STATISTICS PROJECT

ON

EXPLORATORY ANALYSIS ON Counter Strike: Global Offensive Tournament Dataset

SUMITTED TO-

Praxis Business School

SUBMITTED BY-
Aditya Pathania-A21002
Ashutosh Sharma -A21006

Exploratory Data Analysis and Descriptive Statistics of Counter Strike Tournament Dataset.
STS – I Project.

Submitted by –

1. Aditya Pathania (A21002)
2. Ashutosh Sharma (A21006)

<u>Abstract</u> –

This project explores the nature of multiplayer first-person shooter video games, which have been very popular. We have chosen Counter-Strike: Global Offensive as the benchmark in this study. Many data from official Counter-Strike: Global Offensive tournaments and public match are collected for the analysis, whereas game refinement measure is employed for the assessment with a focus on two aspects: gameplay and rounds system. The results show that the gameplay aspect has lower game refinement, i.e., the game is highly skilful, whereas the rounds system aspect has higher game refinement, i.e., the game is highly stochastic. The combination of the two aspects successfully balances the skilfulness and chance, which results in their popularity. In addition, since their release, Counter-Strike: Global Offensive has sold more than 40 million game copies, awarded as “Best eSports Game of The Year” and updated maps and successfully maintain the game balance of the maps, as measured by game refinement.

Exploratory Data Analysis and Descriptive Statistics of Counter-Strike: Global Offensive Tournament.

Introduction:

Counter-Strike: Global Offensive is a multiplayer first-person shooter developed by Valve and Hidden Path Entertainment. It is the fourth game in the Counter-Strike series. Developed for over two years, Global Offensive was released for Windows, macOS, Xbox 360, and PlayStation 3 in August 2012, and for Linux in 2014. Valve still regularly updates the game, both with smaller balancing patches and larger content additions.

The game pits two teams, Terrorists and Counter-Terrorists, against each other in different objective-based game modes. The most common game modes involve the Terrorists planting a bomb while Counter-Terrorists attempt to stop them, or Counter-Terrorists attempting to rescue hostages that the Terrorists have captured. There are nine official game modes, all of which have distinct characteristics specific to that mode. The game also has matchmaking support that allows players to play on dedicated Valve servers, in addition to community-hosted servers with custom maps and game modes. A battle-royal game-mode, "Danger Zone", was introduced in December 2018.

Global Offensive received positive reviews from critics on release, who praised the game for its gameplay and faithfulness to the Counter-Strike series, though it was criticized for some early features and the differences between the console and PC versions. Since its release, it has drawn in an estimated 11 million players per month, and remains one of the most played games on Valve's Steam platform. In December 2018, Valve transitioned the game to a free-to-play model, focusing on revenue from cosmetic items.

The game has an active esports scene, continuing the history of international competitive play from previous games in the series. Teams compete in professional leagues and tournaments, and Global Offensive is now one of the largest global esports.

Global Offensive has one of the most popular esport scenes in the world. The Global Offensive professional scene consists of leagues and tournaments hosted by third-party organisations, and Valve-sponsored tournaments known as Major Championships. Majors are considered the most prestigious tournaments in Counter-Strike circuit and have among the largest prize pools; originally announced at US$250,000, the prize pools for Majors have risen to US$1,000,000 since MLG Columbus 2016. Astralis is the most successful Global Offensive team of all time, with the core members of that team winning four Majors together.

In 2014, the "first large match fixing scandal in the Global Offensive community took place, where team iBuyPower purposefully lost a match against NetCodeGuides.com. The seven professional players that were involved in the scandal were permanently banned from all Majors by Valve, although some other organizers eventually allowed the players to compete at their tournaments.

Publication	Score
Destructoid	PC: 9.5/10
Eurogamer	PC: 9/10
GameSpot	PC/PS3/X360: 8.5/10
GameSpy	PC: ★★★★☆
IGN	PC: 8/10
PC Gamer (US)	PC: 84%

Understanding the Data-

We downloaded the dataset from Hltv.org which is the official scoring and news official website for all the tournament games around the world . For initial works we have the dataset for 3.5 GB , which we have to divide the dataset and analyse the dataset accordingly which is most important as per the analysis by categorising them into various categories like Maps being used and the matches data , we chose to analyse the master data set first to pick up the best teams and the maps , and then analyse the game .

