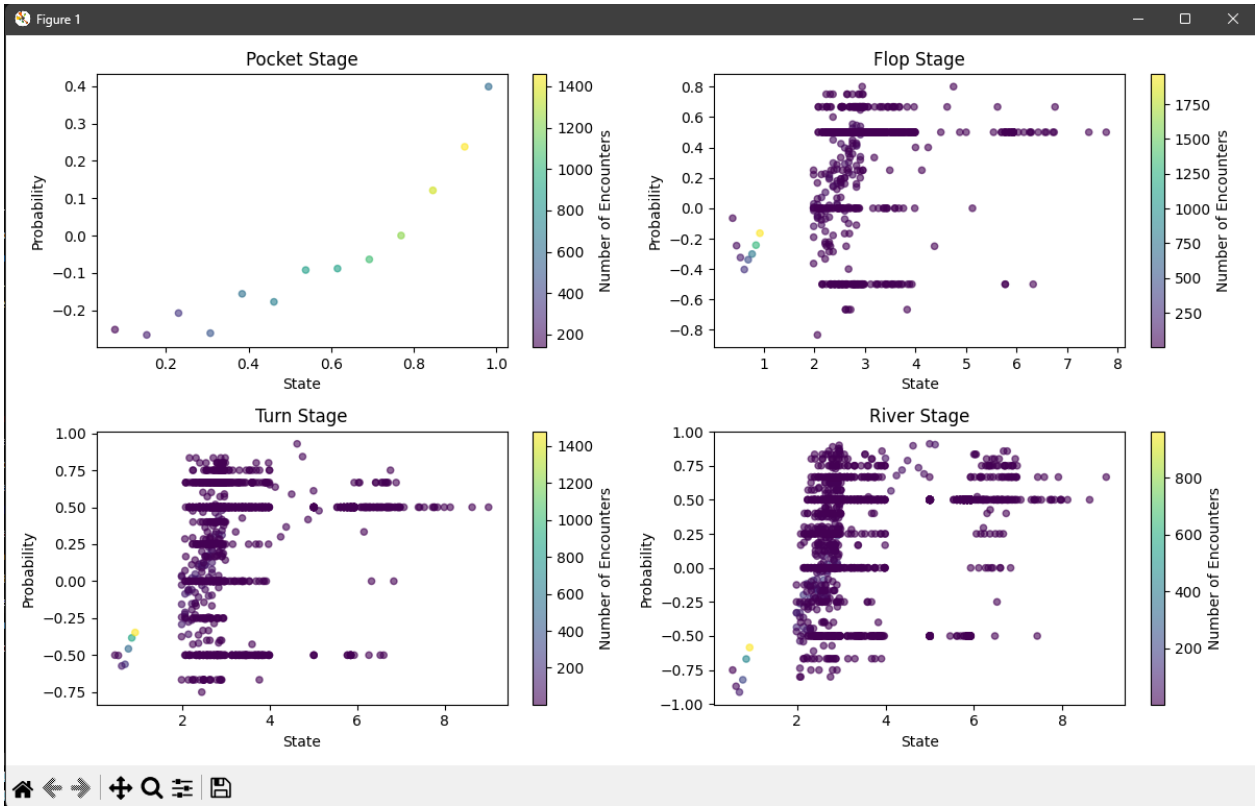


Рисунок 3.2 – Діаграми залежностей на різних етапах гри

Для отримання даних та побудови діаграм залежностей розроблено метод read_results, який зчитує дані із файлів в які зберігались симульовані ігри. В процесі підготовки до побудови діаграм прогрма залишає лише вірні дані в яких точно досягнуто повне навчання з правильним завершенням гри.

```
def read_results(filename):
    states = []
    probs = []
    encountered = []
    with open(filename, "r") as file:
        for line in file:
        parts = line.lstrip().split(":")
        if len(parts) == 3:
            state, prob, count = parts
            states.append(float(state))
            probs.append(float(prob))
            encountered.append(int(count))
return states, probs, encountered
```

