КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Кваліфікаційна робота

на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

ПРОГНОЗУВАННЯ КОРИСТУВАЦЬКОЇ ПОВЕДІНКИ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

Виконав: студент 4-го курсу
Юрій МАЦКЕВИЧ

(підпис)

Науковий керівник:
асистент, кандидат технічних наук
Сергій КОНДРАТЮК

(підпис)

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

(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри теоретичної кібернетики
«____» _____ 2023 р.,
протокол № ___
Завідувач кафедри
доктор фіз.-мат. наук, професор
Юрій КРАК

(підпис)

КИЇВ-2023

РЕФЕРАТ

Обсяг роботи 41 сторінка, 9 ілюстрацій, 6 джерел посилань, 2 додатки
БЛЕКДЖЕК, МОДЕЛЬ ПРОГНОЗУВАННЯ, ГРАВЦІ, ОПТИМАЛЬНІ РІШЕННЯ

Метою дослідження було розробити модель, яка надає гравцям оптимальні рекомендації під час гри, допомагаючи їм приймати найвигідніші рішення.

У роботі було використано набір даних гри блекджек, який був завантажений та підготовлений для подальшого аналізу. Необхідні стовпці були вибрані, мітки класів були перетворені у числові значення, а також було створено набір ознак, необхідних для тренування моделі.

Для розробки моделі прогнозування було використано нейронну мережу з використанням бібліотеки TensorFlow. Була визначена архітектура моделі, включаючи шари Dense з активацією ReLU та остаточний шар з активацією sigmoid. Модель була скомпільована з оптимізатором Adam, функцією втрат binary_crossentropy та метрикою точності.

Для тренування моделі було використано набір даних, і проведено 10 епох тренування. Після тренування модель мала здатність прогнозувати оптимальні рішення на основі початкових карт гравця та карти дилера.

Дані користувача вводилися для отримання рекомендацій моделі. За допомогою навченої моделі, введені дані були використані для прогнозування оптимального рішення. Модель видала рекомендацію гравцеві про те, чи потрібно забрати додаткову карту чи залишитися зі своїми поточними картами.

У підсумку, розроблена модель прогнозування користувацької поведінки в грі блекджек може бути корисною для гравців, які бажають приймати оптимальні рішення в грі. Ця модель може допомогти гравцям зробити обґрунтовані вибори та підвищити їх шанси на успіх у грі блекджек.

ЗМІСТ

ВСТУП

Розробка та прогнозування користувацької поведінки за допомогою методів машинного навчання. Основною сферою застосування розробленого рішення є ігрова індустрія, зокрема гра блекджек, яка реалізована на платформі Unreal Engine 5.

Актуальність цієї роботи обумовлена постійним розвитком ігрової індустрії та необхідністю розробки ефективних методів прогнозування користувацької поведінки з метою покращення ігрового процесу. Традиційні підходи до управління поведінкою гравців базуються на правилах та емпіричних спостереженнях, що може обмежувати їх ефективність та гнучкість.

Метою даної роботи є розробка та застосування методів машинного навчання для прогнозування користувацької поведінки в грі блекджек. Основним завданням є побудова моделі, яка здатна прогнозувати рішення гравця в залежності від початкових карт та поточного стану гри.

Об'єктом дослідження є гра блекджек, реалізована на платформі Unreal Engine 5. Методи дослідження включають використання методів машинного навчання, зокрема алгоритмів класифікації та прогнозування, а також аналіз та обробку даних гри.

Можливі сфери застосування розробленого рішення включають використання його в онлайн-казино для автоматичного прогнозування рішень гравців, а також як інструмент для навчання та аналізу гри блекджек. Крім того, дане дослідження може знайти застосування в інших галузях, де важливо прогнозувати та управляти користувацькою поведінкою на основі аналізу даних.

Апробація роботи передбачає тестування та оцінку розробленої моделі на різних сценаріях гри блекджек з використанням різних початкових карт та

поточного стану гри. Також можлива апробація шляхом порівняння результатів прогнозів моделі з експертними стратегіями гри.

В цій дипломній роботі буде проведено аналіз та розробка моделі прогнозування користувацької поведінки в грі блекджек з використанням методів машинного навчання та TensorFlow. Результати цього дослідження можуть мати значимий вплив на розвиток ігрової індустрії та дослідження в області машинного навчання, а також сприяти покращенню геймплею та задоволенню користувачів.

Мета роботи:

Основною метою даної дипломної роботи є розробка та застосування методів машинного навчання з використанням TensorFlow для прогнозування користувацької поведінки в грі блекджек. Розроблена модель повинна бути здатна аналізувати початкові карти гравця та карту дилера, а також поточний стан гри, і робити прогноз щодо оптимального рішення гравця: добрати додаткову карту або залишитися з поточним набором карт.

Завдання роботи:

Для досягнення поставленої мети, робота передбачає виконання наступних завдань:

1. Збір та підготовка даних: зібрати достатню кількість даних з реальних ігор блекджек, включаючи початкові карти гравця та карту дилера, а також результат гри (наприклад, перемога або поразка гравця).
2. Аналіз та підготовка даних: провести аналіз зібраних даних, включаючи перевірку на наявність пропущених значень та аномалій. Потім необхідно підготувати дані для подальшого використання у моделі, зокрема, виконати масштабування, векторизацію та розділити дані на тренувальний та тестовий набори.
3. Розробка моделі машинного навчання: використовуючи бібліотеку TensorFlow, розробити модель, що базується на нейронних

мережах, для прогнозування оптимального рішення гравця в грі блекджек. Встановити архітектуру моделі, вибрати оптимальні параметри та здійснити процес навчання моделі на тренувальних даних.

4. Оцінка та валідація моделі: оцінити ефективність розробленої моделі за допомогою метрик якості, таких як точність, чутливість, специфічність та F-мера. Виконати валідацію моделі на тестовому наборі даних, щоб переконатися в її здатності до точних прогнозів.
5. Застосування моделі в грі блекджек: інтегрувати розроблену модель у гру блекджек, реалізовану на Unreal Engine 5. Провести тестування та оцінку моделі в реальних умовах гри, переконатися в її здатності приймати оптимальні рішення відповідно до передбаченої користувацької поведінки.
6. Апробація роботи: оцінити ефективність розробленої моделі та її вплив на геймплей та користувацьке задоволення шляхом проведення апробації на реальних користувачах або експертах в грі блекджек.

Результати даної роботи можуть мати значимий вплив на розвиток ігрової індустрії та дослідження в області машинного навчання. Повністю автоматизована модель прогнозування користувацької поведінки в грі блекджек може покращити геймплей, забезпечити більш адаптивний та цікавий досвід гравця та сприяти досягненню більш високих результатів в грі.

1 ПРЕДМЕТНА ОБЛАСТЬ

1.1 Моделі нейронних мереж у прогнозуванні користувацької поведінки

Використання нейронних мереж для аналізу та прогнозування користувацьких дій є актуальним напрямком досліджень, який став можливим завдяки розширенню машинного навчання та розвитку обчислювальної потужності. Нейронні мережі є потужним інструментом для обробки та аналізу складних даних, зокрема послідовностей дій користувачів.

У контексті аналізу користувацької поведінки, нейронні мережі можуть бути застосовані для розпізнавання патернів, класифікації дій, прогнозування тенденцій та виявлення аномальних поведінкових зразків. Вони можуть вивчати залежності між послідовностями дій та відображати складні зв'язки, які не завжди можна виявити за допомогою традиційних статистичних методів.

Наприклад, нейронні мережі можуть бути використані для аналізу дій користувачів на веб-сайті або в мобільному додатку. За допомогою навчання на попередніх даних, модель може виявити характерні шаблони дій, такі як перегляд певних сторінок, виконання певних дій або взаємодія з елементами інтерфейсу. На основі цих шаблонів можна класифікувати дії користувачів, прогнозувати їхні подальші кроки або виявляти незвичайну поведінку, яка може вказувати на шахрайство або інші проблеми.