The kill data set has 377629 rows × 10 columns, the master dataset had 955466 rows × 33 columns which had the best map data analysed so we picked that data to check the Economy and the positioning of the players while in the game to analyse how can a team win , and what are the areas of focus for the team and the player for winning the game, we also tried to analyse which spots are the best to hit for getting effective kills before any other player secures the kill.

All the datasets and the analysis has been shared as PY full file below .

Before diving deep into the analysis , we should know the rules and understanding the game is must , so the rules are such –

1. A team consists of 5 players each , including 1 mentor /coach for the team which also is a stand by player
2. There are 30 Rounds in the 1st leg of the game , first team to get to 16 wins the game .
3. There are 2 sides on the game , one is counter-terrorist and other is terrorist .
4. Team will have to play knife round , whoever wins gets the option to play the side for the first 15 rounds respectively .
5. If at the end of 30 Rounds , there is a tie of 15-15 there will be a 6 round played again , with 3 on each side each team.
6. Teams switch side after 15 rounds.
7. A single round is for 1 :45 minutes.
8. The main focus of the game basically based on the terrorist side – who either have to secure the round by eliminating all the 5 players on the counter terrorist side , or by successfully crashing the bomb site.
9. There are 2 sides to plant the bomb A and B respectively , terrorists either successfully crash the site with the Bomb plant or either eliminate all the others before 1 :45 minutes.
10. The main task of the counter terrorist is to defend both the site A and B .
11. The teams have to buy guns , rifles and other equipment's , the only stuff the players get is a 32 round pistol , at the initial round players only get 800 $ only , each kill ,plant and the win has some reward as terms of money with this money the players buy on the next rounds , which is important as you need to play as a team and you need each player in the team with all buy , if there is a lose end the whole team can lose , that is why economy and positioning is important.

There are various modes in the game , such as Casual , Deathmatch , Wingman and competitive –
The main of all these is Competitive , while the others are either used as a practice or to play the game in luxury time . The dataset that we analysed is for the same , as the main scenes all around the world are based on the same competitive sessions .

The reason why we decided to go with this data was , as we have seen that as the Data science is evolving all around the world in all sectors , the gaming world have also started to hire analysts to coach the team accordingly , as seen the same in basketball , cricket , football ,as the technology and the network services are developing in the world since 2011 , there are lot of people around the world that have taken PC gaming as the full time profession , players are earning not less than any other Game ,Even India have started to

emerge at the world stage , as Esports organised the PC gaming tournament at worth 2 Million dollars of prize poll , which had let many players and teams to come up to the big stage. So let's start and analyse the world of Counter-Strike: Global Offensive

Exploratory Data Analysis and Descriptive Statistics of Counter-Strike: Global Offensive Tournament Rounds

```
Author: Ali Ibrahim Sonay
Date: 2021-11-17
Email: aisonay@gmail.com
Linkedin: https://www.linkedin.com/in/ali-ibrahim-sonay/
Kaggle: https://www.kaggle.com/aliibrahimsonay
```

IMPORT THE VARIOUS DATASET FOR Counter-Strike: Global Offensive

```
!pip install -q -U kaggle
!pip install --exists-action i -q "git+https://github.com/Kaggle/kagglehub.git"
!pip install -q -U kagglehub[hf-datasets]
!pip install -q -U datasets[vision,nlp]
```

```
mkdir: cannot create directory '/root/.kaggle': File exists
```

