НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
КАФЕДРА ІНФОРМАТИКИ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

КУРСОВА РОБОТА

з дисципліни «Аналіз даних в інформаційних системах»

на тему: «Прогнозування переможця у раунді на основі показників стану ігрового поля у грі Counter-Strike: Global Offensive методами: Random Forest, Decision Tree та K-Nearest Neighbors»

Студента 2 курсу групи ІП-13
Спеціальності: 121
«Інженерія програмного забезпечення»
Паламарчука Олександра Олександровича

«ПРИЙНЯВ» з оцінкою

_____

–

доц. Ліхоузова Т.А. / доц. Олійник Ю.О.

_____
Підпис Дата

Київ - 2023 рік

Національний технічний університет України “КПІ ім. Ігоря Сікорського”

Кафедра інформатики та програмної інженерії

Дисципліна Аналіз даних в інформаційно-управляючих системах

Спеціальність 121 "Інженерія програмного забезпечення"

Курс 2 Група ІП-13 Семестр 4

ЗАВДАННЯ

на курсову роботу студента

Паламарчука Олександра Олександровича

1.Тема роботи Прогнозування переможця у раунді на основі показників стану ігрового поля у грі Counter-Strike: Global Offensive методами: Random Forest, Decision Tree та K-Nearest Neighbors

2.Строк здачі студентом закінченої роботи 08.06.2023

3. Вхідні дані до роботи методичні вказівки до курсової робота, обрані дані з сайту https://www.kaggle.com/datasets/christianlillelund/csgo-round-winner-classification

4.Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1.Постановка задачі

2.Аналіз предметної області

3.Робота з даними

4.Інтелектуальний аналіз даних

5.Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

6.Дата видачі завдання 09.02.2023

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсової роботи	Термін виконання етапів роботи	Підписи керівника, студента
1.	Отримання теми курсової роботи	09.02.2023	
2.	Визначення зовнішніх джерел даних	01.03.2023	
3.	Пошук та вивчення літератури з питань курсової роботи	02.04.2023	
5.	Обґрунтування методів інтелектуального аналізу даних	02.05.2023	
6.	Застосування та порівняння ефективності методів інтелектуального аналізу даних	04.06.2023	
7.	Підготовка пояснювальної записки	06.06.2023	
8.	Здача курсової роботи на перевірку	08.06.2023	
9.	Захист курсової роботи	09.06.2023	

Студент Паламарчук О.О.

(підпис) (прізвище, ім'я, по батькові)

Керівник доц. Ліхоузова Т.А

(підпис) (прізвище, ім'я, по батькові)

Керівник _____ доц. Олійник Ю.О.

(підпис) (прізвище, ім'я, по батькові)

"8" червня 2023 р.

АНОТАЦІЯ

Пояснювальна записка до курсової роботи: 33 сторінок, 27 рисунків, 2 таблиці, 9 посилань.

Об’єкт дослідження: інтелектуальний аналіз даних.

Предмет дослідження: створення програмного забезпечення, що проводить аналіз даних з подальшим прогнозуванням та графічним відображенням результатів.

Мета роботи: пошук, обробка та аналіз даних, реалізація програмного забезпечення для роботи з даними, їх подальшого аналізу та прогнозування.

Дана курсова робота включає в себе: опис створення програмного забезпечення для інтелектуального аналізу даних, їх графічного відображення та прогнозування за допомогою різних моделей.

МОДЕЛЬ ПРОГНОЗУВАННЯ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, RANDOM FOREST CLASSIFIER, DECISION TREE CLASSIFIER, K-NEAREST NEIGHBORS.

ЗМІСТ

ВСТУП

Сьогодні все більше і більше набирає популярності гра Counter-Strike: Global Offensive, на даний момент вона вважається однією з найпопулярніших і найперспективніших ігор. Кожного дня у неї грають одномоментно в середньому мільйон гравців з різних куточків світу а піковий онлайн становить 1.8 мільйонів гравців. Це тисячі матчів, сотні тисяч транзакцій, мільйони інформації, яку можна аналізувати, і це лише за день. Також ця гра входить до складу змагальних ігор (кіберспорт). З 2000-го року, починаючи ще з молодшої серії гри Counter-Strike 1.6, кожного року між собою змагаються найкращі команди за титули, а гравці прагнуть стати найкращими у світі за статистичними показниками. Усі великі турніри транслюються на стрімінгових платформах (Twitch, YouTube і тд.), які одномоментно переглядають десятки тисяч глядачів. Кожного дня тисячі людей роблять прогнози на професійні матчі Counter-Strike.

В ході роботи ми проаналізуємо вхідні дані, проведемо їх чистку, а потім за допомогою моделей класифікуємо на переможців і переможених команд у раунді.

Для завантаження, попередньої обробки, а також для побудови моделей було використано такі технології: Python[1], Pandas[2], Matplotlib[3], Sklearn[4], Seaborn[5], NumPy[6].

1. ПОСТАНОВКА ЗАДАЧІ

Етапами та головними задачами курсової роботи є: аналіз предметної області, завантаження, опис та обробка даних, первинний аналіз даних, поділ даних для навчання моделі, вибір методів для прогнозування, аналіз та порівняння результатів кожного методу.

Створення застосунку, який буде приймати дані відповідної структури, чистити, трансформувати, аналізувати, відображати потрібні графіки та будувати моделі для класифікації даних на два класи: CT (Counter-Terrorists), T (Terrorists), тобто яка із сторін переможе у раунді.

Прогнозування виконується за допомогою моделей для класифікації, а саме: Random Forest, Decision Tree та K-Nearest Neighbors. Потрібно знайти оптимальні параметри моделей, при яких показники mean squared error, R2 score та accuracy будуть найкращими. Далі потрібно виконати порівняння оцінок усіх моделей та знайти найкращу для найбільш точного прогнозування результатів.

Вхідними параметрами є snapshot ігрового поля у конкретний період часу раунду, який включає в себе інформацію про карту, кількість здоров’я, гравців, зброї певного виду, гранат певного виду, броні певного виду, наборів для знешкодження вибухівки, часу, який залишився до закінчення раунду, грошей, чи встановлено бомбу, поточний рахунок та переможця раунду.

2. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Counter-Strike: Global Offensive (CS:GO) є однією з найпопулярніших комп'ютерних шутерних ігор у світі, яка має велике вплив на галузь кіберспорту та ігрову спільноту.

Геймплей: гравці розподіляються на дві команди - Counter-Terrorists (CT) та Terrorists (T). CT-гравці повинні запобігти T-гравцям виконати завдання, такі як розмінування вибухівки.

Карти: кожна карта має свою унікальну структуру та ключові позиції, що вимагає від гравців адаптації до різних умов та тактик.

Зброя та екіпірування: гравці можуть обирати різні види стрілецької зброї а також різні види гранат.

Кіберспорт: CS:GO має велику професійну кіберспортивну сцену з численними турнірами, лігами та чемпіонатами, де найкращі гравці та команди змагаються за призові фонди та почесні звання.

Економіка гри: CS:GO має внутрішню економічну систему, де гравці можуть отримувати гроші за успішне виконання завдань та перемоги в раундах. Ці гроші використовуються для покупки зброї, екіпірування та стратегічних рішень.

3. ПОПЕРЕДНЯ РОБОТА З ДАНИМИ

3.1 Опис даних

Для інтелектуального аналізу даних із сайту https://www.kaggle.com було обрано датасет https://www.kaggle.com/code/saabet/csgo-round-winner-prediction. Датасет містить 1 файл, який складається з 97 колонок:

1. time_left - час, який залишився до кінця раунда.
2. ct_score - кількість виграних раундів стороною CT.
3. t_score - кількість виграних раундів стороною T.
4. map - карта, на якій змагаються команди.
5. bomb_planted - прапорець, який показує чи встановлення бомба.
6. ct_health - сума здоров'я гравців команди CT.
7. t_health - сума здоров'я гравців команди T.
8. ct_armor - сума броні гравців команди CT.
9. t_armor - сума броні гравців команди T.
10. ct_money - сума грошей гравців команди CT.
11. t_money - сума грошей гравців команди T.
12. ct_helmets - кількість шоломів команди CT.
13. t_helmets - кількість шоломів команди T.
14. ct_defuse_kits - кількість наборів для знешкодження вибухівки команди CT.
15. ct_players_alive - кількість живих гравців команди CT.
16. t_players_alive - кількість живих гравців команди T.
17. ct_weapon_* - кількість зброї певного типу команди CT, де * - це назва зброї.
18. t_weapon_* - кількість зброї певного типу команди T, де * - це назва зброї.
19. ct_grenade_* - кількість гранат певного типу команди CT, де * - це назва гранати.
20. t_grenade_* - кількість гранат певного типу команди T, де * - це назва гранати.
21. round_winner - сторона, що перемогла у раунді, CT або T.