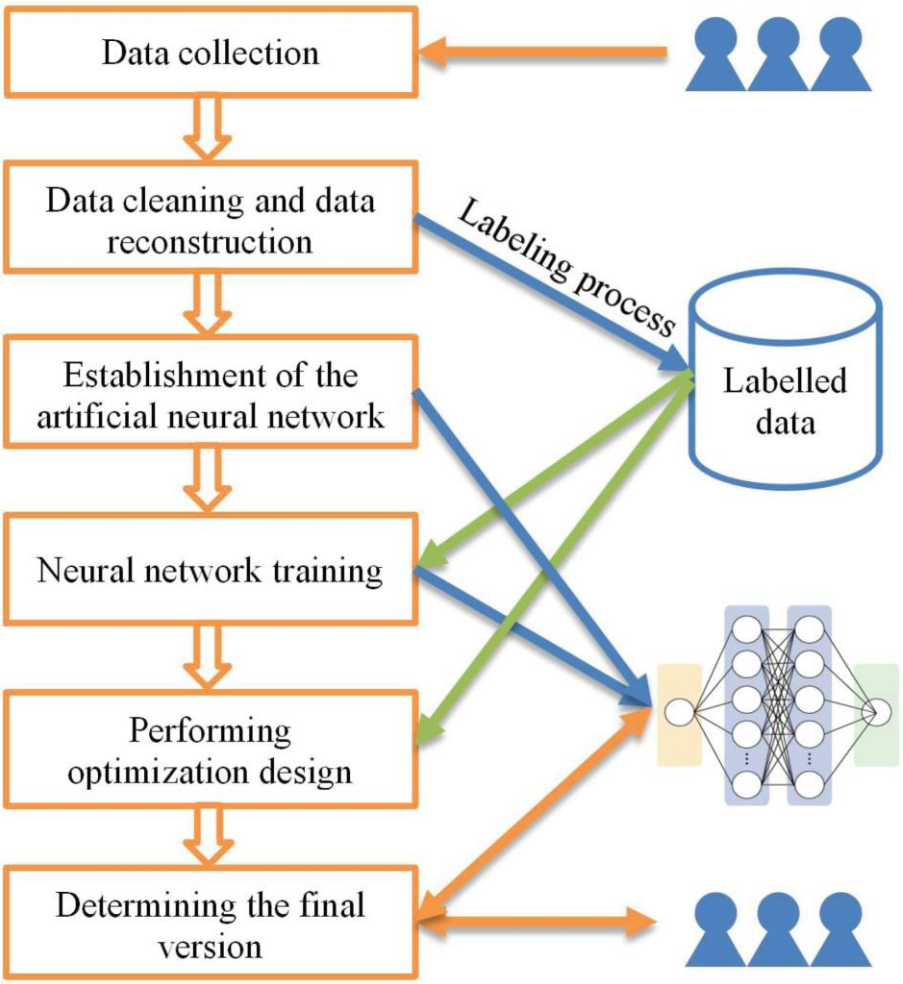


Рисунок 1.1 - Демонстрація процесу обробки інформації нейронною мережею

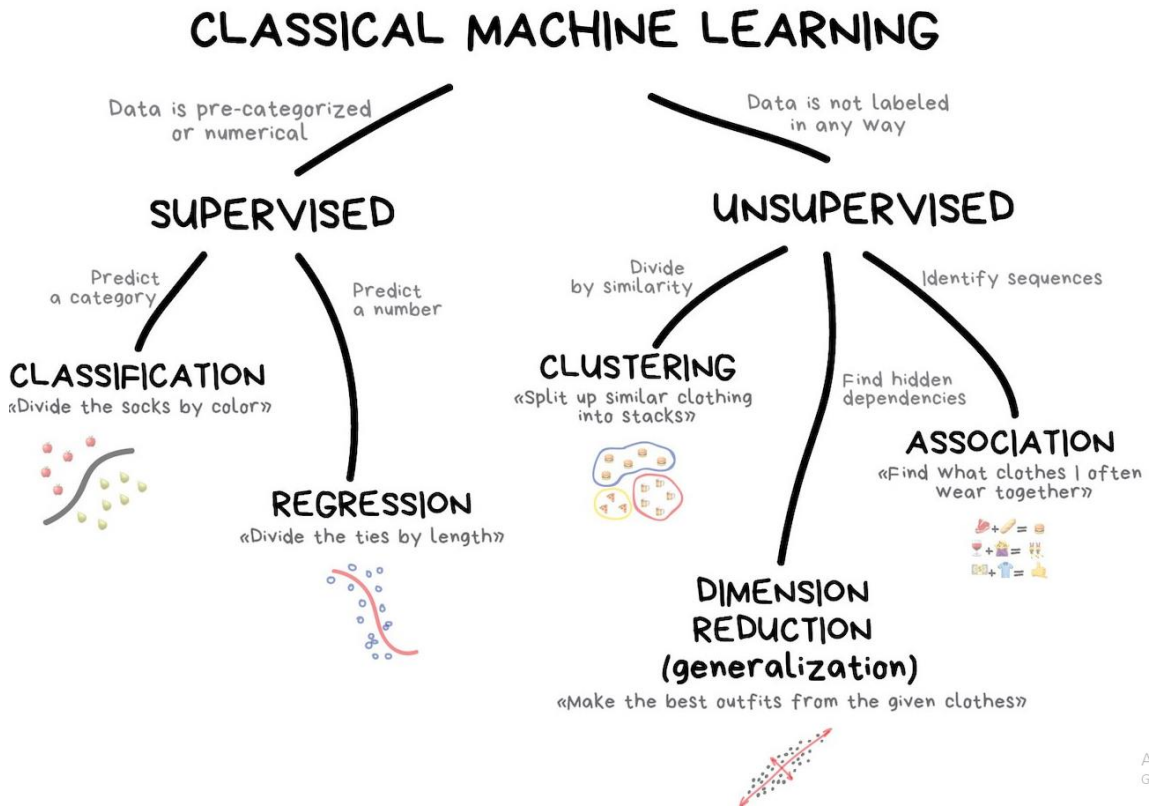
Для досягнення цих цілей розробляються різні типи нейронних мереж. Наприклад, рекурентні нейронні мережі (RNN) можуть ефективно працювати з послідовністю дій, оскільки вони можуть зберігати інформацію про попередні стани та використовувати її для прийняття рішень. Це особливо корисно при аналізі послідовностей дій користувачів, де попередні дії можуть мати вплив на подальші.

Також, згадані раніше згорткові нейронні мережі (CNN) використовуються для аналізу зображень або текстових описів дій користувачів. Вони можуть виявляти характеристики та зразки в даних, що допомагає в розпізнаванні об'єктів або взаємодії з текстовими елементами. CNN використовуються для аналізу зображень, де вони можуть визначати обличчя,

об'єкти або емоції, що може бути важливим для розуміння користувацької поведінки.

Отже, використання нейронних мереж для аналізу та прогнозування користувацьких дій відкриває безліч можливостей для вдосконалення продуктів та послуг. Це дозволяє отримати більш глибоке розуміння користувацьких потреб, покращити персоналізацію та забезпечити більш ефективну взаємодію з користувачами.

Застосування машинного навчання для моделювання поведінкових шаблонів є важливою галуззю досліджень і розробок. Ця область зосереджена на використанні різних алгоритмів та моделей машинного навчання для аналізу і передбачення поведінки людей або систем.



1.2 - Класичний приклад використання машинного навчання

Одним з ключових аспектів застосування машинного навчання для моделювання поведінкових шаблонів є збір та аналіз великого обсягу даних про поведінку. Ці дані можуть бути зібрані з різних джерел, таких як соціальні мережі, мобільні додатки, електронні транзакції і т.д. Збирання цих даних дозволяє отримати розуміння структури та характеристик поведінки.

Далі, застосовуються різні алгоритми та моделі машинного навчання для аналізу цих даних та побудови моделей поведінкових шаблонів. Наприклад, можуть використовуватися алгоритми класифікації, регресії, кластеризації, асоціативного аналізу та інші. Ці моделі навчаються на зібраних даних, виявляють залежності та закономірності в поведінці та дозволяють зробити передбачення щодо майбутньої поведінки.

Однією з сфер застосування є персоналізований маркетинг. За допомогою моделей поведінкових шаблонів можна аналізувати інформацію про попередню покупкову історію та поведінку користувачів, щоб розробити персоналізовані пропозиції та рекламні компанії.

Іншою сферою застосування є боротьба зі шахрайством та кібербезпека. Шаблони поведінки можуть бути використані для виявлення незвичайних або підозрілих активностей, що допомагає виявити шахраїв та запобігти злочинам.

Також застосування машинного навчання для моделювання поведінкових шаблонів може бути корисним в галузі медицини, де використовуються дані про пацієнтів для прогнозування ризиків, діагностики та лікування різних захворювань.

Узагальнюючи, застосування машинного навчання для моделювання поведінкових шаблонів відкриває широкі можливості в різних сферах, де аналіз та передбачення поведінки є ключовими факторами для досягнення успіху. Використання цих моделей дозволяє виявити залежності, зрозуміти тенденції та зробити передбачення, що сприяє прийняттю кращих рішень та поліпшенню результатів.

Використання TensorFlow для побудови та навчання моделей нейронних мереж є однією з найпопулярніших і потужних стратегій у галузі машинного навчання. TensorFlow - це відкрите програмне забезпечення бібліотека машинного навчання, розроблена компанією Google. Вона надає широкий спектр інструментів та функціональностей, які дозволяють легко побудувати, навчити та використовувати нейронні мережі для різних завдань.

Одним з головних переваг TensorFlow є його гнучкість та скальдованість. Він дозволяє визначити складні архітектури нейронних мереж за допомогою високорівневих API, таких як Keras, або з нижчим рівнем контролю за допомогою TensorFlow Core. TensorFlow підтримує різні типи нейронних мереж, включаючи звичайні нейронні мережі, згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та багатошарові перцептрони (MLP).

Крім того, TensorFlow має потужні функції для навчання моделей нейронних мереж. Він надає широкий спектр алгоритмів оптимізації, таких як стохастичний градієнтний спуск (SGD), адам (Adam), RMSprop та інші, що допомагають знаходити оптимальні параметри моделі. TensorFlow також забезпечує можливість використання апаратного прискорення за допомогою графічних процесорів (GPU) та тензорних процесорів (TPU), що дозволяє прискорити процес навчання моделей.