Для побудови самих діаграм було використано бібліотеку matplotlib. Нижче наведено приклад основного коду для створення однієї із діаграм, її розміщення та відображення на екрані.

```
plt.figure(figsize=(12, 7))
plt.subplot(2, 2, 1)
plt.scatter(statesPocket, probPocket, c=encounteredPocket, cmap='viridis', s=20, alpha=0.6)
plt.colorbar(label='Number of Encounters')
plt.xlabel('State')
plt.ylabel('Probability')
plt.title('Pocket Stage')
```

plt.subplot(2, 2, 1) викликає функцію `subplot`, яка визначає розміщення підграфіка на головному графіку. В даному випадку створюється підграфік у вигляді таблиці 2 на 2, і цей блок коду вказує надроботою з першим підграфіком.

plt.scatter(statesPocket, probPocket, c=encounteredPocket, cmap='viridis', s=20, alpha=0.6) - ця команда викликає функцію `scatter`, яка створює діаграму розсіювання (scatter plot). Кожна точка на графіку відповідає певному стану гри і її розташування визначається за значеннями на вісіх x та y. Кольори точок визначаються за допомогою кольорової палітри 'viridis', а їх розмір встановлюється на 20 пікселів з прозорістю 0.6.

Команда plt.colorbar(label='Number of Encounters') додає колірну смугу (colorbar) до графіку, яка пояснює значення кольорів на графіку.

plt.xlabel('State') і plt.ylabel('Probability') - команди які додають підписи до відповідних вісей x та y, що допомагає зрозуміти, що саме вони представляють.

plt.title('Pocket Stage') додає заголовок до підграфіка, який вказує на те, що саме він відображає.

Метод choose_action призначений для вибору дії комп’ютера під час різних раундів гри. Комп’ютер робить вибір на основі аналіз

Логіка роботи:

- Функція перевіряє, чи належить значення strength до відповідного списку (valuesPocket, valuesFlop, valuesTurn або valuesRiver). Якщо ні, повертається значення -1, що означає помилку.

- Далі, за допомогою індексу `i`, функція знаходить індекс сили руки у відповідному списку ймовірностей.

- Якщо ймовірність менше 0.3, повертається значення -1, що вказує на здійснення дії фолд (відмова від гри).

- Якщо ймовірність знаходиться між 0.3 і 0.6, повертається значення 1, що означає здійснення дії продовження гри.

- Якщо ймовірність більше або рівна 0.6, повертається значення 2, що вказує на здійснення дії бет (підищеняя ставка).

Нижче наведено частину коду із методу для одного із раундів гри, інші раунди обчислюються аналогічно.

```
elif part == 2:
    if strength not in valuesTurn:
        return -1
    i = valuesTurn.index(strength)
    if probTurn[i] < .3:
        action = -1
    elif probTurn[i] < .6:
        action =  1
    elif probTurn[i] >= .6:
        action = 2
```

На основі вже розроблених частин програм, модулів та класів спроектовано та написано головну логіку програми де користувач може грати із комп’ютером в покер. Гра відбувається в кілька раундів відповідно до правил техаського холдему, в кожному з яких гравцям роздаються карти, наступає раунд ставок та роздаються загальні карти (флоп, терн, рівер).

Кожен раунд починається зі створення об'єкту гри (окремого раунду), після чого відображаються карти для кожного з гравців. Важливим рішенням на яке варто звернути увагу в процесі навчання це дозвіл користувачеві бачити карти комп’ютера. Такий підхід в сумісності із показом результатів кожного раунду дозволить користувачеві швидше засвоїти процес формування та оцінки комбінацій. Обчислення комбінації та ймовірні шанси на перемогу в грі відбуваються практично безперервно, що дозволяє забезпечити повний доступ користувачеві до інформації про зміни стану гри. Гравці можуть здійснювати різні

дії, такі як check або call (підтримати ставку), fold (скинути карти), bet (підняти ставку), в залежності від стратегії гри та оцінки власних карт.

Після кожного зроблено ходу відображаються шанси на перемогу в грі та комбінації гравців в числовому еквіваленті з дробовою частиною. В такому числі ціла частина це сила комбінацій за зростанням від одного до десяти (наприклад 1.384615384615385 – пара). Дробова частина це результат оцінки програми сили комбінації враховуючи доступні карти (наприклад 1. 384615384615385 – може бути парою четвірок яка програє парі десяток з числовим еквівалентом приблизно 2. 6367293721118).