3.2 Огляд та обробка даних.

Імпортуємо усі потрібні бібліотеки (рис. 3.1).

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix
```

Рисунок 3.1

За допомогою бібліотеки Pandas[2] прочитаємо дані у dataframe з файла ‘csgo_round_snapshots.csv’ (рис. 3.2) та виведемо інформацію про dataframe на екран (рис. 3.3, 3.4, 3.5, 3.6).

```
df = pd.read_csv('resources/csgo_round_snapshots.csv')
df.info()
```

Рисунок 3.2

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 122410 entries, 0 to 122409
Data columns (total 97 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   time_left                   122410 non-null  float64
 1   ct_score                    122410 non-null  float64
 2   t_score                     122410 non-null  float64
 3   map                         122410 non-null  object
 4   bomb_planted                122410 non-null  bool
 5   ct_health                   122410 non-null  float64
 6   t_health                    122410 non-null  float64
 7   ct_armor                    122410 non-null  float64
 8   t_armor                     122410 non-null  float64
 9   ct_money                    122410 non-null  float64
 10  t_money                     122410 non-null  float64
 11  ct_helmets                  122410 non-null  float64
 12  t_helmets                   122410 non-null  float64
 13  ct_defuse_kits              122410 non-null  float64
 14  ct_players_alive            122410 non-null  float64
 15  t_players_alive             122410 non-null  float64
 16  ct_weapon_ak47              122410 non-null  float64
 17  t_weapon_ak47               122410 non-null  float64
 18  ct_weapon_aug               122410 non-null  float64
 19  t_weapon_aug                122410 non-null  float64
 20  ct_weapon_awp               122410 non-null  float64
 21  t_weapon_awp                122410 non-null  float64
 22  ct_weapon_bizon             122410 non-null  float64
 23  t_weapon_bizon              122410 non-null  float64
 24  ct_weapon_cz75auto          122410 non-null  float64
```

Рисунок 3.3

```
25  t_weapon_cz75auto             122410 non-null  float64
26  ct_weapon_elite               122410 non-null  float64
27  t_weapon_elite                122410 non-null  float64
28  ct_weapon_famas               122410 non-null  float64
29  t_weapon_famas                122410 non-null  float64
30  ct_weapon_g3sg1               122410 non-null  float64
31  t_weapon_g3sg1                122410 non-null  float64
32  ct_weapon_galilar             122410 non-null  float64
33  t_weapon_galilar              122410 non-null  float64
34  ct_weapon_glock               122410 non-null  float64
35  t_weapon_glock                122410 non-null  float64
36  ct_weapon_m249                122410 non-null  float64
37  t_weapon_m249                 122410 non-null  float64
38  ct_weapon_m4a1s               122410 non-null  float64
39  t_weapon_m4a1s                122410 non-null  float64
40  ct_weapon_m4a4                122410 non-null  float64
41  t_weapon_m4a4                 122410 non-null  float64
42  ct_weapon_mac10               122410 non-null  float64
43  t_weapon_mac10                122410 non-null  float64
44  ct_weapon_mag7                122410 non-null  float64
45  t_weapon_mag7                 122410 non-null  float64
46  ct_weapon_mp5sd               122410 non-null  float64
47  t_weapon_mp5sd                122410 non-null  float64
48  ct_weapon_mp7                 122410 non-null  float64
49  t_weapon_mp7                  122410 non-null  float64
50  ct_weapon_mp9                 122410 non-null  float64
51  t_weapon_mp9                  122410 non-null  float64
52  ct_weapon_negev               122410 non-null  float64
53  t_weapon_negev                122410 non-null  float64
54  ct_weapon_nova                122410 non-null  float64
55  t_weapon_nova                 122410 non-null  float64
```

Рисунок 3.4

```
56  ct_weapon_p90                 122410 non-null  float64
57  t_weapon_p90                  122410 non-null  float64
58  ct_weapon_r8revolver          122410 non-null  float64
59  t_weapon_r8revolver           122410 non-null  float64
60  ct_weapon_sawedoff            122410 non-null  float64
61  t_weapon_sawedoff             122410 non-null  float64
62  ct_weapon_scar20              122410 non-null  float64
63  t_weapon_scar20               122410 non-null  float64
64  ct_weapon_sg553               122410 non-null  float64
65  t_weapon_sg553                122410 non-null  float64
66  ct_weapon_ssg08               122410 non-null  float64
67  t_weapon_ssg08                122410 non-null  float64
68  ct_weapon_ump45               122410 non-null  float64
69  t_weapon_ump45                122410 non-null  float64
70  ct_weapon_xm1014              122410 non-null  float64
71  t_weapon_xm1014               122410 non-null  float64
72  ct_weapon_deagle              122410 non-null  float64
73  t_weapon_deagle               122410 non-null  float64
74  ct_weapon_fiveseven           122410 non-null  float64
75  t_weapon_fiveseven            122410 non-null  float64
76  ct_weapon_usps                122410 non-null  float64
77  t_weapon_usps                 122410 non-null  float64
78  ct_weapon_p250                122410 non-null  float64
79  t_weapon_p250                 122410 non-null  float64
80  ct_weapon_p2000               122410 non-null  float64
81  t_weapon_p2000                122410 non-null  float64
82  ct_weapon_tec9                122410 non-null  float64
83  t_weapon_tec9                 122410 non-null  float64
84  ct_grenade_hegrenade          122410 non-null  float64
85  t_grenade_hegrenade           122410 non-null  float64
86  ct_grenade_flashbang          122410 non-null  float64
```

Рисунок 3.5

```
87  t_grenade_flashbang           122410 non-null  float64
88  ct_grenade_smokegrenade       122410 non-null  float64
89  t_grenade_smokegrenade        122410 non-null  float64
90  ct_grenade_incendiarygrenade  122410 non-null  float64
91  t_grenade_incendiarygrenade   122410 non-null  float64
92  ct_grenade_molotovgrenade     122410 non-null  float64
93  t_grenade_molotovgrenade      122410 non-null  float64
94  ct_grenade_decoygrenade       122410 non-null  float64
95  t_grenade_decoygrenade        122410 non-null  float64
96  round_winner                  122410 non-null  object
```

Рисунок 3.6

Можемо побачити, що всі дані було прочитано коректно, усі вони non-null, а також мають коректні типи даних.

Такі колонки як map, bomb_planted, round_winner мають типи даних string та bool, які не підходять для подальшого аналізу, тому перетворення значень цих колонок на числові.

Для початку отримаємо унікальні значення колонки map (рис 3.7).

```
df['map'].unique()
```

```
array(['de_dust2', 'de_mirage', 'de_nuke', 'de_inferno', 'de_overpass',
       'de_vertigo', 'de_train', 'de_cache'], dtype=object)
```

Рисунок 3.7

На даному етапі ми переконуємося, що усі назви карт є правильними і готовими для подальшої обробки.