Одна з ключових особливостей TensorFlow - це його здатність до автоматичного диференціювання. Це означає, що ви можете визначити функцію втрати (loss function) та автоматично обчислювати градієнти параметрів моделі. Це важливо для здійснення оптимізації параметрів моделі за допомогою алгоритмів оптимізації з використанням градієнтного спуску.

Крім того, TensorFlow надає зручні інструменти для візуалізації та моніторингу процесу навчання моделей. За допомогою TensorBoard, ви можете відстежувати показники процесу навчання, як-от точність, втрати, градієнти та інші, і відображати їх у зручному для сприйняття вигляді.

Використання TensorFlow для побудови та навчання моделей нейронних мереж є потужним інструментом у галузі машинного навчання. Він дозволяє розробникам та дослідникам ефективно працювати з нейронними мережами, створювати складні архітектури, навчати моделі на великих обсягах даних та досягати високої точності у різних завданнях прогнозування та класифікації.

1.2 Задача класифікації в контексті прогнозування користувацької поведінки

Призначення задачі класифікації в аналізі поведінки користувачів полягає у визначенні категорії або класу, до якого належить конкретна поведінка користувача. Ця задача є однією з найпоширеніших у галузі аналізу даних та машинного навчання і використовується для різних цілей, включаючи рекомендації, персоналізацію, антифрод, класифікацію ризиків та багато іншого.

Основні особливості задачі класифікації в аналізі поведінки користувачів:

1. Набір даних: Для розв'язання задачі класифікації потрібен добре підготовлений набір даних, який містить інформацію про поведінку користувачів. Цей набір даних може включати різноманітні фактори, такі як активність на сайті, купівельні звички, інтереси, демографічні дані тощо.
2. Вибір моделі: Для класифікації поведінки користувачів можуть використовуватись різні моделі машинного навчання, включаючи логістичну регресію, дерева рішень, наївний Баєс, ансамблеві методи (наприклад, випадковий ліс), нейронні мережі та інші. Вибір моделі залежить від конкретного завдання та характеристик набору даних.
3. Особливості даних: Дані, що стосуються поведінки користувачів, можуть мати свої особливості. Наприклад, вони можуть бути незбалансованими, коли одних класів (поведінка) має значно менше, ніж інших. Це може вплинути на процес навчання моделі та вимагати застосування стратегій балансування класів.
4. Вибір ознак: Важливим етапом у класифікації поведінки користувачів є вибір релевантних ознак (факторів), які будуть використовуватись для навчання моделі. Це можуть бути, наприклад, часові параметри, покупки, відвідування певних сторінок, інтереси, сіткові дані тощо. Важливо правильно вибрати ознаки, які найбільше впливають на класифікацію поведінки.

5. Оцінка результатів: Оцінка результатів класифікації є важливою частиною процесу. Використовуються різні метри, такі як точність (accuracy), чутливість (recall), специфічність (specificity), F-мера (F-measure) тощо, для визначення ефективності моделі і порівняння різних підходів.

Застосування машинного навчання для класифікації поведінки користувачів дозволяє отримати цінні інсайти щодо їхньої активності, звичок та інтересів. Це може бути використано для покращення персоналізації, рекомендаційних систем, антифрод заходів та інших аспектів, що сприяють покращенню користувацького досвіду та бізнес-результатів.

Вибір та побудова відповідних ознакових описів для класифікації поведінкових дій є ключовим етапом у розв'язанні задачі аналізу поведінки користувачів. Від правильного вибору та побудови ознак залежить ефективність та точність моделі класифікації. Давайте розглянемо цей процес в деталях:

1. Розуміння завдання: Перш ніж перейти до вибору ознак, важливо чітко зрозуміти завдання класифікації та контекст, в якому вона застосовується. Наприклад, якщо метою є виявлення шахрайської активності, то можуть бути важливі фактори, пов'язані з транзакціями, геолокацією, іншими взаємодіями тощо. Розуміння завдання допоможе визначити, які аспекти поведінки користувачів є релевантними для класифікації.
2. Збір даних: Для побудови ознакового опису потрібно мати наявність достатньої кількості даних, які відображають поведінку користувачів. Ці дані можуть бути зібрані з різних джерел, таких як журнали активності, бази даних, веб-аналітика тощо. Важливо забезпечити якість та достовірність даних, щоб уникнути спотворень при побудові ознак.

3. Вибір ознак: Вибір відповідних ознак залежить від характеру завдання та доступних даних. Ознаки можуть бути кількісними (наприклад, кількість відвідувань сторінок, тривалість сесій) або якісними (наприклад, тип дії, категорія взаємодії). Важливо врахувати, що обрані ознаки повинні мати семантичну вагу і впливати на класифікацію поведінки.
4. Інженерія ознак: Інженерія ознак включає створення нових ознак шляхом поєднання, трансформації або вибору наявних ознак. Наприклад, можна об'єднати кількість покупок з часовими параметрами для створення ознаки "середня кількість покупок на день". Також можна використовувати методи зведення розмірності, наприклад, використовувати аналіз головних компонент для виділення найінформативніших ознак.
5. Нормалізація та масштабування: Перед використанням ознак для навчання моделі важливо здійснити нормалізацію та масштабування. Це допомагає уникнути проблем з різницею в масштабах ознак і забезпечити стабільність навчання моделі.
6. Відбір ознак: Якщо маємо велику кількість ознак, може бути доцільно здійснити відбір найінформативніших ознак. Це допоможе зменшити розмірність простору ознак і покращити швидкість та ефективність класифікації.
7. Перевірка ознак: Важливо перевірити, чи обрані ознаки відображають дійсні закономірності поведінки користувачів. Це можна зробити шляхом візуалізації даних та аналізу їхнього впливу на класифікацію. При необхідності можна внести коригування в процесі вибору та побудови ознак.

Вибір та побудова відповідних ознакових описів для класифікації поведінкових дій є складним процесом, який вимагає глибокого розуміння завдання та аналізу доступних даних. Вірно обрані ознаки можуть значно покращити

точність та ефективність моделі класифікації, що дозволяє зробити більш точні та інформативні прогнози щодо поведінки користувачів.

Порівняння та оцінка ефективності різних алгоритмів класифікації є важливим кроком у процесі розробки моделі для аналізу та прогнозування користувацьких дій. Цей процес дозволяє визначити, який алгоритм краще впорається з поставленою задачею та надає найкращу точність та надійність. Оцінка ефективності алгоритмів класифікації включає такі етапи:

1. Розділення даних на тренувальний і тестовий набори: Перед оцінкою ефективності необхідно розділити наявні дані на тренувальний набір (для навчання моделі) і тестовий набір (для оцінки результатів). Таке розділення дозволяє перевірити, наскільки добре модель справляється з новими, раніше не баченими даними.
2. Вибір метрик для оцінки ефективності: Існує ряд метрик, які використовуються для оцінки ефективності моделей класифікації, таких як точність (accuracy), показник відновлення (recall), показник точності (precision), F-міра (F-measure) тощо. Вибір відповідних метрик залежить від особливостей задачі та вимог замовника.
3. Тренування та оцінка моделей: Після вибору алгоритмів класифікації необхідно навчити моделі на тренувальному наборі даних. Потім виконується оцінка результатів на тестовому наборі даних за допомогою обраної метрики. Цей процес повторюється для кожного алгоритму класифікації.
4. Порівняння результатів: Результати оцінки ефективності для кожного алгоритму класифікації порівнюються. Оцінка здійснюється з урахуванням метрик та вимог до точності та надійності моделі. Найефективніші алгоритми можуть бути вибрані для подальшого використання в аналізі та прогнозуванні користувацьких дій.

Важливо зазначити, що оцінка ефективності алгоритмів класифікації є ітеративним процесом. Залежно від результатів оцінки, можуть вноситися

зміни в параметри моделей, вибиратися інші алгоритми або застосовуватися ансамблеві методи для поліпшення результатів.

Таким чином, порівняння та оцінка ефективності різних алгоритмів класифікації допомагають визначити найкращий підхід для аналізу та прогнозування поведінкових дій користувачів, забезпечуючи більш точні та надійні результати.

1.3 Інструменти реалізації

1.3.1 Мова Python

Python є високорівневою, інтерпретованою мовою програмування, яка була створена у 1991 році Гвідо ван Россумом. Вона має простий і зрозумілий синтаксис, що робить її дуже популярною серед початківців та професіоналів у галузі програмування.

Основні особливості Python:

Синтаксис: Python використовує простий та зрозумілий синтаксис, що дозволяє легко читати та розуміти код. Він базується на використанні відступів (пробіли або табуляції) для визначення блоків коду, що сприяє зрозумілості і структурованості програм.

Інтерпретованість: Python - це інтерпретована мова програмування, що означає, що код виконується рядок за рядком без необхідності компіляції. Це робить розробку та відлагодження програм швидшими та зручнішими.