Гра триває до того моменту поки один з гравців не вирішує скинути свої карти або поки не буде останній раунд. У кінці гри оголошується переможець комбінація якого найсильніша або який залишився одним в грі. Нижче наведено код програми відповідальні за перший раунд, де створюється цикл та умова доки Інші раунди працюють аналогічно використовуючи ту саму логіку для появи нових карт та математичних розрахунків для підказок користувачеві.

```
player_combinations = []
computer_combinations = []
player_chances = []
computer_chances = []
while game_going:
game.start_round()
game.display_player(1)
game.display_player(2)
player1_combination = game.get_hand_strength(1)
player2_combination = game.get_hand_strength(2)
player_combinations.append(player1_combination)
computer_combinations.append(player2_combination)

player_chances.append(game.get_winning_chances(1))
computer_chances.append(game.get_winning_chances(2))
print("Player combination:", player1_combination)
print("Player winning chances:", player_chances[-1])
print("Computer combination:", player2_combination)
print("Computer winning chances:", computer_chances[-1])
```

3.3 Тестування програмного модуля

Тестування програмного забезпечення є критично важливим етапом у процесі розробки програм. Декілька ключових причин, чому тестування є важливим це - забезпечення якості де тестування допомагає виявити дефекти та помилки на ранніх етапах розробки, що дозволяє забезпечити високу якість кінцевого продукту. Чим раніше виявлені помилки, тим дешевше і легше їх виправити. Підвищення надійності - таке тестування гарантує, що програма працює належним чином за різних умов, що підвищує її надійність. Відповідність вимогам це перевірка чи збігається кінцева програма з зазначеним вимогам Це включає як функціональні, так і нефункціональні вимоги, такі як продуктивність, зручність використання тощо.

Для проведення функціональне тестування гри в покер, потрібно створити тести, які перевірятимуть основні функції гри на відповідність їх вимогам. Список тестів, які можна зробити для проведення такого тестування подано та протестовано в таблиці 3.1

Таблиця 3.1 Функціональне тестування

Тест	Статус
Роздача карт гравцям	Пройдено
Визначення комбінацій гравців	Пройдено
Розрахунок шансів на виграш	Пройдено
Виконання ходів гравців	Пройдено
Визначення переможця	Пройдено
Побудова статистичних діаграм	Пройдено

Завершальним етапом тестування та завершення роботи над програмою є перевірка повного функціоналу. В даному випадку розглядається повний тест всіх функцій програмного модуля навчання гри в покер та проведення повноцінної гри.

На рисунку 3.3 зображено приклад роботи програмного модуля де можна побачити як саме працює модуль.