За допомогою класу LabelEncoder з бібліотеки Sklean[4] перетворимо якісні характеристики колонки ‘map’ у числові та виведемо результат на екран (рис. 3.8).

```
label_encoder = preprocessing.LabelEncoder()
df['map'] = label_encoder.fit_transform(df['map'])
df['map'].unique()
```

```
array([1, 3, 4, 2, 5, 7, 6, 0])
```

Рисунок 3.8

Повторимо такі дії для колонок bomb_planted та round_winner (рис 3.9)

```
df['bomb_planted'].unique()
```

```
array([False,  True])
```

```
df['bomb_planted'] = df['bomb_planted'].astype(int)
df['bomb_planted'].unique()
```

```
array([0, 1])
```

```
df['round_winner'].unique()
```

```
array(['CT', 'T'], dtype=object)
```

```
df['round_winner'] = label_encoder.fit_transform(df['round_winner'])
df['round_winner'].unique()
```

```
array([0, 1])
```

Рисунок 3.9

Виведемо розподіл перемог для кожної сторони (рис. 3.10)

```
count = df['round_winner'].value_counts()
count.index = count.index.map({0: 'CT', 1: 'T'})
plt.pie(count, labels=count.index, autopct='%1.1f%%')
plt.title('Count CT and T wins')
plt.show()
```

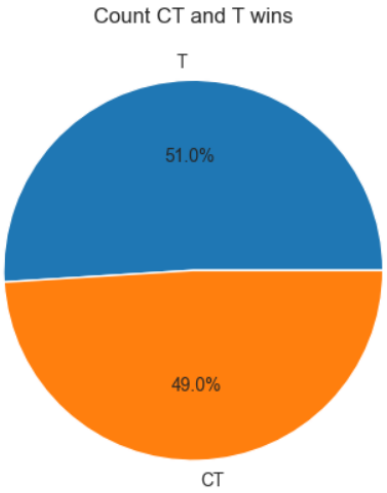

Рисунок 3.10

На даній круговій діаграмі можемо бачити, що розподіл перемог для обох сторін приблизно однаковий, однак сторона T має трохи більше перемог, що свідчить про її перевагу.

Побудуємо дві діаграми розмаху, перша показує розподіл кількості наборів для знешкодження вибухівки в залежності від факту перемоги і поразки сторони CT (рис. 3.11), друга показує розподіл кількості броні в залежності від факту результату раунду (рис. 3.12).

```
ct_kits_when_win = df[df['round_winner'] == 0]['ct_defuse_kits']
ct_kits_when_lose = df[df['round_winner'] == 1]['ct_defuse_kits']

plt.boxplot([ct_kits_when_win, ct_kits_when_lose], labels=['Win', 'Lose'])
plt.xlabel('Result')
plt.ylabel('Number of Kits')
plt.title('Number of Kits when CT wins and loses')
plt.show()
```

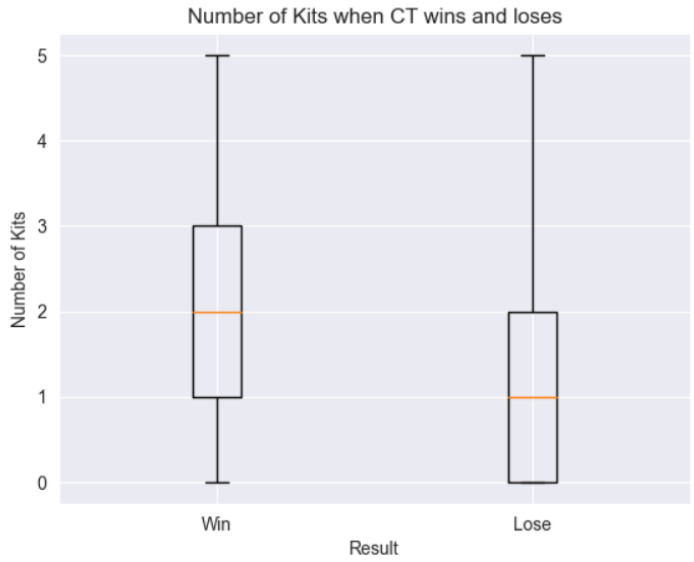

Рисунок 3.11

```
ct_armors_when_win = df[df['round_winner'] == 0]['ct_armor']
ct_armors_when_lose = df[df['round_winner'] == 1]['ct_armor']
t_armors_when_win = df[df['round_winner'] == 1]['t_armor']
t_armors_when_lose = df[df['round_winner'] == 0]['t_armor']

plt.boxplot([ct_armors_when_win, ct_armors_when_lose, t_armors_when_win, t_armors_when_lose],
            labels=['Win (CT)', 'Lose (CT)', 'Win (T)', 'Lose (T)'])
plt.xlabel('Result')
plt.ylabel('Number of Armors')
plt.title('Number of Armors when CT wins and loses')
plt.show()
```

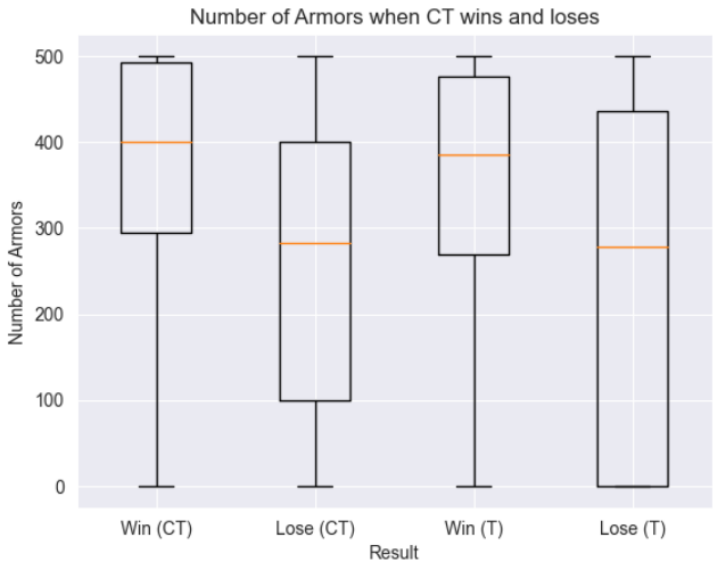

Рисунок 3.12

Як можемо бачити на першій діаграмі набір для знешкодження вибухівки є важливим фактором виграшу для команди, що грає за сторону CT. На другій діаграмі видно, що кількість броні є важливим фактором для перемоги як сторони CT так і сторони T.

Побудуємо таблицю кореляції для наших характеристик для того, щоб побачити позитивну, негативну залежність або її відсутність. Оскільки наш датасет містить дуже велику кількість колонок, то помістити їх усі на тепловій карті дуже складно, отже знайдемо топ найвпливовіших, на прогнозуюче значення, характеристик (рис. 3.13) та виведемо її на екран (рис. 3.14).

```
correlation_matrix = df.corr()
top_correlation_cols = correlation_matrix[abs(correlation_matrix['round_winner']) > 0.20].index

plt.figure(figsize=(15, 10))
sns.heatmap(correlation_matrix.loc[top_correlation_cols, top_correlation_cols], annot=True)

plt.title(f'Top Correlation')
plt.show()
```