Багата стандартна бібліотека: Python має велику стандартну бібліотеку, що надає широкий набір корисних функцій та модулів. Вона включає модулі для роботи з рядками, файлами, мережами, базами даних, математикою, науковими обчисленнями та багато іншого.

Машинне навчання та наука даних: Python став популярним серед спеціалістів у галузі машинного навчання та науки даних завдяки таким потужним бібліотекам, як TensorFlow, Keras, PyTorch, scikit-learn та pandas.

Вони надають широкі можливості для розробки та розгортання моделей машинного навчання та аналізу даних.

Розширюваність: Python підтримує інтеграцію з іншими мовами програмування, такими як C, C++, Java, що дозволяє використовувати швидкі та оптимізовані функції з інших мов.

Спільнота розробників: Python має велику та активну спільноту розробників. Існує багато ресурсів, форумів, бібліотек, курсів та пакетів допомоги, що полегшують вивчення та використання мови.

Python знаходить застосування в різних сферах, таких як веб-розробка, наукові дослідження, автоматизація, робототехніка, аналіз даних, машинне навчання та багато інших. Його простота, легкість вивчення та потужність роблять його однією з найпопулярніших мов програмування у світі.

1.3.2 Модуль TensorFlow

TensorFlow є однією з найпопулярніших відкритих бібліотек для машинного навчання та глибокого навчання. Він розроблений компанією Google та надає широкий набір інструментів для побудови та тренування моделей штучних нейронних мереж.

Основні особливості модуля TensorFlow:

Графові обчислення: TensorFlow базується на графовій моделі обчислень, де операції представлені як вузли графа, а дані передаються як ребра графа. Це дозволяє ефективно виконувати обчислення та оптимізувати використання ресурсів.

Множинна платформена підтримка: TensorFlow підтримує різні платформи, включаючи настільні комп'ютери, сервери та мобільні пристрої. Це дозволяє розгортати моделі на різних пристроях та забезпечує гнучкість у виборі апаратного забезпечення.

Глибоке навчання: TensorFlow надає потужні інструменти для побудови глибоких нейронних мереж. Він має вбудовану підтримку для різних типів

штучних нейронних мереж, включаючи згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та довготривалу пам'ять (LSTM).

Автоматичне диференціювання: TensorFlow надає автоматичне диференціювання, що дозволяє обчислювати градієнти функцій автоматично. Це особливо корисно при навчанні моделей машинного навчання з використанням методу зворотного поширення помилки.

Розширені можливості: TensorFlow має багато інструментів та функцій, що полегшують розробку та відлагодження моделей. Він має вбудовану підтримку для візуалізації графів, збереження та завантаження моделей, віддаленого обчислення та розподіленого навчання.

Велика спільнота та ресурси: TensorFlow має активну спільноту розробників, яка надає багато ресурсів, таких як документація, приклади коду, блоги та форуми. Це дозволяє розробникам ефективно використовувати TensorFlow та отримувати підтримку в разі потреби.

Використання TensorFlow дозволяє швидко та ефективно реалізувати складні моделі машинного навчання та глибокого навчання. Він дозволяє дослідникам та розробникам експериментувати з новими алгоритмами та архітектурами, а також застосовувати готові моделі для рішення реальних завдань. TensorFlow є потужним інструментом у сфері штучного інтелекту та аналітики даних.

1.3.3 Unreal Engine

Unreal Engine є потужною та популярною ігровою розробницькою платформою, створеною компанією Epic Games. Це інтегроване середовище розробки програмного забезпечення (IDE), яке надає розробникам широкий набір інструментів та можливостей для створення вражаючих ігор, віртуальної реальності (VR), анімації та візуалізації.

Особливості Unreal Engine дозволяють розробникам створювати гранично реалістичні ігрові світи та інтерактивні віртуальні досвіди. Незалежно від того, чи ви новачок у галузі розробки ігор або досвідчений професіонал, Unreal Engine надає інтуїтивно зрозумілий інтерфейс та широкий набір інструментів, що полегшують процес створення ігор та забезпечують високу якість і результативність.

Основні переваги Unreal Engine включають:

Потужний графічний двигун: Unreal Engine має вражаючий графічний двигун, що дозволяє створювати реалістичні візуальні ефекти, включаючи реалістичне освітлення, тіні, фізичну симуляцію та динамічну обробку матеріалів. Це дозволяє створювати гріплей, який вражає своєю візуальною якістю та реалізмом.

Багатоплатформова підтримка: Unreal Engine підтримує різні платформи, включаючи ПК, консолі, мобільні пристрої та віртуальну реальність. Це дає розробникам можливість створювати ігри, які працюють на різних пристроях та операційних системах, забезпечуючи широку аудиторію та максимальну доступність.

Мови програмування та скриптування: Unreal Engine використовує мови програмування C++ та Blueprint. C++ є мовою програмування загального призначення, що надає розробникам високий рівень контролю та продуктивності. Blueprint - це візуальна скриптова мова, яка дозволяє створювати логіку гри за допомогою вузлів та зв'язків, спрощуючи процес розробки та прототипування.

Розширюваність та спільнота: Unreal Engine надає розробникам можливість розширювати функціональність двигуна за допомогою плагінів та модулів. Це дозволяє використовувати сторонні інструменти та розширювати можливості Unreal Engine. Крім того, існує активна спільнота розробників, яка надає підтримку, документацію, приклади коду та ресурси, що сприяють ефективному використанню платформи.

Фізична симуляція та інтерактивність: Unreal Engine має вбудовану систему фізичної симуляції, яка дозволяє створювати реалістичну поведінку об'єктів та фізичні взаємодії. Це дозволяє створювати ігри з високою ступенем взаємодії та реалістичними фізичними ефектами.

Unreal Engine є потужним та високоефективним інструментом для розробки ігор та інтерактивних віртуальних досвідів. Він надає розробникам широкі можливості для втілення їхніх творчих ідей та створення неперевершених ігрових вражень.

1.3.4 Інші інструменти реалізації

UMG (Unreal Motion Graphics) є модулем Unreal Engine, який дозволяє розробникам створювати інтерфейси користувача та графічні елементи для віртуальних досвідів. Він надає широкий набір інструментів для створення розширених та інтерактивних інтерфейсів.

Завдяки UMG, розробники можуть швидко створювати привабливий та функціональний інтерфейс для своїх віртуальних досвідів у Unreal Engine.

Git - це розподілена система керування версіями, що використовується для керування змінами у програмному коді та спільної роботи над проектами. Завдяки Git розробники можуть відстежувати зміни у коді, створювати гілки для паралельної розробки, об'єднувати зміни з різних гілок, відміняти непотрібні зміни та зберігати повну історію розробки.

GitLab - це веб-платформа для керування репозиторіями Git та забезпечення спільної роботи над проектами. Вона надає можливості зберігання, керування та відстеження версій коду, а також функції управління завданнями, змінами та перегляду коду.

PyCharm - це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. Воно надає широкий набір функцій та інструментів, що полегшують процес розробки, таких як автодоповнення коду, відстеження помилок, рефакторинг, вбудована система контролю версій та інші.

2 ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Аналіз моделі

Вибрали модель нейронної мережі з використанням TensorFlow із такою архітектурою:

- Початковий шар Dense з 64 нейронами і функцією активації ReLU.
- Потім ще один Dense шар з 64 нейронами і функцією активації ReLU.
- Завершальний Dense шар з 1 нейроном і функцією активації Sigmoid.

Ми обрали цю модель, оскільки вона має добру здатність до навчання на різних типах даних і є добре підходящою для задач класифікації з двома можливими вихідними класами (прогнозування дії "добрати карту" або "залишитися").

Така архітектура дозволяє моделі вивчати складні залежності між вхідними картами гравця та картою дилера, а також здійснювати прогнози щодо оптимального рішення гравця.

Ця модель була обрана на основі розгляду різних альтернативних моделей та їхньої придатності для даної задачі.

2.2 Тренування моделі

Для тренування нейронної мережі використовувалась набір даних, який містив інформацію про картки гравця та дилера, а також відповідні результати гри (виграш, програш, нічия). Процес тренування складався з кількох кроків:

- Завантаження даних: Набір даних був завантажений з використанням бібліотеки Pandas. Він був представлений у вигляді

таблиці, де кожен рядок представляв одну гру зі своїми вхідними параметрами та результатом гри.

- Підготовка даних: З набору даних були вибрані необхідні вхідні параметри (картки гравця та дилера) та цільова змінна (результат гри). Вхідні параметри були перетворені в числові значення для подальшої обробки.
- Розбиття на тренувальний та тестовий набори: Набір даних був розбитий на тренувальний набір із відсотком прикладів для тренування моделі та тестовий набір для оцінки точності моделі.
- Побудова моделі: Була побудована нейронна мережа з використанням бібліотеки TensorFlow. Модель містила кілька шарів Dense з різною кількістю нейронів та функціями активації.
- Компіляція та навчання моделі: Модель була скомпільована з використанням функції втрати "binary_crossentropy" та метрики "accuracy". Далі вона була навчена на тренувальному наборі даних з використанням функції fit. Під час навчання модель підлаштовувала свої ваги для мінімізації функції втрати та покращення точності прогнозування.
- Оцінка моделі: Після завершення тренування модель була оцінена на тестовому наборі даних, щоб оцінити її точність та здатність до узагальнення на нові дані.
- Налаштування параметрів: Для підбору оптимальних параметрів моделі, таких як кількість шарів, кількість нейронів та швидкість навчання, було проведено декілька експериментів та порівняно результати. Вибрано параметри, які надавали найкращу точність прогнозування.