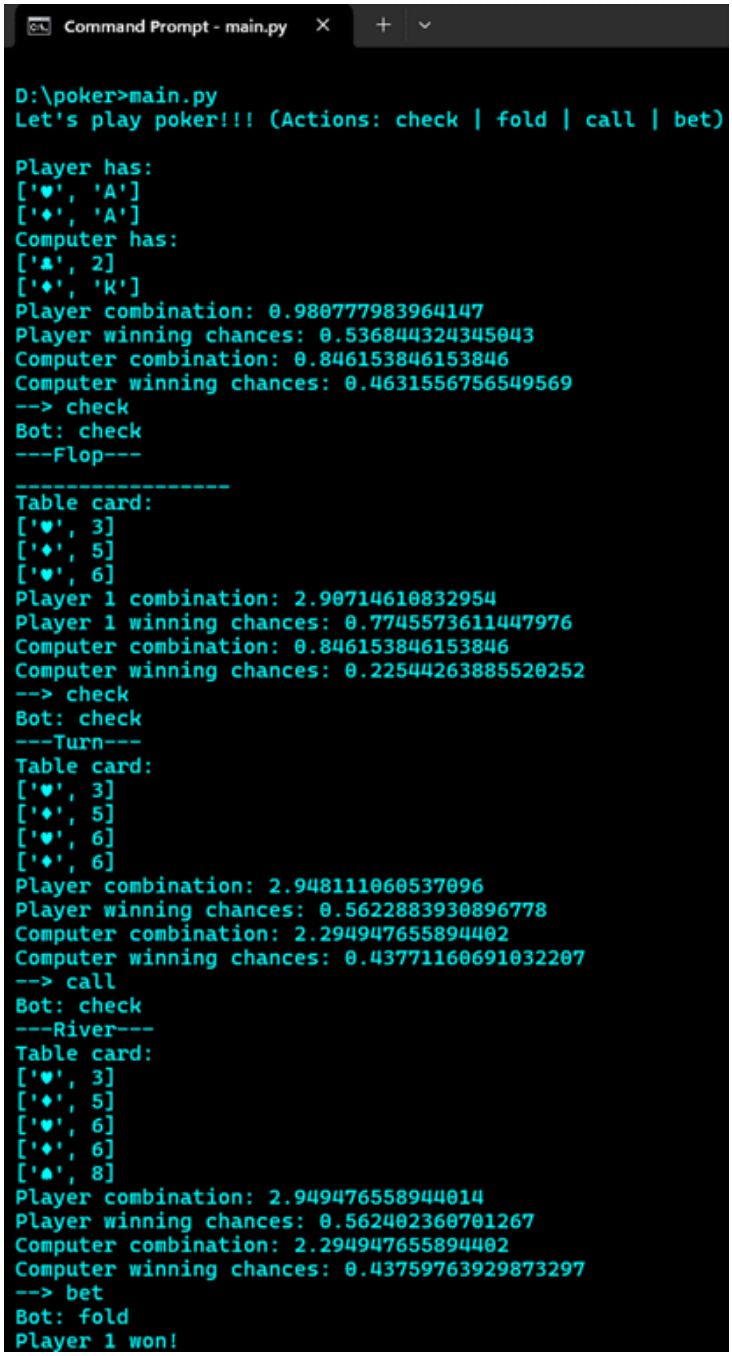


Рисунок 3.3 – Приклад роботи програмного модуля

Після завершення гри програмний модуль всі завантажить збережені дані про кожен раунд гри і представить два графіка показаних на рисунку 3.4. На графіках буде відображатись як зростали сили комбінацій гравців від час гри та який був їх відсоток на перемогу в різних раундах.

Рисунок 3.4 – Графіки змін комбінацій та шансів на перемогу

ВИСНОВКИ

В результаті виконання першого розділу проаналізовано предметну область, розглянуто відомі рішення та сформовано основні задачі та їх вимоги для програмного модуля.

В другому розділі було розроблено загальну структуру проектованого модуля, що включає в себе схеми роботи програми, основний алгоритм гри та таблиці вхідної та вихідної інформації для програмного модуля. Додатково розглянуто основні математичні формули теорії ймовірності на яких базується прийняття рішень та розглянуто основні модулі та бібліотеки, які необхідні для належного функціонування програмного модуля

Використовуючи мову програмування Python та її можливостей в третьому розділі кваліфікаційної роботи на етапі програмної реалізації було розроблено основні класи та методи. Використовуючи математичні розрахунки було досягнуто можливості для розробки власного штучного інтелекту, який проаналізувавши тисячі можливих випадків навчився робити правильні ходи та допомагати користувачеві з розрахунком теорій ймовірності та покерних комбінацій.

Завдяки збереженню симуляційних даних про стан кожного раунди гри було проведено аналіз результатів, що дало змогу побудови статистичних діаграм та графіків. Отримані результати дозволили візуалізувати та краще зрозуміти динаміку гри, що сприяє ефективнішому навчанню користувача.

На етапі тестування впроваджено функціональне тестування та створено тести окремих базових методів програми. Після успішно функціонального тестування відбулось тестування повноцінного програмного модуля.

Розроблений програмний модуль має високу практичну цінність для навчання гри в покер. Він може використовуватися як інструмент для тренування гравців різного рівня підготовки, а також для аналізу стратегій гри. Модуль може бути інтегрований в навчальні платформи та застосунки для самостійного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rubin, J., & Watson, I. Computer poker: A review. *Artificial Intelligence*, 2011, 175(5-6), 958-987.