Рисунок 3.13

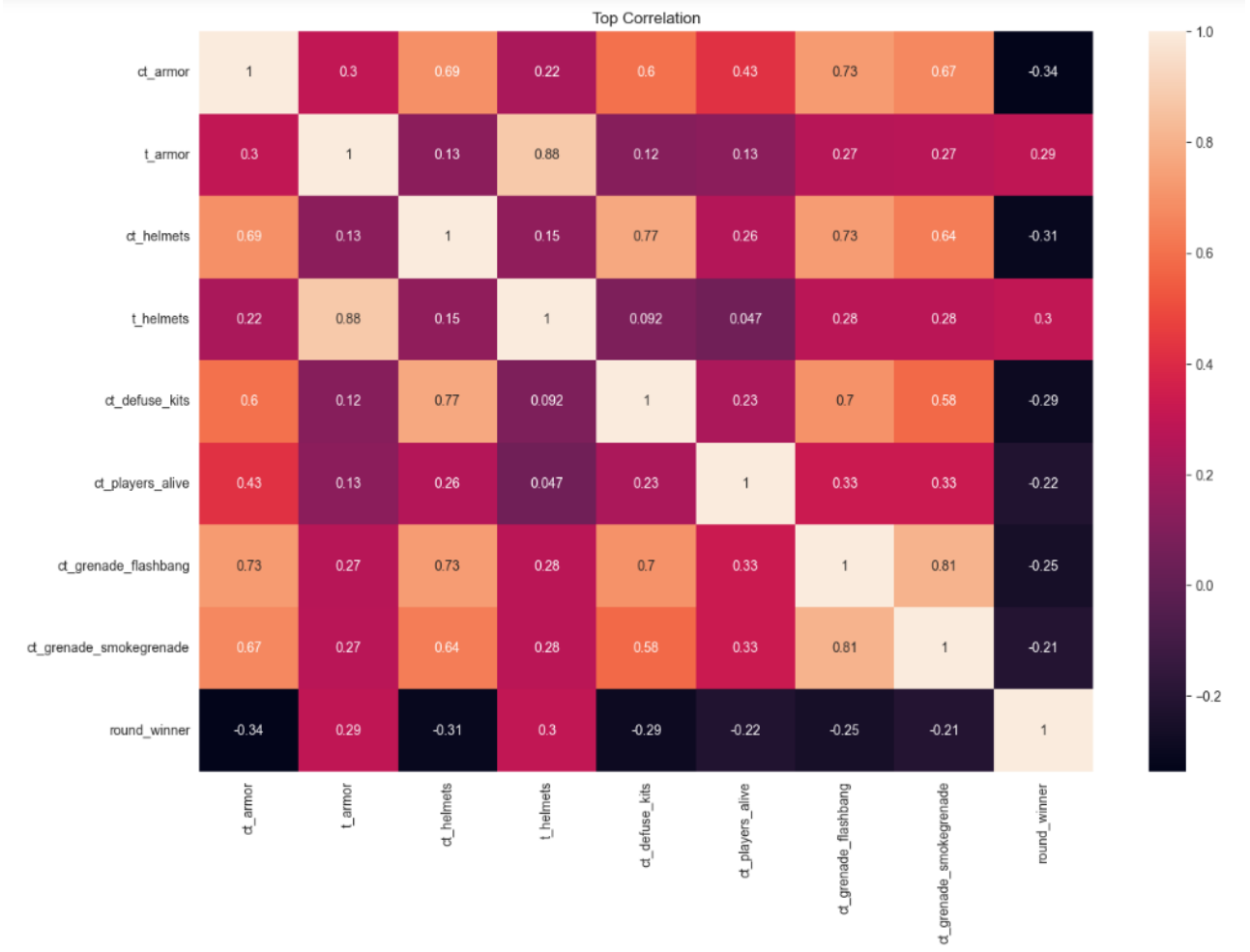

Рисунок 3.14

На даній матриці кореляції можемо побачити явну залежність між кількістю багатьох видів зброї, це досить логічно, оскільки Counter-Strike - це гра, у якій існує багато тактик та командних заготовок, відповідно існують ситуації, коли команди купують одну і ту саму зброю протягом багатьох раундів. Наприклад, візьмемо дві характеристики t_armor і t_helmets, кореляція між якими становить 0.88, така кореляція цілком логічна, оскільки стрілецька зброя, яку зазвичай купує при “повному закупі” команда, що грає за сторону CT не вбиває з одного попадання у голову гравця команди T при умові, що гравець команди T має шолом, отже цілком очевидно, що гравці команди T мають більше шансів вижити, якщо придбають і броню і шолом. Кореляція між цими параметрами для команди сторони CT трішки менше і становить 0.69, це пов’язано з тим, що зброя, яку зазвичай купує при “повному закупі” команда, що грає за сторону T вбиває з одного попадання у голову гравця команди CT, навіть якщо той придбав шолом, отже якщо гравець команди CT знає, що у команди T “повний закуп”, то розумніше буде не купувати шолом тим самим заощадити кошти для наступного раунду.

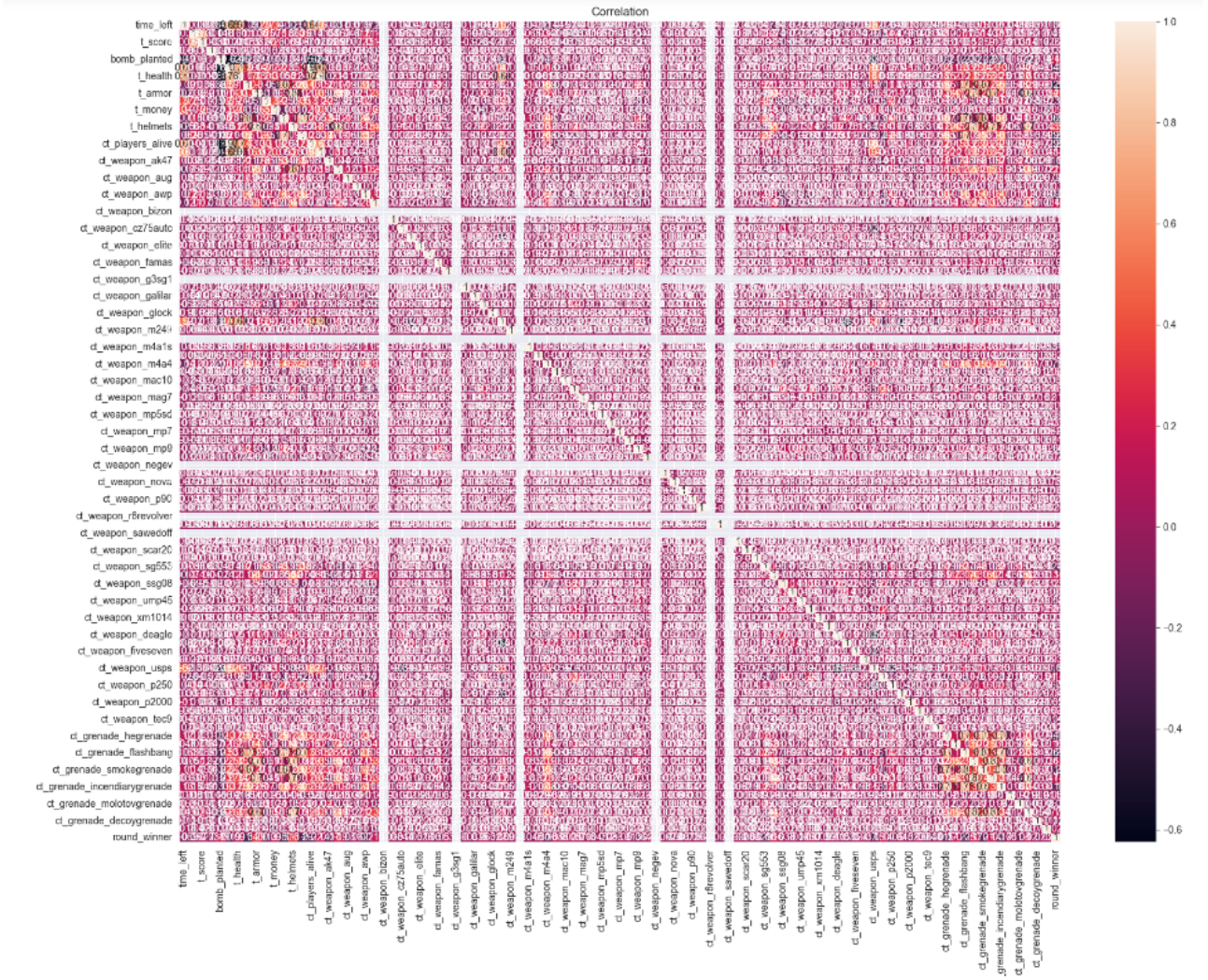

Рисунок 3.15 - Матриця кореляції для всіх характеристик

3.3 Розподіл даних

За допомогою бібліотеки Sklearn[4] розділимо дані на тренувальні та тестові у відношенні 70% на 30% відповідно. Тренувальні дані ми будемо використовувати для того, щоб тренувати моделі, а тестові дані для прогнозування та верифікації результатів (рис. 3.16).

```
X = df.drop('round_winner', axis=1)
y = df['round_winner']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10)
```

Рисунок 3.16

4. ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

4.1 Обґрунтування вибору методів інтелектуального аналізу

У курсовій роботі для визначення переможців у раунді у комп’ютерній грі Counter-Strike Global Offensive було обрано методи класифікації об’єктів: Random Forest, Decision Tree та K-Nearest Neighbors. Спираючись на попередні зіграні раунди за допомогою математичних обчислень можна визначити переможця у наступному раунді, що є актуальним для різних букмекерських контор, або для створення різних інтерактивних внутрішньоігрових предикторів.

Random Forest - це алгоритм машинного навчання, який використовується для задач класифікації та регресії. Він поєднує кілька дерев рішень в одну модель і робить прогноз на основі голосування або середнього значення відповіді дерев. Вибір методу Random Forest був зроблений із розрахунку на те, що даний алгоритм ефективно працює із невеликою кількістю класів і великою кількістю характеристик та знаходить найбільш інформативні з них для класифікації, що відповідає нашому випадку, оскільки у нас є багато характеристик в наборі даних, таких як кількість гравців, їхній стан здоров'я, наявність зброї тощо, а також лише два класи CT та T.