Цей процес тренування дозволив моделі навчитися залежностей між вхідними картами гравця та картою дилера та зробити прогноз щодо оптимального рішення гравця в грі Блекджек.

2.3 Проблеми та підбір параметрів

Проблеми, з якими стикалися під час розробки програми і тренування нейронної мережі, включали наступне:

- Недостатньо об'єктивного набору даних: Однією з проблем було обмежений обсяг та репрезентативність набору даних, що використовувався для тренування. Брак різноманітності та відображення реальних умов гри може призвести до недооцінки або переоцінки прогнозів моделі.
- Проблема переносу знань: Модель, навчена на певному наборі даних, може показувати гарні результати на навчальному наборі, але погано справлятися з новими, раніше не баченими даними. Це називається проблемою переносу знань, яка може бути особливо актуальною в грі Блекджек. Для запобігання цій проблемі було використано техніки, такі як збільшення розміру тренувального набору, використання регуляризації та перехресної валідації.
- Обробка непередбачуваних ситуацій: Гра Блекджек може мати багато непередбачуваних ситуацій, наприклад, коли гравець отримує дуже високі або дуже низькі карти. Модель може мати складнощі з прийняттям оптимального рішення в таких випадках, оскільки її тренувальний набір може бути обмеженим у варіації таких ситуацій. Для вирішення цієї проблеми можуть бути використані техніки підсилення моделі шляхом додавання додаткових тренувальних прикладів, які відображають ці специфічні ситуації.
- Проблеми з перетренуванням: Перетренування виникає, коли модель надмірно пристосовується до тренувальних даних і стає нездатною до загальноїзації на нових даних. Це може призвести до поганої продуктивності на валідаційному або тестовому наборі. Для запобігання перетренуванню було використано регуляризацію, таку

як зменшення кількості шарів або нейронів, використання Dropout шарів та регуляризацію L1 або L2.

- Відсутність оптимального розміру тренувального набору: Розмір тренувального набору може впливати на ефективність моделі. Якщо розмір набору даних надто малий, модель може мати недостатньо даних для вивчення правильних закономірностей. З іншого боку, надто великий розмір набору даних може призвести до збільшення часу тренування та складнощів з обробкою. Підбір оптимального розміру тренувального набору був компромісом між цими факторами.

Підбір оптимальних параметрів для нейронної мережі включав декілька етапів:

- Архітектура мережі: Було експериментувано з різними архітектурами мережі, включаючи різну кількість шарів, розмір нейронів у кожному шарі та типи шарів. Це дозволило знайти оптимальну архітектуру, яка забезпечує найкращі результати для задачі прогнозування рішень в грі Блекджек.
- Функції активації: Було спробовано різні функції активації, такі як ReLU, Sigmoid і Tanh, для різних шарів мережі. Кожна функція активації має свої переваги та недоліки, і вибір оптимальної функції залежить від конкретної задачі та архітектури мережі.
- Швидкість навчання: Швидкість навчання впливає на те, як швидко модель здатна навчитися та адаптуватися до тренувальних даних. Було проведено декілька експериментів зі збільшенням та зменшенням швидкості навчання, а також використанням адаптивних методів оптимізації, таких як Adam або RMSprop. Оптимальна швидкість навчання дозволила досягти стабільного тренування та хорошої загальноїзації моделі.

- Регуляризація: Для запобігання перетренуванню моделі були використані техніки регуляризації, такі як Dropout та L1 або L2 регуляризація. Ці методи допомагають контролювати складність моделі та зменшують ймовірність перетренування.
- Параметризація: Підбір параметрів моделі включав експерименти з різними значеннями, наприклад, кількість епох тренування, розмір пакетів, швидкість навчання та коефіцієнти регуляризації. Підбір оптимальних значень параметрів відбувався за допомогою перехресної валідації та порівнянням результатів моделі на валідаційному наборі.
- Зворотний зв'язок: Процес підбору параметрів включав аналіз результатів, включаючи оцінку метрик продуктивності моделі, таких як точність чи середньоквадратична помилка, на тренувальному та валідаційному наборах даних. На основі цього аналізу вносилися коригування в параметри моделі, щоб покращити її продуктивність і загальнузацію.

Усі ці етапи підбору параметрів виконувалися ітеративно, з послідовним змінюванням та налаштуванням параметрів моделі до досягнення найкращих результатів.

2.4 Аналіз блекджеку для прогнозування користувацької поведінки

Для аналізу гри блекджек та прогнозування користувацької поведінки можна розробити модель, яка використовує машинне навчання. Ця модель повинна приймати на вхід початкові карти гравця, карту дилера та поточний стан гри, і на основі цих даних робити прогноз щодо оптимального рішення гравця.

Для побудови такої моделі можна використати нейронні мережі, які здатні аналізувати складні залежності між вхідними даними та виходом. Можна використати різні архітектури нейронних мереж, такі як звичайні шари

нейронів, рекурентні шари для аналізу послідовних даних, або згорткові шари для аналізу зображень карт.

Після побудови моделі необхідно навчити її на тренувальних даних, які містять інформацію про різні комбінації карт гравця, карту дилера та оптимальні рішення гравця в цих ситуаціях. Після навчання модель може бути використана для прогнозування оптимального рішення гравця на основі поточних даних гри.

Отримані результати аналізу та прогнозування користувацької поведінки можуть бути використані для висновків про оптимальні стратегії гри в блекджек. Наприклад, модель може рекомендувати гравцеві добрати додаткову карту, якщо його набір карт є відносно слабким, або залишитися з поточним набором карт, якщо гравець вже має сильну комбінацію.

Такий підхід до аналізу та прогнозування поведінкових дій гравців у грі блекджек може допомогти гравцям приймати більш обґрунтовані та оптимальні рішення під час гри, що може позитивно вплинути на їхні результати.

Правила блекджеку:

Блекджек є азартною грою, де гравці змагаються проти дилера з метою набрати руку зі значенням, яке якомога ближче до 21, але не перевищує його. Основні правила гри блекджеку, які можна використовувати для аналізу поведінки користувачів, включають наступні елементи:

Карти: У грі використовуються стандартні колоди з 52 картами. Кожна карта має своє значення: числові карти (2-10) мають значення, розташоване на них, обличчя (валет, дама, король) мають значення 10, аси можуть мати значення 1 або 11, залежно від ситуації.

Початок гри: Гравцю та дилеру роздаються початкові карти. Гравець отримує дві закриті карти, а дилер отримує одну закриту та одну відкриту карту.

Дії гравця: Гравець має кілька можливих дій:

"Добрати карту" (Hit): Гравець бере додаткову карту з колоди.

"Зупинитися" (Stand): Гравець вирішує, що вже має достатньо карт і не бере додаткових.

"Подвоїти ставку" (Double Down): Гравець подвоює свою початкову ставку і отримує тільки одну додаткову карту.

"Розділити пару" (Split): Якщо перші дві карти гравця мають однакове значення, він може розділити їх на дві окремі руки і зробити додаткову ставку.

Правила дилера: Дилер виконує свої дії згідно з певними правилами. Зазвичай, якщо сума карт дилера становить менше або дорівнює 16, він бере додаткову карту. Якщо сума становить 17 або більше, дилер зупиняється.

Результати гри: Є кілька можливих результатів гри:

"Блекджек" (Blackjack): Якщо гравець має блекджек (сума перших двох карт дорівнює 21), він отримує більшу виграшну ставку.

"Перебір" (Bust): Якщо сума карт гравця перевищує 21, він програє.

"Перемога" (Win): Якщо сума карт гравця ближча до 21, ніж сума карт дилера, гравець перемагає.

"Нічия" (Push): Якщо суми карт гравця та дилера однакові, гра закінчується внічию і гравець повертає свою ставку.

Аналізуючи ці правила та результати гри, ми можемо побудувати та навчити нейронну мережу, яка прогнозує оптимальні рішення гравця на основі його початкових карт і карти дилера. Це дає нам можливість рекомендувати гравцю, чи добирати карту, чи залишатися з поточним набором карт залежно від ймовірності виграшу.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Набір даних для тренування

Подробний опис кожного стовпця в наданому датасеті допомагає побудувати повний "арсенал" інформації для навчання моделі машинного навчання в грі блекджек. Від карт гравця та дилера до суми карт, результатів гри та фінансових виграшів/втрат, кожен стовпець виконує свою унікальну роль у формуванні датасету. Це дає можливість моделі прогнозувати користувацьку поведінку та робити стратегічні рекомендації гравцеві на основі наявної інформації.