2. Brilliant.org. "Nash Equilibrium." URL: https://brilliant.org/wiki/nash/equilibrium/

3. Ganzfried, S., & Lanctot, M. EC-16 Tutorial on Computer Poker. Proceedings of the 2016 ACM Conference on Economics and Computation (EC '16), 759-760.

4. Bowling, M., Burch, N., Johanson, M., & Tammelin, O. Heads-up Limit Hold'em Poker is Solved. Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G2E8, Canada.

5. Wikipedia. Poker URL: https://en.wikipedia.org/wiki/Poker

6. Кумецька, О. Техаський Холдем: базові правила. Cardmates. Отримано з https://cardmates.ua/tehasjkyj_holdem_bazovi_pravyla.

7. Toreli A. he Poker Coach: Practical Strategies to Manage Your Bankroll and Outsmart Your Opponents. 2020, 174 p.

8. Red write. URL: https://readwrite.com/gambling/guides/poker-strategy/

9. Hardin, A. Essential Poker Math: Expanded Edition: Fundamental No-Limit Hold'em Mathematics You Need to Know, 2015. 277 p.

10. Hellmuth P. Play Poker Like the Pros. 2003, 416p.

11. BBC Reveals How Math and Psychology Give Poker Players the Upper Hand. URL:https://nerdist.com/article/bbc-reveals-how-math-and-psychology-give-poker-players-the-upper-hand/

12. Advanced Poker Training. URL: https://www.advancedpokertraining.com/.

13. PokerSnowie. URL: https://www.pokersnowie.com/.

14. DriverHUD 2. URL : https://drivehud.com/.

15. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп'яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за першим (бакалаврським) рівнем

вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

16. Pokercode. "GTO Poker: How to Use Game Theory Optimal Strategy." URL: https://www.pokercode.com/blog/gto-poker

17. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. "Introduction to Algorithms." 3rd ed., MIT Press, 2009.

18. Hardin, A. Essential Poker Math: Expanded Edition: Fundamental No-Limit Hold'em Mathematics You Need to Know, 2015. 277 p.

19. Wikipedia. Probability theory URL: https://en.wikipedia.org/wiki/Probability_theory.

20. Britannica Probability theory. URL: https://www.britannica.com/science/probability-theory.

21. Acevedo, M. Modern Poker Theory: Building an unbeatable strategy based on GTO principles. United States: D&B Publishing. 2019, 811 p.

22. Careerfoundry. Programming library guide https://careerfoundry.com/en/blog/web-development/programming-library-guide/