Decision Tree - це алгоритм машинного навчання, який використовується для вирішення завдань класифікації та регресії. Він побудований у вигляді дерева, де кожен внутрішній вузол представляє розбиття за певною ознакою, а кожне листовий вузол відповідає класифікації або прогнозу. Вибір методу Decision Tree був зроблений із розрахунку, що він є простим і інтерпретованим алгоритмом, який може допомогти зрозуміти, які ознаки є найбільш важливими для класифікації. У нашому випадку такими ознаками є використання різного спорядження, факт про те, чи встановлено бомбу, кількість здоров’я та броні у гравців тощо. Дерево рішень може допомогти виявити ключові фактори, які впливають на переможця раунду.

Алгоритм K-Nearest Neighbors - це алгоритм, який використовується для класифікації та регресії. Він базується на принципі "близькі сусіди", де класифікація або прогноз залежить від класів (у випадку класифікації) або

значень (у випадку регресії) найближчих сусідів. Основна ідея алгоритму K-Nearest Neighbors полягає у знаходженні K найближчих сусідів для кожного нового прикладу, який потребує класифікації або прогнозу. Вибір методу K-Nearest Neighbors був зроблений із розрахунку, що даний алгоритм є простим алгоритмом, який використовує найближчих сусідів для класифікації нових прикладів. У нашому випадку, можна сподіватися, що стиль гри і тактика команди, що програє або перемагає, можуть мати подібні риси та характеристики. Використання k-Найближчих сусідів може допомогти знайти схожі групи прикладів і виробити прогнози на основі їхнього класу.

4.2 Аналіз отриманих результатів для методу Random Forest

Перед початком прогнозування потрібно знайти найкращу модель, для цього підберемо найкращий параметр для нашої моделі, у випадку алгоритму Random Forest цим параметром є кількість дерев рішень, які працюють під капотом. Після того, як найкращу модель буде знайдено спрогнозуємо наше значення та оцінимо отриманий результат (рис 4.1).

```
rf_classifier = RandomForestClassifier()
rf_grid_search = GridSearchCV(rf_classifier, param_grid={'n_estimators': [10, 50, 100, 200, 500]}, cv=5, scoring='accuracy')
rf_grid_search.fit(X_train, y_train)

rf_model = rf_grid_search.best_estimator_
Y_pred = rf_model.predict(X_test)

print("Random Forest Classifier")
print('MSE: %.2f' % mean_squared_error(y_test, Y_pred))
print('R2 score: %.2f' % r2_score(y_test, Y_pred))
print('Accuracy score: %.2f' % accuracy_score(y_test, Y_pred))
```

```
Random Forest Classifier
MSE: 0.12
R2 score: 0.53
Accuracy score: 0.88
```

Рисунок 4.1

Як можемо бачити значення MSE (mean squared error) досить низьке, в той час як Accuracy score достатньо високий, що свідчить про достатню точність побудованої моделі.

Напишемо функцію для побудови матриці невідповідностей на основі спрогнозованих значень (рис. 4.2).

```
def show_confusion_matrix(prediction, name):
    columns = ['CT', 'T']
    confusion = confusion_matrix(y_test, prediction)
    confusion_df = pd.DataFrame(confusion, index=columns, columns=columns)
    sns.heatmap(confusion_df, annot=True, fmt='d')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'Confusion Matrix for {name}')
    plt.show()
```

Рисунок 4.2

Для наочної оцінки точності спрогнозованих результатів побудуємо матрицю невідповідностей для алгоритму Random Forest (рис. 4.3).

```
show_confusion_matrix(Y_pred, 'Random Forest Classifier')
```

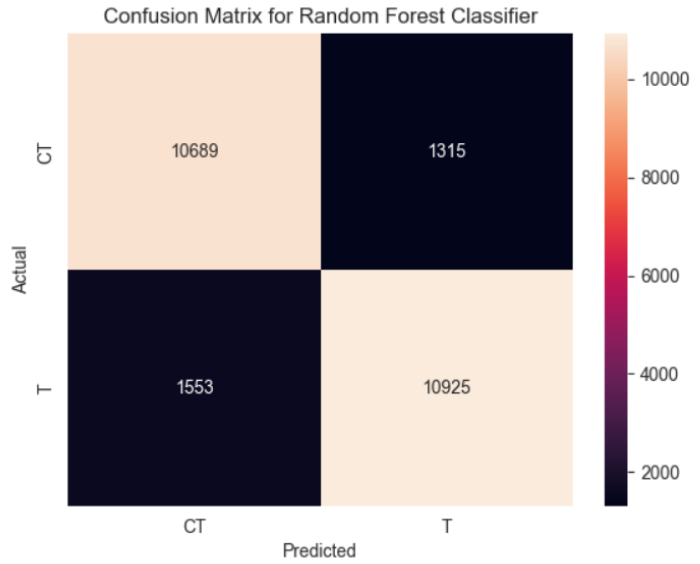

Рисунок 4.3

Матриця невідповідностей для двох класів має 4 клітинки, де (CT, CT) - комірка, яка містить кількість істинних прогнозів ситуацій, коли перемагає команда, яка грає за CT. (T, CT) - комірка, яка містить кількість хибних прогнозів ситуацій, коли перемагає команда, яка грає за CT. (CT, T) -комірка, яка містить кількість хибних прогнозів ситуацій, коли перемагає команда, яка грає за T. (T, CT) -комірка, яка містить кількість істинних прогнозів ситуацій, коли перемагає команда, яка грає за T.

Оскільки алгоритм Random Forest має властивість виділяти характеристики, які більш за все впливають на прогноз, то напишемо функцію для побудови графіка важливостей характеристик (рис. 4.4).

```
importance = rf_model.feature_importances_

def show_feature_importance(imp):
    sorted_indices = np.argsort(imp)[::-1]
    sorted_importance = importance[sorted_indices]
    nonzero_indices = np.nonzero(sorted_importance)
    sorted_importance = sorted_importance[nonzero_indices]
    sorted_features = df.columns[:-1][sorted_indices][nonzero_indices]
    plt.figure(figsize=(10, 8))
    plt.bar(range(len(sorted_features)), sorted_importance, tick_label=sorted_features)
    plt.xlabel('Features')
    plt.ylabel('Importance')
    plt.title('Feature Importances')
    plt.xticks(rotation=70)
    plt.show()

show_feature_importance(importance)
```

Рисунок 4.4

Для наочної оцінки важливості певних характеристик побудуємо графік важливостей характеристик (рис. 4.5).