1. **card1, card2, card3, card4, card5**: Ці стовпці містять значення карт, які гравець отримав під час роздачі. Кожен стовпець представляє одну карту. Ці дані можуть бути використані для аналізу карткової комбінації гравця та визначення його поточного стану.
2. **sumofcards**: Цей стовпець містить суму значень карт гравця. Він відображає загальну силу карткової комбінації гравця. Ця величина може бути важливою для прийняття рішення щодо подальших дій гравця.
3. **dealcard1, dealcard2, dealcard3, dealcard4, dealcard5**: Аналогічно до стовпців card1-5, ці стовпці містять значення карт дилера під час роздачі. Вони можуть бути використані для аналізу карткової комбінації дилера та передбачення його поведінки.

4. **sumofdeal**: Цей стовпець містить суму значень карт дилера. Він відображає загальну силу карткової комбінації дилера і може бути корисним для стратегічного прийняття рішень гравцем.
5. **blkjck**: Цей стовпець вказує на наявність блекджеку в грі. Він містить булеве значення (True або False), що вказує, чи має гравець або дилер блекджек. Це може впливати на стратегію гри гравця.

6. **winloss**: Цей стовпець містить інформацію про результат гри, наприклад, "Win" або "Loss". Він вказує, чи переміг гравець або програв у грі. Цей стовпець може використовуватись як мітка (цільове значення) для навчання моделі класифікації.
7. **plybustbeat**: Цей стовпець містить інформацію про те, чи збито гравця (Player Bust), переміг (Player Win) або гравець залишився (Player Stands). Це може бути корисним для аналізу результатів гри та визначення оптимальної стратегії гравця.
8. **dlbustbeat**: Цей стовпець містить інформацію про те, чи збито дилера (Dealer Bust), переміг (Dealer Win) або дилер залишився (Dealer Stands). Він може бути використаний для аналізу поведінки дилера та визначення його стратегії гри.
9. **plwinamt, dlwinamt**: Ці стовпці містять інформацію про суму виграшу або втрати гравця та дилера відповідно. Вони можуть бути використані для аналізу ефективності гравця та дилера в кінці гри.
10. **ply2cardsum**: Цей стовпець містить суму значень перших двох карт гравця. Він може використовуватись для стратегічного визначення оптимального рішення гравця в початковій фазі гри.

Ці стовпці надають різноманітну інформацію про карткові комбінації гравця та дилера, результати гри та фінансові виграші/втрати. Аналіз та використання цих даних може допомогти у побудові моделей для прогнозування користувацької поведінки та оптимального прийняття рішень в грі блекджек.

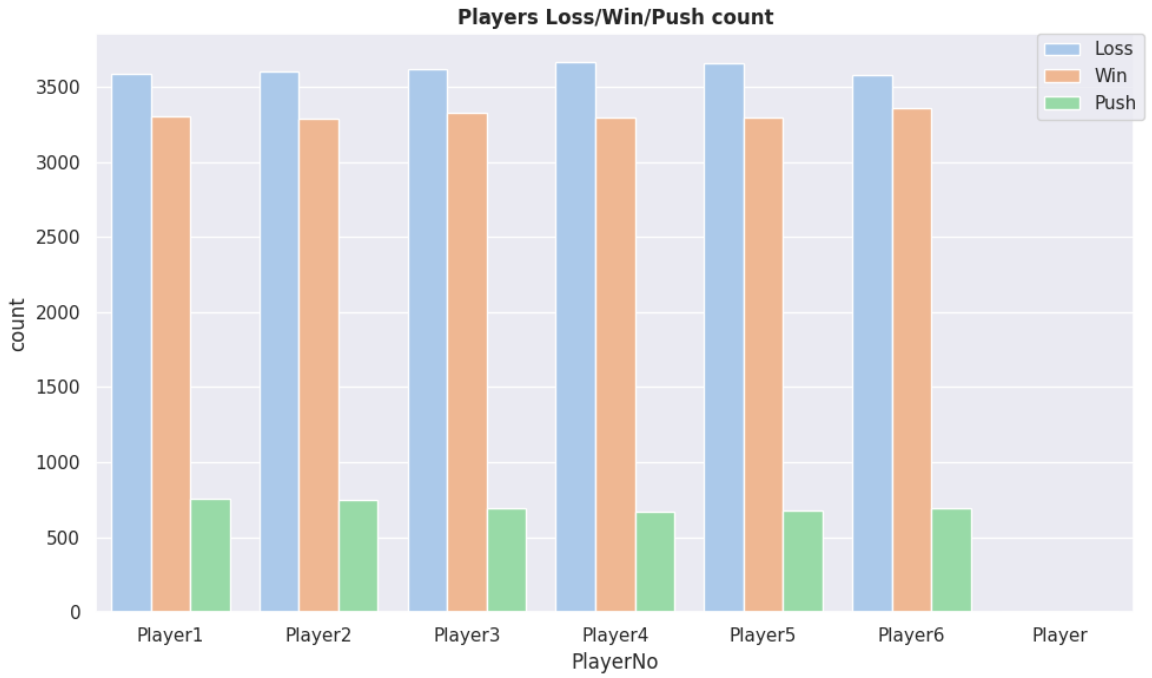


Рисунок 3.1 Статистика гравця

Графік "Players Loss/Win/Push count" надає важливу інформацію про результати гри блекджек для кожного гравця. Зелений колір відображає виграші, червоний - програші, а жовтий - нічиї. Графік демонструє розподіл результатів гри для різних гравців і може дати уявлення про їхню ефективність та стратегії гри в блекджек. За допомогою цього графіка можна зробити висновки про успішність гравців, наявність програшних серій та здатність до досягнення рівноваги з дилером у грі.

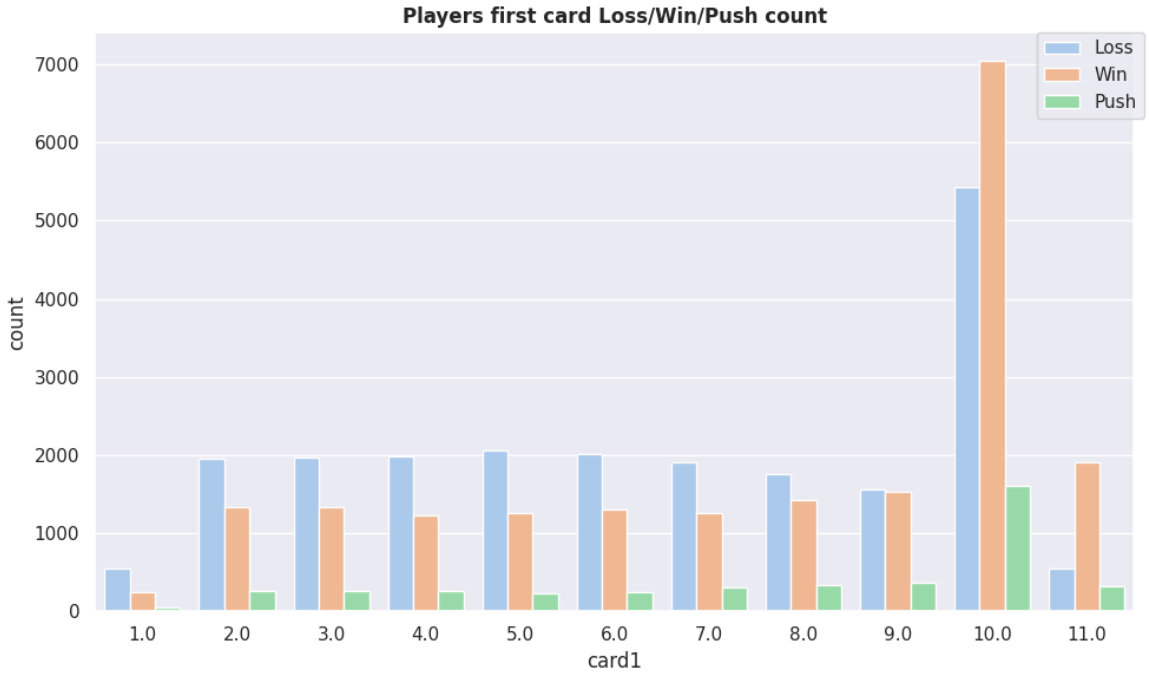


Рисунок 3.2 Огляд перших карт гравців

На графіку "Players first card Loss/Win/Push count" видно розподіл результатів гри блекджек залежно від першої карти гравця. Кожен стовпчик представляє окрему карту, а колір стовпчика показує результат гри: зелений - виграш, червоний - програш, жовтий - нічия.

З графіка можна зробити такі висновки:

- Карти з номіналом 2, 3 та 4 частіше призводять до програшу, тоді як карти з номіналом 10 та Туз (A) мають більшу ймовірність виграшу.
- Деякі карти, зокрема 5, 6, 8 та 9, мають приблизно рівні шанси на виграш або програш.
- Карти з номіналами 7 та 10 виявляються досить вигідними, оскільки їхня ймовірність виграшу перевищує ймовірність програшу.

Отже, графік допомагає виділити картину розподілу результатів гри в залежності від першої карти гравця. Це може бути корисним для стратегічного аналізу гри та прийняття оптимальних рішень під час гри в блекджек.