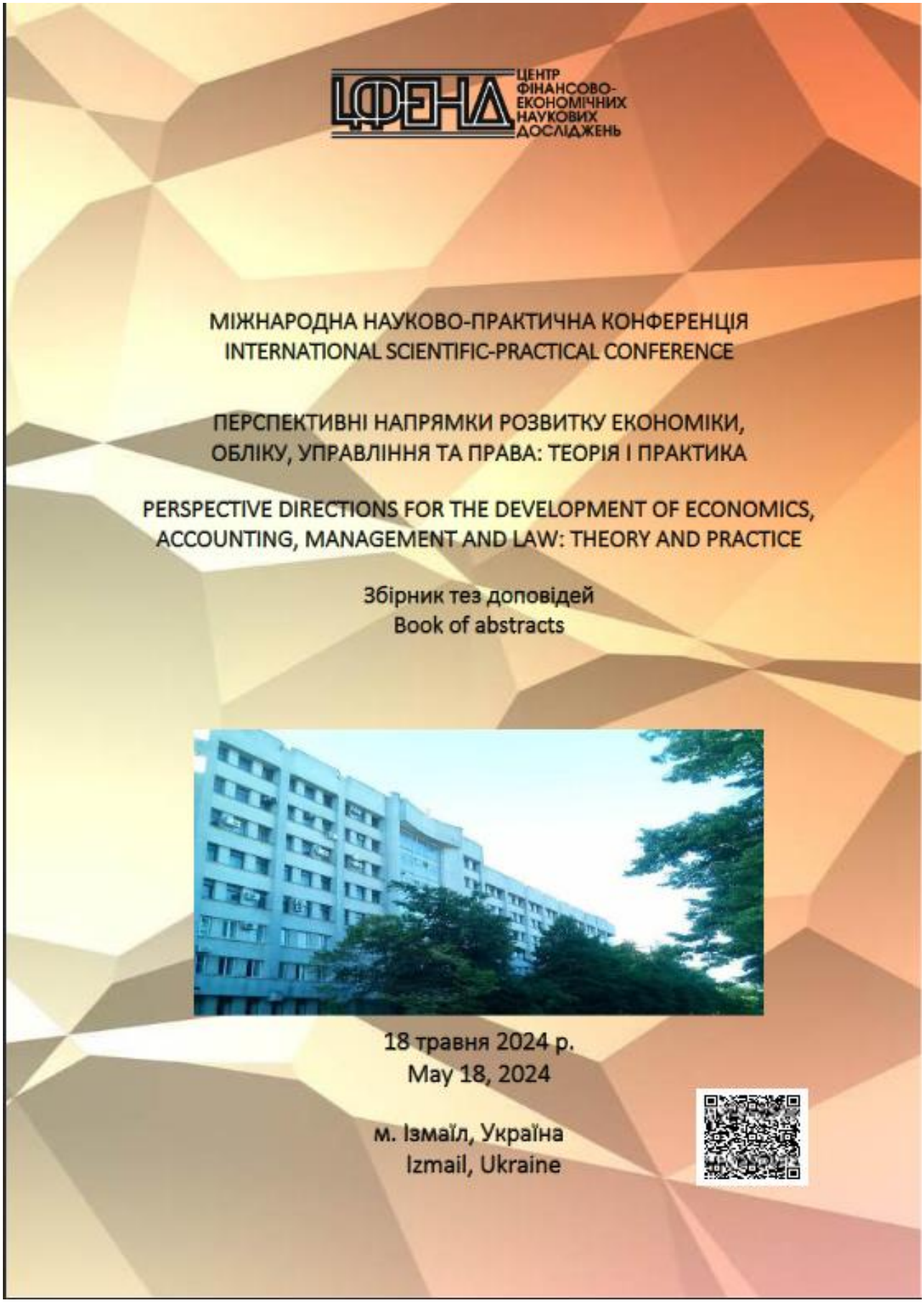
23. Dawson, M. Python Programming for the Absolute Beginner. (3rd ed.). United States: Cengage Learning. 2010, 480 p.

24. Sublime HQ Pty Ltd. Sublime Text Documentation. Отримано з https://www.sublimetext.com/docs/.

25. Перспективні напрямки розвитку економіки, обліку, управління та права: теорія і практика: збірник тез доповідей міжнародної науково-практичної конференції (Ізмаїл, 18 травня 2024 р.). Ізмаїл: ЦФЕНД, 2024. 63 с

ДОДАТОК А

Апробація отриманих результатів



УДК 004.9

Кучар О. М.
студент групи КН-42,
Західноукраїнський національний університет

ПРОГРАМНИЙ МОДУЛЬ НАВЧАННЯ ГРИ В ПОКЕР

Покер, як інтелектуальна гра, поєднує в собі елементи математики, психології та стратегії. Останні десятиліття він набув величезної популярності не лише як азартна гра, а й як спортивна дисципліна, що вимагає високих аналітичних навичок та глибокого розуміння теоретичних основ [1].

У зв'язку з цим, навчання грі в покер стає актуальною темою для дослідження, адже знання та вміння, здобуті під час навчання, можуть бути застосовані не лише в грі, а й у різних сферах життя. Багато основних покерних стратегій зосереджені на фундаментальних поняттях, таких як позиція та розмір ставки, проте новачки часто розглядають їх ізольовано [2].

Навчити гравця можна через систематичне і послідовне вивчення ключових аспектів гри, таких як правила, стратегії, математичні розрахунки, правильна оцінка карт на руках в гравця та на ігровому столі . Постійна практика і навчання на реальних або програмно симульованих ігрових ситуаціях дозволяють гравцеві закріплювати отримані знання та розвивати власний стиль гри.