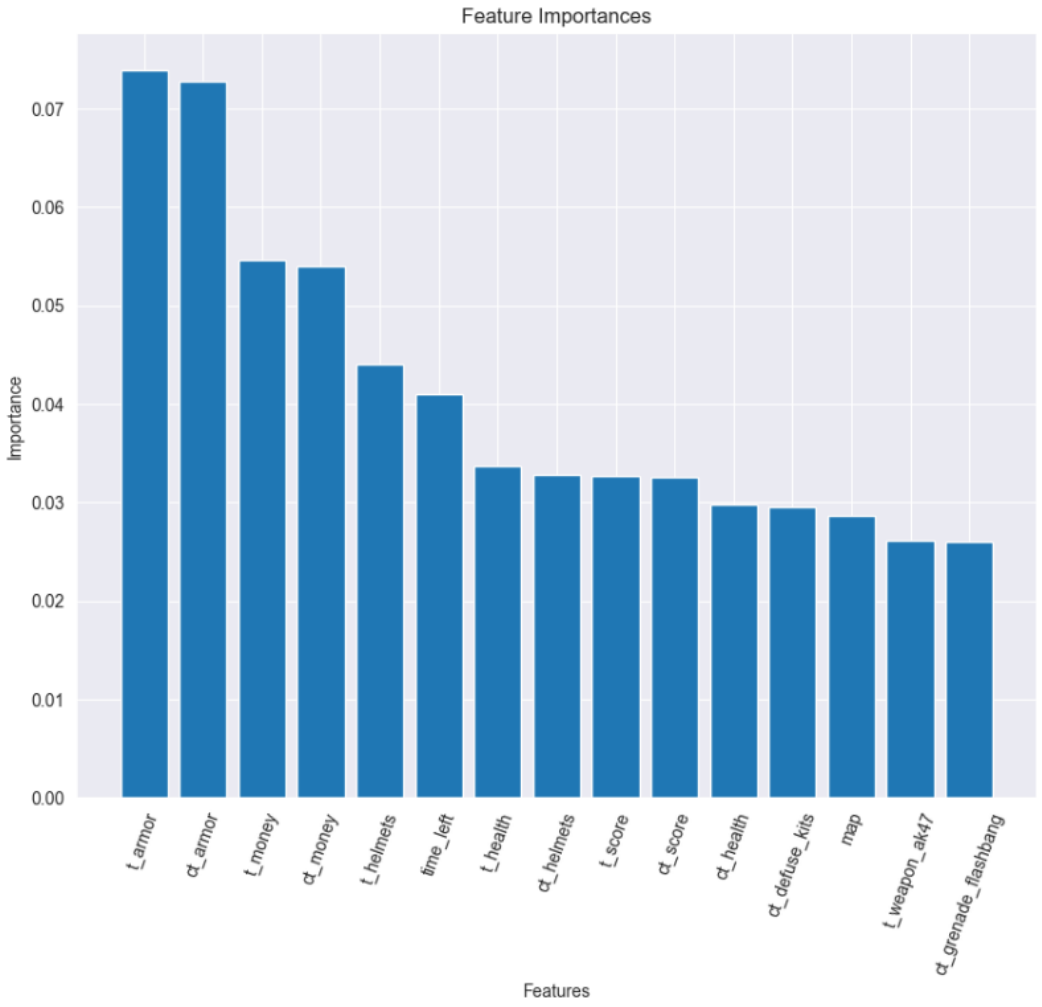

Рисунок 4.5

На цьому графіку можемо бачити, що найбільш важливими рисами як для однієї так і для іншої сторони (CT та T) є наявність броні, що в принципі логічно, оскільки вона значно збільшує ймовірність того, що гравець виграє дуель, а це у свою чергу дає команді перевагу у вигляді однієї одиниці зброї.

Також досить впливовими факторами є економіка команди, що також є доволі логічно, оскільки наявність міцної економіки дає можливість кожного раунду купувати найефективнішу стрілецьку зброю, броню, шоломи а також дуже важливі гранати. На 5 місці по важливості стоїть характеристика t_helmets (кількість шоломів у гравців команди T), раніше, коли ми переглядали матрицю кореляції, ми зробили висновок, що шоломи для команди T є більш важливими ніж для команди CT і тепер ми можемо побачити, що дійсно наявність шоломів для команди T є важливим фактором для перемоги.

4.3 Аналіз отриманих результатів для методу Decision Tree

Аналогічно до пункту 4.2 знайдемо найкращу модель та проведемо прогноз, але цього разу задамо параметр, який впливає на глибину дерева (рис. 4.6).

```
dt_classifier = DecisionTreeClassifier()
dt_grid_search = GridSearchCV(dt_classifier, {'max_depth': [i for i in range(40, 50)]}, cv=5, scoring='accuracy')
dt_grid_search.fit(X_train, y_train)

dt_model = dt_grid_search.best_estimator_
Y_pred = dt_model.predict(X_test)

print("Decision Tree Classifier")
print('MSE: %.2f' % mean_squared_error(y_test, Y_pred))
print('R2 score: %.2f' % r2_score(y_test, Y_pred))
print('Accuracy score: %.2f' % accuracy_score(y_test, Y_pred))
```

```
Decision Tree Classifier
MSE: 0.18
R2 score: 0.27
Accuracy score: 0.82
```

Рисунок 4.6

Як можемо бачити значення MSE (mean squared error) досить низьке, в той час як Accuracy score достатньо високий, що свідчить про достатню точність побудованої моделі.

Для наочної оцінки точності спрогнозованих результатів побудуємо матрицю невідповідностей для алгоритму Decision Tree (рис. 4.7).

```
show_confusion_matrix(Y_pred, 'Decision Tree Classifier')
```

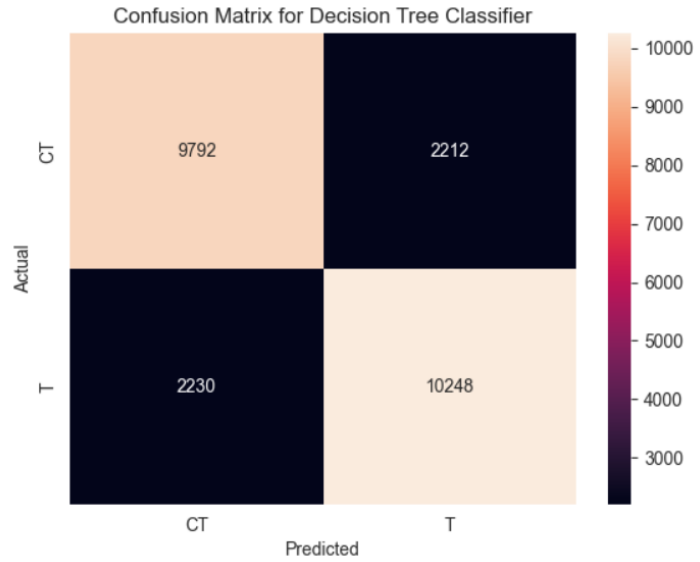

Рисунок 4.7

Аналогічно у матриці невідповідностей можемо побачити кількість істинних та хибних прогнозів для обох сторін.

Для наочної оцінки важливості певних характеристик побудуємо графік важливостей характеристик (рис. 4.8).

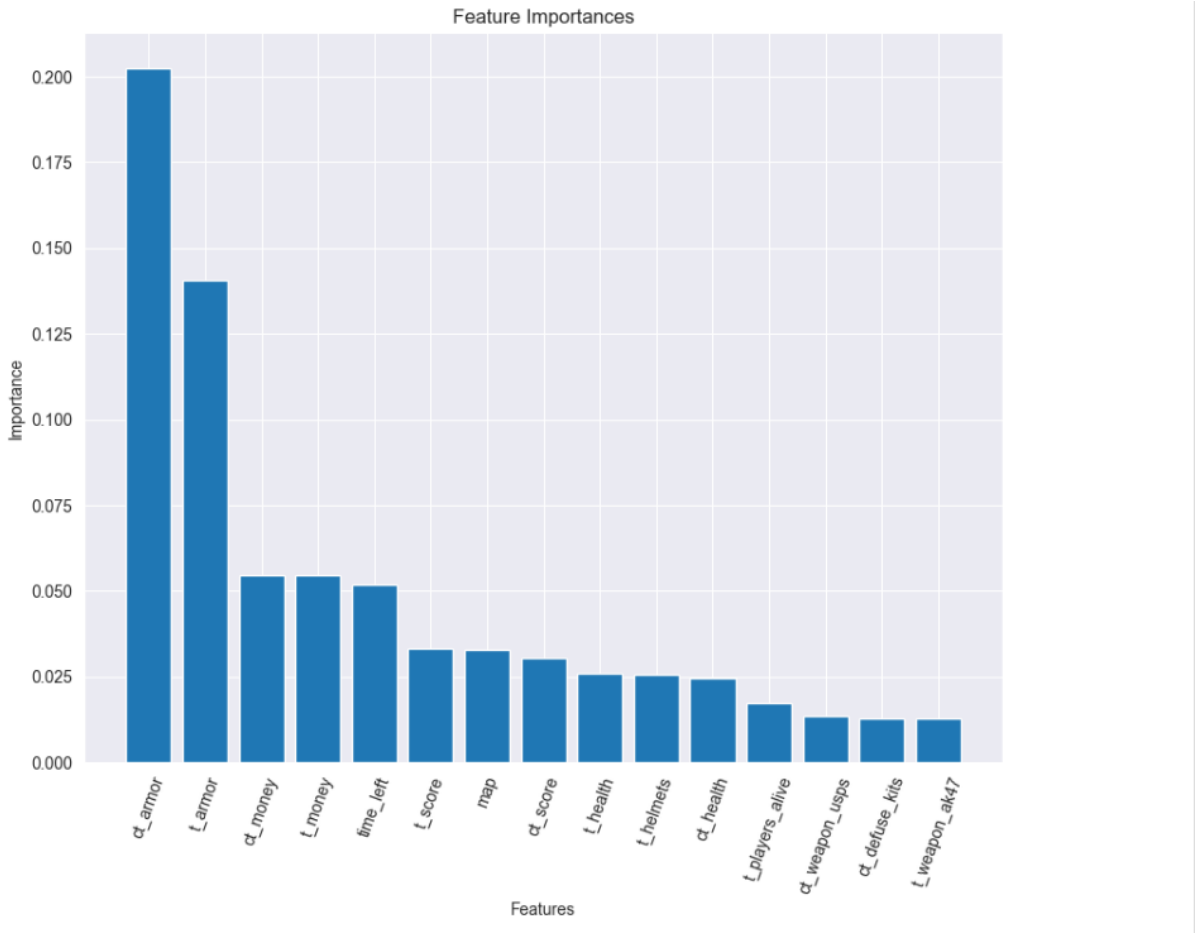

Рисунок 4.8

На даній стовпчастій діаграмі можемо бачити практично аналогічну ситуацію що і в Random Forest моделі, це не дивно, оскільки під капотом у Random Forest працює цілий ліс Decision Tree.