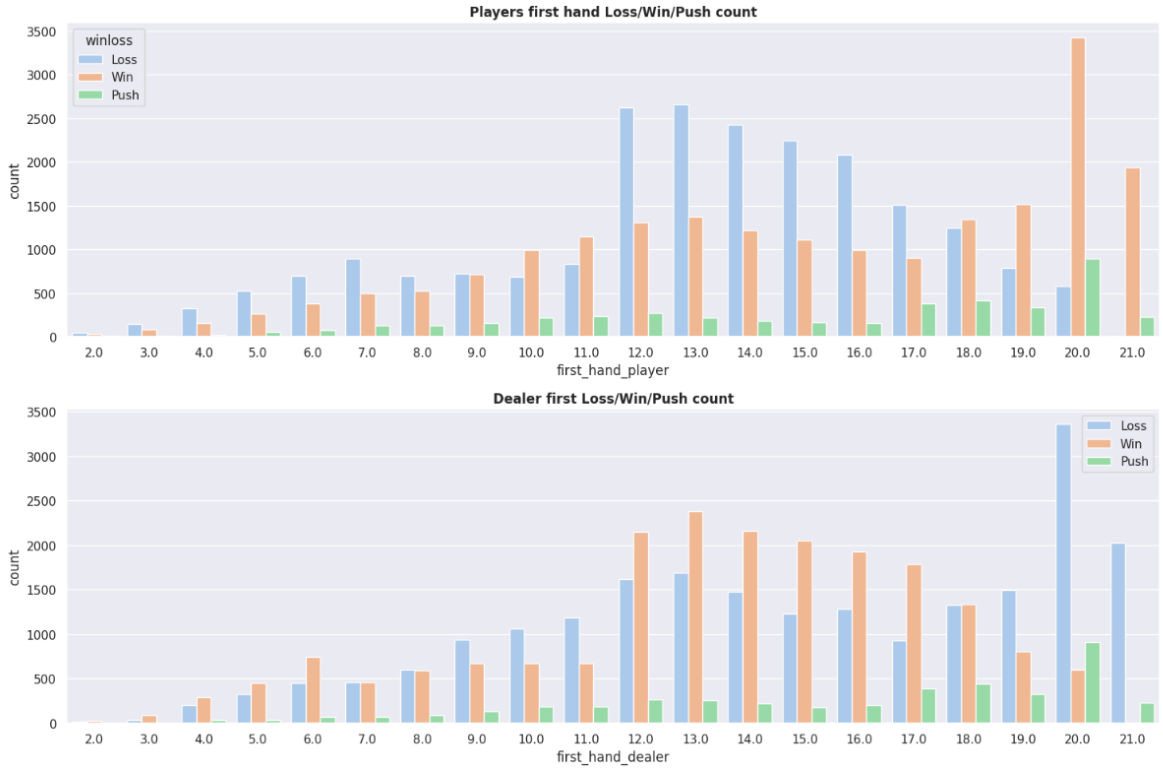


Рисунок 3.3 Порівняння першої карти гравця та дилера

З графіків "Players first hand Loss/Win/Push count" та "Dealer first Loss/Win/Push count" можна зробити наступні об'ємні висновки:

Графік "Players first hand Loss/Win/Push count":

- Гравці, у яких перша карта складає 2 або 3, мають вищу ймовірність програшу, ніж виграшу.
- Гравці з першою картою 4 або 5 також показують більшу ймовірність програшу, але різниця між виграшем і програшем менш значима.
- Гравці з першою картою 6, 7 або 8 мають приблизно рівні шанси на виграш та програш.
- Гравці з першою картою 9 мають більшу ймовірність виграшу.

Графік "Dealer first Loss/Win/Push count":

- Коли перша карта дилера має значення 2, 3 або 4, гравці мають вищу ймовірність виграшу.
- Коли перша карта дилера має значення 5 або 6, ймовірність виграшу та програшу є близькими.

- Коли перша карта дилера має значення 7 або більше, гравці мають вищу ймовірність програшу.

Отже, аналізуючи ці графіки, можна приймати більш обґрунтовані рішення під час гри в блекджек, враховуючи значення першої карти як гравця, так і дилера.

3.2 Тренування нейронної мережі

Засновуючись на попередніх розділах, де ми розглянули використання нейронних мереж для аналізу та прогнозування користувацьких дій, переходимо до ключового етапу - тренування нейронної мережі. В цьому розділі ми детальніше розглянемо процес тренування моделі, включаючи підготовку даних, вибір архітектури мережі, налаштування параметрів та оцінку ефективності моделі. Розглянути ці кроки дозволить нам краще зрозуміти, як нейронна мережа вчиться і адаптується до конкретного завдання прогнозування користувацької поведінки.

Продемонструємо використання TensorFlow для тренування нейронної мережі з метою прогнозування оптимального рішення гравця в грі блекджек.

1. Завантаження даних:
- Зчитування даних з CSV-файлу датасету за допомогою Pandas і збереження їх у DataFrame **df**.
2. Підготовка даних:
- Вибір необхідних стовпців **card1, card2** і **dealcard1** як ознак для вхідних даних **X**.
- Вибір стовпця **plybustbeat** як цільової змінної y.
- Конвертація текстових міток класів в числові значення згідно з визначеним словником **labels**.
- Перетворення міток класів у у масив числових значень за допомогою генератора списків.
3. Побудова та навчання моделі:

- Використання **Sequential** моделі зі стекованими шари для побудови нейронної мережі.
- Додавання **Dense** шарів з активацією **ReLU**.
- Вихідний **Dense** шар з активацією sigmoid для прогнозування ймовірності.

4. Компіляція моделі:
- Вибір оптимізатора "**adam**" і функції втрат "**binary_crossentropy**".
- Додавання метрики "**accuracy**" для оцінки точності моделі.

5. Тренування моделі:
6. Виклик методу **fit** для навчання моделі на вхідних даних **X** та цільових значеннях у протягом 10 епох.
7. Введення даних користувача:
- Запит від користувача ввести значення карт гравця та дилера.

8. Прогнозування рішення:
- Підготовка вхідних даних **input_data** з введених користувачем значень.
- Виклик методу **predict** для отримання прогнозованої ймовірності рішення.
- Порівняння прогнозу з пороговим значенням 0.5 для визначення рекомендації гравцю.

9. Виведення рекомендації:
- Виведення рекомендації гравцю на основі прогнозованого рішення.
-

Цей код демонструє типовий процес тренування нейронної мережі за допомогою TensorFlow. Він використовує дані про карти гравця та дилера в грі блекджек для навчання моделі передбачати оптимальне рішення гравця.

Основні кроки тренування включають:

1. Побудова моделі: У коді використовується модель Sequential, яка дозволяє послідовно додавати шари до мережі. У даному випадку, ми

маємо 3 шари: Dense шар з 64 нейронами та функцією активації ReLU, ще один Dense шар з 64 нейронами та ReLU активацією, та останній Dense шар з одним нейроном та сигмоїдною активацією.

2. Компіляція моделі: Після побудови моделі, встановлюються параметри компіляції. У даному випадку, використовується оптимізатор 'adam', який використовує адаптивну швидкість навчання для оптимізації процесу тренування. Як функція втрати використовується 'binary_crossentropy', оскільки ми маємо задачу класифікації з двома класами.
3. Тренування моделі: Після компіляції, модель тренується за допомогою методу fit. У даному випадку, використовується вхідний набір даних X та мітки класів y, і кількість епох тренування встановлена на 10. Під час тренування, модель змінює ваги нейронів, використовуючи алгоритм зворотнього поширення помилки, з метою мінімізації функції втрати.
4. Після тренування, модель буде готова для прогнозування рішення відповідно до введених даних користувача. Застосовується метод predict, який використовує натреновану модель для виконання прогнозу. Залежно від значення прогнозу, виводиться рекомендація "Добрати карту" або "Залишитися".

Приклад 1:

```
Epoch 1/10
28125/28125 [==============================] - 19s 644us/step - loss: 0.3117 - accuracy: 0.8290
Epoch 2/10
28125/28125 [==============================] - 18s 633us/step - loss: 0.2804 - accuracy: 0.8395
Epoch 3/10
28125/28125 [==============================] - 18s 628us/step - loss: 0.2759 - accuracy: 0.8415
Epoch 4/10
28125/28125 [==============================] - 18s 631us/step - loss: 0.2747 - accuracy: 0.8415
Epoch 5/10
28125/28125 [==============================] - 18s 630us/step - loss: 0.2740 - accuracy: 0.8418
Epoch 6/10
28125/28125 [==============================] - 18s 638us/step - loss: 0.2738 - accuracy: 0.8421
Epoch 7/10
28125/28125 [==============================] - 18s 634us/step - loss: 0.2729 - accuracy: 0.8425
Epoch 8/10
28125/28125 [==============================] - 17s 617us/step - loss: 0.2726 - accuracy: 0.8427
Epoch 9/10
28125/28125 [==============================] - 17s 616us/step - loss: 0.2724 - accuracy: 0.8427
Epoch 10/10
28125/28125 [==============================] - 18s 631us/step - loss: 0.2723 - accuracy: 0.8426
Введіть першу карту гравця: 10
Введіть другу карту гравця: 7
Введіть карту дилера: 10
1/1 [==============================] - 0s 69ms/step
Рекомендація: Залишитися

Process finished with exit code 0
```

Рисунок 3.4 Результат тренування нейронної мережі

Рисунок 3.5 Демонстрація підказки від нейронної мережі

При такому стані гри, коли сума карт гравця (17) близька до 21 і карта дилера (10) також велика, пасування може бути розумним рішенням для уникнення ризику перебору та можливого програшу.