Важливою частиною процесу навчання є також аналіз гри, можливість отримання певних підказок, навчання правильності розрахунку шансів перемоги тої чи іншої комбінації в конкретній ситуації та чіткої реакції на дії опонентів, що допомагає виявити сильні та слабкі сторони та робити висновки для подальшого вдосконалення. Шлях до володіння покером вимагає від гравця не лише знань, але й наполегливості, самодисципліни та відкритості до постійного розвитку своїх навичок [3].

Тому задача створення програмного модуля для навчання гри в покер є актуальною. Використання програмних засобів забезпечує створення віртуальних суперників, які адаптуються до стилю гри користувача та надають реалістичне відчуття гри. Крім того, модуль дає поради, які допомагають зрозуміти основні концепції стратегії в покері. Використання математичних моделей для оцінки шансів перемоги та аналізу ігрових ситуацій дозволяє значно підвищити ефективність навчання. Інформаційні технології, такі як алгоритми машинного навчання та штучний інтелект, можуть бути використані для створення адаптивних навчальних систем, які надають індивідуальні рекомендації та забезпечують постійний розвиток навичок гравців.

Таким чином, програмний модуль навчання гри в покер дозволяє не лише покращувати свої вміння, але й розвивати здатність швидко адаптуватися до змін у грі та ефективно використовувати отримані знання.

Список літератури

1. BBC Reveals How Math and Psychology Give Poker Players the Upper Hand - [Електронний ресурс] - https://nerdist.com/article/bbc-reveals-how-math-and-psychology-give-poker-players-the-upper-hand/

2. Top 10 Poker Strategy Tips for Beginners - [Електроний ресурс] - https://readwrite.com/gambling/guides/poker-strategy/

3. Покер – [Електронний ресурс] - https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BA%D0%B5%D1%80

ДОДАТОК Б
Основні частини лістингу програмного модуля

Клас Card – відповідальний за створення, перемішування колоди карт та містить методи для роботи з картами під час гри

```
class Card:
    def __init__(self, suit, number):
        self.suit = suit
        self.number = number
    def get_card(self):
        return [self.suit, self.number]
    def get_card_display(self):
        return [self.suit, self.__convert_to_display(self.number)]
    def get_suit(self):
        return self.suit
    def get_suit_number(self):
        if self.suit == "♣":
            return 0
        elif self.suit == "♦":
            return 1
        elif self.suit == "♥":
            return 2
        elif self.suit == "♠":
            return 3
    def get_number(self):
        return self.number
    def get_number_display(self):
        return self.__convert_to_display(self.number)
    def _convert_to_display(self, number):
        if number < 11:
            return number
        elif number == 11:
            return "J"
        elif number == 12:
            return "Q"
        elif number == 13:
            return "K"
        elif number == 14:
            return "A"
        else:
            return "Unknown"
    def _create_deck(self):
        deck = []
        suits = ['♠', '♥', '♣', '♦']
        count = 0
        for suit in suits:
            for i in range(2, 15):
                deck.append(Card(suit, i))
        return deck
    def _shuffle(self, deck):
        for i in range(0, 3):
```

```
for j in range(0, 52):
            rand = int(random.uniform(0, 52))
            temp = deck[j]
            deck[j] = deck[rand]
            deck[rand] = temp
        return deck
```