Для кращого розуміння як працює Decision Tree виведемо дерево на екран (рис. 4.9)

```
plt.figure(figsize=(10, 8), dpi=750)
plot_tree(dt_model, feature_names=df.columns[:-1], class_names=count.index, filled=True, max_depth=2)
plt.show()
```

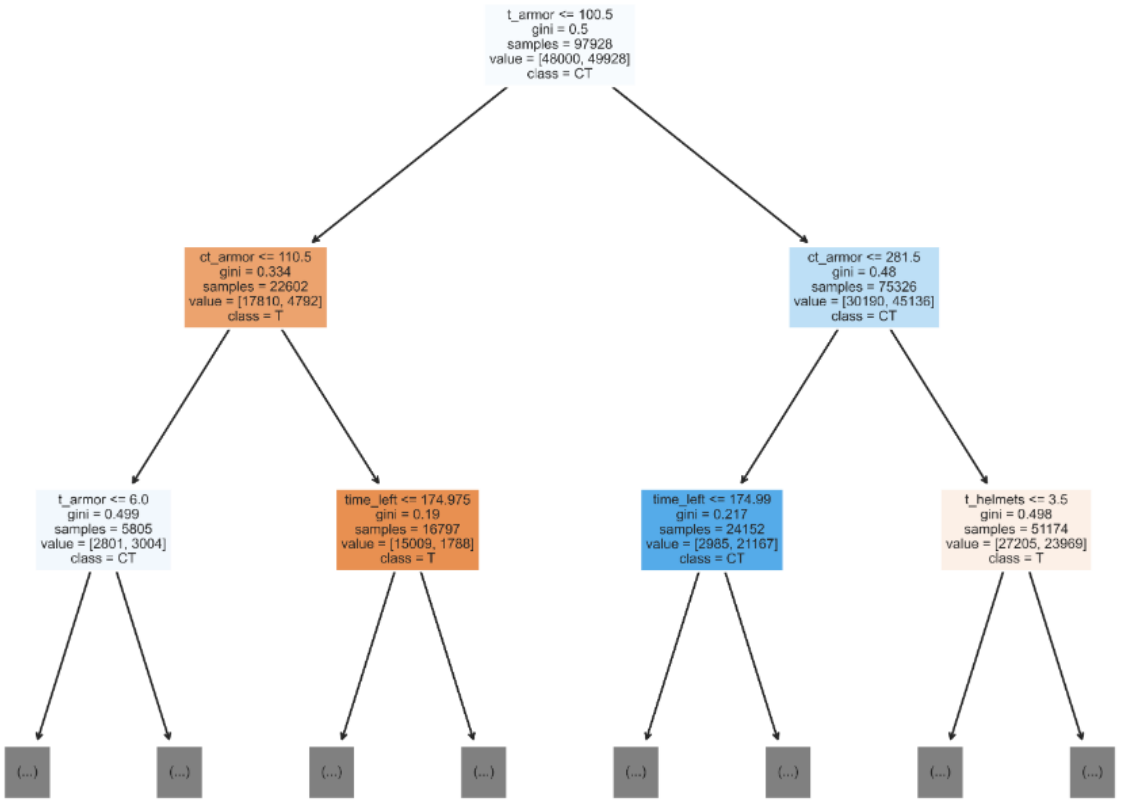

Рисунок 4.9

4.4 Аналіз отриманих результатів для методу K-Nearest Neighbors

Аналогічно до пунктів 4.2 та 4.5 знайдемо найкращу модель та проведемо прогноз, але цього разу задамо параметр, на кількість сусідів (рис. 4.10).

```
kn_model_classifier = KNeighborsClassifier()
kn_model_grid_search = GridSearchCV(kn_model_classifier, {'n_neighbors': [1, 5, 10, 15]}, cv=5, scoring='accuracy')
kn_model_grid_search.fit(X_train, y_train)

kn_model = kn_model_grid_search.best_estimator_
Y_pred = kn_model.predict(X_test)

print("K Neighbors Classifier")
print('MSE: %.2f' % mean_squared_error(y_test, Y_pred))
print('R2 score: %.2f' % r2_score(y_test, Y_pred))
print('Accuracy score: %.2f' % accuracy_score(y_test, Y_pred))
```

```
K Neighbors Classifier
MSE: 0.18
R2 score: 0.26
Accuracy score: 0.82
```

Рисунок 4.10

Як можемо бачити значення MSE (mean squared error) досить низьке, в той час як Accuracy score достатньо високий, що свідчить про достатню точність побудованої моделі.

Для наочної оцінки точності спрогнозованих результатів побудуємо матрицю невідповідностей для алгоритму K-Nearest Neighbors (рис. 4.11).

```
show_confusion_matrix(Y_pred, 'K Neighbors Classifier')
```

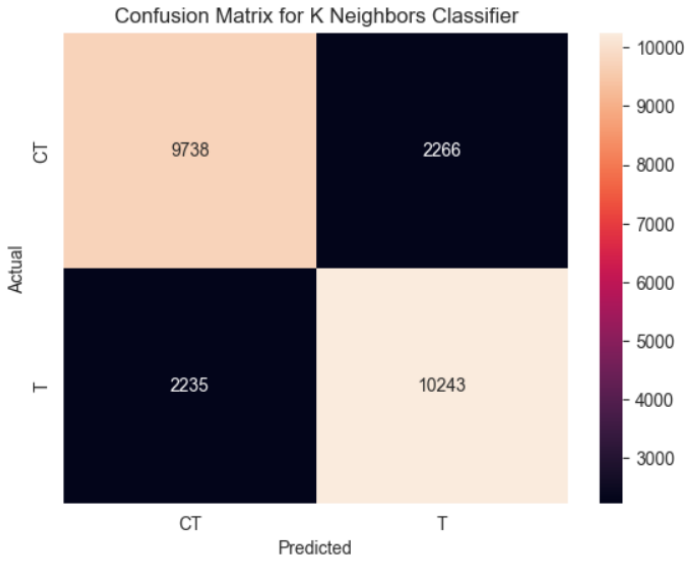

Рисунок 4.11

Аналогічно у матриці невідповідностей можемо побачити кількість істинних та хибних прогнозів для обох сторін.

4.5 Порівняння отриманих результатів

Після побудови моделей та прогнозування результатів а також оцінки цих результатів перейдемо до їх порівняння. Порівнювати будемо за показниками MSE та Accuracy. Чим менше MSE тим менше похибок містить результат, Чим більше Accuracy тим більш точна модель (таб. 4.1).

Таблиця 4.1

score/model	Random Forest	Decision Tree	K-Nearest Neighbors
MSE	0.12	0.18	0.18
R2	0.53	0.28	0.26
Accuracy	0.88	0.82	0.82

Отже, найменша похибка (MSE) та найбільше значення Accuracy у моделі RandomForest. Для більш ґрунтовного порівняння проаналізуємо матриці невідповідностей.

Обчислимо відношення кількості помилкових прогнозів до загальної кількості прогнозів та позначимо це літерою G, чим менше G тим краще модель спрогнозувала результат (табл 4.2).

score/model	Random Forest	Decision Tree	K-Nearest Neighbors
G	0.116494	0.180582	0.183849

Отже, як можемо бачити за результатами матриці невідповідностей найкраще спрогнозувала значення модель Random Forest.

Базуючись на цих двох порівняннях можемо дійти висновку, що найкраща модель для прогнозування переможця у раунді у грі Counter-Strike: Global Offensive це Random Forest.