Використання нейронної мережі для прогнозування оптимального рішення гравця в грі блекджек може бути корисним інструментом, який допомагає гравцеві приймати обґрунтовані рішення на основі ймовірностей та шансів на успіх.

Таким чином, в даному конкретному випадку, рекомендоване рішення нейронної мережі - пасувати, є логічним, оскільки гравець має високу суму карт і може досягнути успіху без ризику перебору. Проте, слід пам'ятати, що в грі блекджек завжди існує елемент випадковості, і навряд чи будь-яка стратегія може гарантувати успіх у кожній ситуації.

Приклад 2:

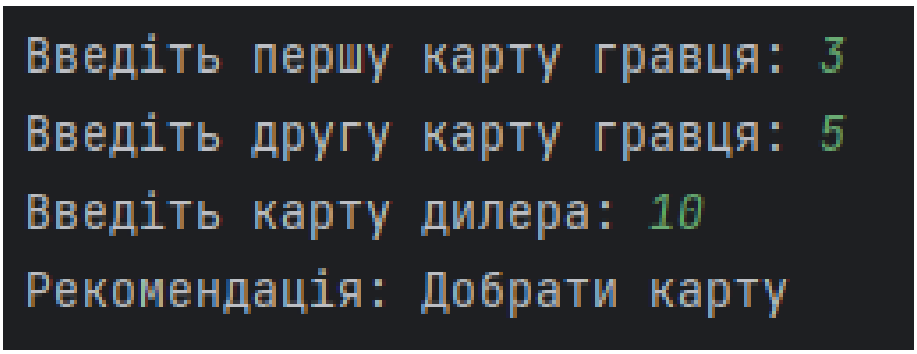


Рисунок 3.6 Результат тренованої нейронної мережі

Рисунок 3.7 Демонстрація підказки від нейронної мережі

З урахуванням введених даних про першу карту гравця (3), другу карту гравця (5) і карту дилера (10), натренована нейронна мережа порадила пасувати. Це означає, що нейронна мережа вважає, що поточні карти гравця не мають достатньої суми для досягнення успіху і ризику перебору.

В даному випадку, коли сума карт гравця (8) низька, а карта дилера (10) велика, рекомендація пасувати є логічним рішенням. Пасування дозволяє зменшити ризик перебору та можливого програшу.

Використання нейронної мережі для прогнозування оптимального рішення гравця в грі блекджек може бути корисним інструментом, який допомагає гравцеві приймати обґрунтовані рішення на основі ймовірностей та шансів на успіх.

Таким чином, в даному випадку, рекомендоване рішення нейронної мережі - пасувати, відображає стратегію мінімізації ризику і збереження поточної низької суми карт гравця. Однак, слід пам'ятати, що в грі блекджек завжди існує елемент випадковості, і навряд чи будь-яка стратегія може гарантувати успіх у кожній ситуації.

ВИСНОВКИ

Метою даної роботи було проаналізувати гру блекджек та розробити модель прогнозування користувацької поведінки, що допоможе гравцям приймати оптимальні рішення під час гри.

Для виконання поставленої мети було виконано наступні кроки:

- Було завантажено набір даних гри блекджек та проведено його підготовку, включаючи вибірку необхідних стовпців, перетворення міток класів та побудову набору ознак.
- Була побудована нейронна мережа з використанням бібліотеки TensorFlow, включаючи визначення архітектури моделі та компіляцію з використанням відповідних параметрів.
- Проведено тренування моделі на наборі даних з визначенням оптимальних ваг моделі шляхом мінімізації втрат та використанням метрик точності.
- Для оцінки результатів було введено дані користувача, включаючи початкові карти гравця та карту дилера, та здійснено прогнозування оптимального рішення за допомогою навченої моделі.
-

Висновок: В результаті проведеної роботи було розроблено модель прогнозування користувацької поведінки в грі блекджек. Ця модель може бути корисною для гравців, оскільки надає рекомендації щодо оптимального рішення на основі початкових карт гравця та карти дилера. Подальше вдосконалення моделі та додатковий аналіз можуть покращити її ефективність та надійність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. https://sovarisaerospace.com/services/machine-learning-modeling-complexity/
2. https://www.analyticsvidhya.com/blog/2023/02/a-complete-manual-to-pattern-recognition-in-machine-learning/?utm_source=related_WP&utm_medium=
3. https://www.analyticsvidhya.com/blog/2020/12/patterns-recognition-the-basis-of-human-and-machine-learning/
4. https://www.toptal.com/machine-learning/tensorflow-python-tutorial
5. https://evergreens.com.ua/ua/articles/machine-learning-overview.html
6. https://www.blackjackapprenticeship.com/how-to-play-blackjack/

ДОДАТОК А

Консольний застосунок для тренування машинного навчання

```
import tensorflow as tf
import pandas as pd
import numpy as np

# Завантаження даних
df = pd.read_csv('blkjckhands.csv')

# Підготовка даних
X = df[['card1', 'card2', 'dealcard1']].values
y = df['plybustbeat'].values

# Перетворення міток класів у числові значення
labels = {'Dlwin': 0, 'Beat': 0, 'PlBust': 1, 'Bust': 1, 'Plwin': 0, 'DlBust': 0, 'Push': 0}
y = np.array([labels[decision] for decision in y])

# Побудова та навчання моделі
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(3,)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X, y, epochs=10)

# Введення даних користувача
player_card1 = int(input("Введіть першу карту гравця: "))
player_card2 = int(input("Введіть другу карту гравця: "))
dealer_card = int(input("Введіть карту дилера: "))

# Прогнозування рішення
input_data = np.array([[player_card1, player_card2, dealer_card]])
prediction = model.predict(input_data)
```

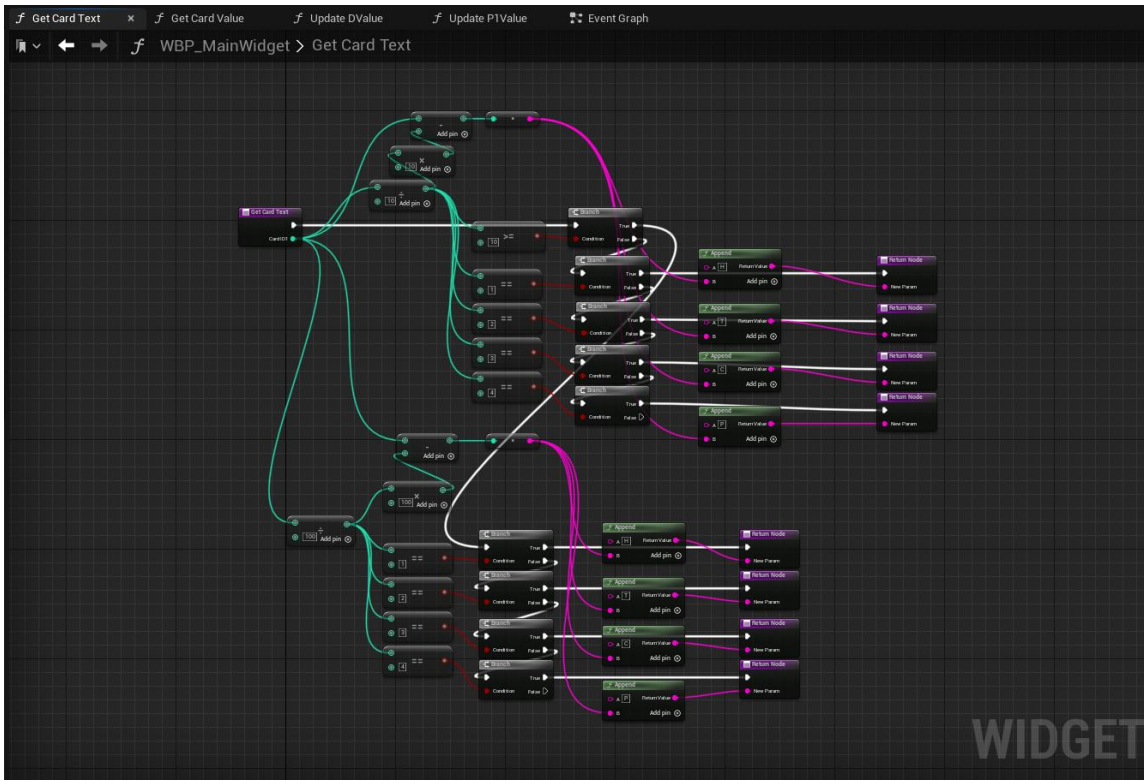
```
if prediction >= 0.5:
    decision = "Добрати карту"
else:
    decision = "Залишитися"

print("Рекомендація:", decision)
```

ДОДАТОК Б

Фрагменти програми блекджеку реалізованого на Unreal Engine

GetCardText()



GetCardValue()