Цикл для симуляції ігор, збору та аналізу даних, за допомогою яких комп'ютер навчається приймати рішення та визначати сили комбінацій та шнси на перемогу

```
count = 1
while count < 5000:
    print("Simulating Round " + str(count))
    winner = 0
    main.start_round()
    one = main.get_hand_strength(1)
    two = main.get_hand_strength(2)

    if one in statesPocket:
        pocketOneIndex = statesPocket.index(one)
        encounteredPocket[pocketOneIndex] += 1
    else:
        statesPocket.append(one)
        pocketOneIndex = (len(statesPocket)-1)
        probPocket.append(0)
        encounteredPocket.append(1)
    if two in statesPocket:
        pocketTwoIndex = statesPocket.index(two)
        encounteredPocket[pocketTwoIndex] += 1
    else:
        statesPocket.append(two)
        pocketTwoIndex = (len(statesPocket)-1)
        probPocket.append(0)
        encounteredPocket.append(1)
    main.deal_flop()
    one = main.get_hand_strength(1)
    two = main.get_hand_strength(2)

    if one in statesFlop:
        flopOneIndex = statesFlop.index(one)
        encounteredFlop[flopOneIndex] += 1
    else:
        statesFlop.append(one)
        flopOneIndex = (len(statesFlop)-1)
        probFlop.append(0)
        encounteredFlop.append(1)

    if two in statesFlop:
        flopTwoIndex = statesFlop.index(two)
        encounteredFlop[flopTwoIndex] += 1
    else:
        statesFlop.append(two)
```

```
    flopTwoIndex = (len(statesFlop)-1)
    probFlop.append(0)
    encounteredFlop.append(1)
main.deal_turn()
one = main.get_hand_strength(1)
two = main.get_hand_strength(2)

if one in statesTurn:
    turnOneIndex = statesTurn.index(one)
    encounteredTurn[turnOneIndex] += 1
else:
    statesTurn.append(one)
    turnOneIndex = (len(statesTurn)-1)
    probTurn.append(0)
    encounteredTurn.append(1)
if two in statesTurn:
    turnTwoIndex = statesTurn.index(two)
    encounteredTurn[turnTwoIndex] += 1
else:
    statesTurn.append(two)
    turnTwoIndex = (len(statesTurn)-1)
    probTurn.append(0)
    encounteredTurn.append(1)

main.deal_river()
one = main.get_hand_strength(1)
two = main.get_hand_strength(2)
if one in statesRiver:
    riverOneIndex = statesRiver.index(one)
    encounteredRiver[riverOneIndex] += 1
else:
    statesRiver.append(one)
    riverOneIndex = (len(statesRiver)-1)
    probRiver.append(0)
    encounteredRiver.append(1)

if two in statesRiver:
    riverTwoIndex = statesRiver.index(two)
    encounteredRiver[riverTwoIndex] += 1
else:
    statesRiver.append(two)
    riverTwoIndex = (len(statesRiver)-1)
    probRiver.append(0)
    encounteredRiver.append(1)
winner = main.get_winner()
update(pocketOneIndex,flopOneIndex,turnOneIndex,riverOneIndex,1)
update(pocketTwoIndex,flopTwoIndex,turnTwoIndex,riverTwoIndex,2)

if (msvcrt.kbhit()):
    if (ord(msvcrt.getch()) == 27):
        break
count += 1
```

Код для побудови діаграм та графіків на основі зібраних та обчислених ігрових даних

```
plt.figure(figsize=(12, 7))

# Pocket Stage
plt.subplot(2, 2, 1)
plt.scatter(statesPocket, probPocket, c=encounteredPocket, cmap='viridis', s=20, alpha=0.6)
plt.colorbar(label='Number of Encounters')
plt.xlabel('State')
plt.ylabel('Probability')
plt.title('Pocket Stage')

# Flop Stage
plt.subplot(2, 2, 2)
plt.scatter(statesFlop, probFlop, c=encounteredFlop, cmap='viridis', s=20, alpha=0.6)
plt.colorbar(label='Number of Encounters')
plt.xlabel('State')
plt.ylabel('Probability')
plt.title('Flop Stage')

# Turn Stage
plt.subplot(2, 2, 3)
plt.scatter(statesTurn, probTurn, c=encounteredTurn, cmap='viridis', s=20, alpha=0.6)
plt.colorbar(label='Number of Encounters')
plt.xlabel('State')
plt.ylabel('Probability')
plt.title('Turn Stage')

# River Stage
plt.subplot(2, 2, 4)
plt.scatter(statesRiver, probRiver, c=encounteredRiver, cmap='viridis', s=20, alpha=0.6)
plt.colorbar(label='Number of Encounters')
plt.xlabel('State')
plt.ylabel('Probability')
plt.title('River StaIge')

plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(player_combinations, label='Player', marker='o')
plt.plot(computer_combinations, label='Computer', marker='o')
plt.title('Player Combinations')
plt.xlabel('Round')
plt.ylabel('Combination')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(player_chances, label='Player', marker='o')
plt.plot(computer_chances, label='Computer', marker='o')
plt.title('Player Winning Chances')
```

```
plt.xlabel('Round')
plt.ylabel('Chances')
plt.legend()

plt.tight_layout()
plt.show()
```