ВИСНОВОК

У ході виконання курсової роботи було декомпозовано поставлену задачу на такі етапи: проаналізувати предметну область, провести попередню роботу з даними, провести інтелектуальний аналіз даних, побудувати та проаналізувати моделі для прогнозування переможця у раунді на основі показників стану ігрового поля у грі Counter-Strike: Global Offensive та порівняти отримані результати. Для побудови моделей було використано такі методи: Random Forest, Decision Tree та K-Nearest Neighbors. Було використано такі технології: Python[1], Pandas[2], Matplotlib[3], Sklearn[4], Seaborn[5], NumPy[6]. У ході проведення інтелектуального аналізу було побудовано три моделі, що дають змогу прогнозувати переможця раунду базуючись на інформацію про стан ігрового поля, кожну модель було оцінено за допомогою MSE, R2 score та Accuracy score, до кожної з моделей було побудовано матрицю невідповідностей, а також було побудовано стовпчасту діаграму важливості характеристик для Random Forest та Decision Tree. Отримані дані дали нам змогу провести порівняння побудованих моделей за характеристиками MSE, Accuracy score та оцінки G, за допомогою яких ми визначили найкращу модель Random Forest.

Отже, було проаналізовано предметну область, проведено попередню роботу з даними, проведено інтелектуальний аналіз даних, побудовано та проаналізовано моделі для прогнозування переможця у раунді на основі показників стану ігрового поля у грі Counter-Strike, також було визначено найкращу модель Random Forest за характеристиками MSE, Accuracy score та оцінкою G.

ПЕРЕЛІК ЛІТЕРАТУРИ

1. Python. [Електронний ресурс] - URL: https://www.python.org/
2. Pandas. [Електронний ресурс] - URL: https://pandas.pydata.org/docs/
3. Matplotlib. [Електронний ресурс] - URL: https://matplotlib.org/stable/
4. Sklearn. [Електронний ресурс] - URL: https://devdocs.io/scikit_learn/
5. Seaborn. [Електронний ресурс] - URL: https://seaborn.pydata.org/
6. NumPy. [Електронний ресурс] - URL: https://numpy.org/

ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ

Тексти програмного коду Прогнозування переможця у раунді на основі показників стану ігрового поля у грі Counter-Strike: Global Offensive методами: Random Forest, Decision Tree та K-Nearest Neighbors

(Найменування програми (документа))

SSD

(Вид носія даних)

32 арк, 1.23 Мб

(Обсяг програми (документа), арк., Кб)

студента групи ІП-13 ІІ курсу

Паламарчука О.О.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix
df = pd.read_csv('resources/csgo_round_snapshots.csv')
df.info()
df['map'].unique()
label_encoder = preprocessing.LabelEncoder()
df['map'] = label_encoder.fit_transform(df['map'])
df['map'].unique()
df['bomb_planted'].unique()
df['bomb_planted'] = df['bomb_planted'].astype(int)
df['bomb_planted'].unique()
df['round_winner'].unique()
df['round_winner'] = label_encoder.fit_transform(df['round_winner'])
df['round_winner'].unique()
count = df['round_winner'].value_counts()
count.index = count.index.map({0: 'CT', 1: 'T'})
plt.pie(count, labels=count.index, autopct='%1.1f%%')
plt.title('Count CT and T wins')
plt.show()
ct_kits_when_win = df[df['round_winner'] == 0]['ct_defuse_kits']
ct_kits_when_lose = df[df['round_winner'] == 1]['ct_defuse_kits']

plt.boxplot([ct_kits_when_win, ct_kits_when_lose], labels=['Win', 'Lose'])
plt.xlabel('Result')
plt.ylabel('Number of Kits')
plt.title('Number of Kits when CT wins and loses')
plt.show()
ct_armors_when_win = df[df['round_winner'] == 0]['ct_armor']
ct_armors_when_lose = df[df['round_winner'] == 1]['ct_armor']
t_armors_when_win = df[df['round_winner'] == 1]['t_armor']
```

```
t_armors_when_lose = df[df['round_winner'] == 0]['t_armor']

plt.boxplot([ct_armors_when_win, ct_armors_when_lose, t_armors_when_win,
t_armors_when_lose],
            labels=['Win (CT)', 'Lose (CT)', 'Win (T)', 'Lose (T)'])
plt.xlabel('Result')
plt.ylabel('Number of Armors')
plt.title('Number of Armors when CT wins and loses')
plt.show()
df.sample(5)
correlation_matrix = df.corr()
top_correlation_cols = correlation_matrix[abs(correlation_matrix['round_winner']) >
0.20].index

plt.figure(figsize=(15, 10))
sns.heatmap(correlation_matrix.loc[top_correlation_cols, top_correlation_cols],
annot=True)
plt.title(f'Top Correlation')
plt.show()
correlation_matrix = df.corr()
plt.figure(figsize=(20, 15))
sns.heatmap(correlation_matrix, annot=True)
plt.title('Correlation')
plt.show()
X = df.drop('round_winner', axis=1)
y = df['round_winner']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
def show_confusion_matrix(prediction, name):
    columns = ['CT', 'T']
    confusion = confusion_matrix(y_test, prediction)
    confusion_df = pd.DataFrame(confusion, index=columns, columns=columns)
    sns.heatmap(confusion_df, annot=True, fmt='d')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title(f'Confusion Matrix for {name}')
    plt.show()
rf_classifier = RandomForestClassifier()
rf_grid_search = GridSearchCV(rf_classifier, param_grid={'n_estimators': [10, 50,
100, 200, 500]}, cv=5, scoring='accuracy')
```

```
rf_grid_search.fit(X_train, y_train)

rf_model = rf_grid_search.best_estimator_
Y_pred = rf_model.predict(X_test)

print("Random Forest Classifier")
print('MSE: %.2f' % mean_squared_error(y_test, Y_pred))
print('R2 score: %.2f' % r2_score(y_test, Y_pred))
print('Accuracy score: %.2f' % accuracy_score(y_test, Y_pred))
show_confusion_matrix(Y_pred, 'Random Forest Classifier')
importance = rf_model.feature_importances_

def show_feature_importance(imp):
    sorted_indices = np.argsort(imp)[::-1]
    sorted_importance = importance[sorted_indices]
    nonzero_indices = np.nonzero(sorted_importance)
    sorted_importance = sorted_importance[nonzero_indices][:15]
    sorted_features = df.columns[:-1][sorted_indices][nonzero_indices][:15]
    plt.figure(figsize=(10, 8))
    plt.bar(range(len(sorted_features)), sorted_importance, tick_label=sorted_features)
    plt.xlabel('Features')
    plt.ylabel('Importance')
    plt.title('Feature Importances')
    plt.xticks(rotation=70)
    plt.show()

show_feature_importance(importance)
dt_classifier = DecisionTreeClassifier()
dt_grid_search = GridSearchCV(dt_classifier, {'max_depth': [i for i in range(40,
50)]}, cv=5, scoring='accuracy')
dt_grid_search.fit(X_train, y_train)

dt_model = dt_grid_search.best_estimator_
Y_pred = dt_model.predict(X_test)

print("Decision Tree Classifier")
print('MSE: %.2f' % mean_squared_error(y_test, Y_pred))
print('R2 score: %.2f' % r2_score(y_test, Y_pred))
print('Accuracy score: %.2f' % accuracy_score(y_test, Y_pred))
show_confusion_matrix(Y_pred, 'Decision Tree Classifier')
```

```
importance = dt_model.feature_importances_
show_feature_importance(importance)
plt.figure(figsize=(10, 8), dpi=750)
plot_tree(dt_model, feature_names=df.columns[:-1], class_names=count.index,
filled=True, max_depth=2)
plt.show()
kn_model_classifier = KNeighborsClassifier()
kn_model_grid_search = GridSearchCV(kn_model_classifier, {'n_neighbors': [1, 5,
10, 15]}, cv=5, scoring='accuracy')
kn_model_grid_search.fit(X_train, y_train)
kn_model = kn_model_grid_search.best_estimator_
Y_pred = kn_model.predict(X_test)
print("K Neighbors Classifier")
print('MSE: %.2f' % mean_squared_error(y_test, Y_pred))
print('R2 score: %.2f' % r2_score(y_test, Y_pred))
print('Accuracy score: %.2f' % accuracy_score(y_test, Y_pred))
show_confusion_matrix(Y_pred, 'K Neighbors Classifier')
```