Kaggle Hub is a library that provides a simple way to download datasets, models, notebook outputs, and competition files from Kaggle. It supports caching, environment integration, and offline mode, making it easy to access Kaggle resources in your ML workflows.

```
import kagglehub
from kagglehub import load_dataset, KaggleDatasetAdapter
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import glob
import warnings
warnings.filterwarnings('ignore')
```

```
Downloading from https://kagglehub.com/bundles/kaggle/kagglehub-frontend/assets/vendor.chunk.js
Downloading from https://kagglehub.com/bundles/kaggle/kagglehub-frontend/assets/styles.chunk.js
Downloading from https://kagglehub.com/bundles/kaggle/kagglehub-frontend/assets/common.chunk.js
Downloading from https://kagglehub.com/bundles/kaggle/kagglehub-frontend/assets/notebook-view.chunk.js
Downloading from https://kagglehub.com/bundles/kaggle/kagglehub-frontend/assets/notebook-view.styles.chunk.js
Downloading from https://kagglehub.com/bundles/kaggle/kagglehub-frontend/assets/notebook-view.vendor.chunk.js
```

```
path = kagglehub.dataset_download("mateusz1801/tournament-rounds-counter-strike-global-offensive")
print("Path to dataset files:", path)
```

```
Path to dataset files: /kaggle/input/tournament-rounds-counter-strike-global-offensive
```

```
!ls -lh /kaggle/input/tournament-rounds-counter-strike-global-offensive
```

```
total 1.1M
-rw- 1 root root 1.1M Jun 17 2021 rounds.csv
```

```
df = pd.read_csv('/kaggle/input/tournament-rounds-counter-strike-global-offensive/rounds.csv')
```

```
df.shape
```

```
(15862, 19)
```

```
df.info()
```

```
<class'>
RangeIndex: 15862 entries, 0 to 15861
Data columns (total 19):
 #  Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0  map                    15862 non-null  object
 1  round                  15862 non-null  int
 2  winner                 15862 non-null  object
 3  t_eq_val               15862 non-null  int
 4  ct_eq_val              15862 non-null  int
 5  t_money                15862 non-null  int
 6  ct_money               15862 non-null  int
 7  t_helmets              15862 non-null  int
 8  ct_helmets             15862 non-null  int
 9  t_defuse               15862 non-null  int
 10 ct_defuse              15862 non-null  int
 11 t_players              15862 non-null  int
 12 ct_players             15862 non-null  int
 13 t_weapons              15862 non-null  int
 14 ct_weapons             15862 non-null  int
 15 t_grenades             15862 non-null  int
 16 ct_grenades            15862 non-null  int
 17 t_score                15862 non-null  int
 18 ct_score               15862 non-null  int
dtype: int, object
memory usage: 2.3 MB
```

```
df.head(5)
```

	map	round	winner	t_eq_val	ct_eq_val	t_money	ct_money	t_helmets	ct_helmets	t_defuse	ct_defuse	t_players	ct_players	t_weapons	ct_weapons	t_grenades	ct_grenades	t_score	ct_score
0	de_cache_competitive	1	ct	4000	4000	3900	3900	0	0	0	0	5	5	0	0	0	0	0	0
1	de_cache_competitive	2	ct	4000	4000	3900	3900	0	0	0	0	5	5	0	0	0	0	1	0
2	de_cache_competitive	3	ct	4000	4000	3900	3900	0	0	0	0	5	5	0	0	0	0	2	0
3	de_cache_competitive	4	ct	4000	4000	3900	3900	0	0	0	0	5	5	0	0	0	0	3	0
4	de_cache_competitive	5	ct	4000	4000	3900	3900	0	0	0	0	5	5	0	0	0	0	4	0

```
df.tail(5)
```

	map	round	winner	t_eq_val	ct_eq_val	t_money	ct_money	t_helmets	ct_helmets	t_defuse	ct_defuse	t_players	ct_players	t_weapons	ct_weapons	t_grenades	ct_grenades	t_score	ct_score
15857	de_dust2competitive	26	ct	18000	18000	16000	16000	5	5	0	0	5	5	5	5	0	0	12	13
15858	de_dust2competitive	27	ct	18000	18000	16000	16000	5	5	0	0	5	5	5	5	0	0	13	13
15859	de_dust2competitive	28	ct	18000	18000	16000	16000	5	5	0	0	5	5	5	5	0	0	14	13
15860	de_dust2competitive	29	ct	18000	18000	16000	16000	5	5	0	0	5	5	5	5	0	0	15	13
15861	de_dust2competitive	30	ct	18000	18000	16000	16000	5	5	0	0	5	5	5	5	0	0	16	13

```
df.describe
```

```
<bound_method_DataFrame.describe of                map  round winner  t_eq_val  ct_eq_val  t_money  ct_money  \
0      de_cache_competitive      1     ct      4000       4000     3900      3900
1      de_cache_competitive      2     ct      4000       4000     3900      3900
2      de_cache_competitive      3     ct      4000       4000     3900      3900
3      de_cache_competitive      4     ct      4000       4000     3900      3900
4      de_cache_competitive      5     ct      4000       4000     3900      3900
...                     ...    ...    ...       ...        ...      ...       ...
15857  de_dust2competitive     26     ct     18000      18000    16000     16000
15858  de_dust2competitive     27     ct     18000      18000    16000     16000
15859  de_dust2competitive     28     ct     18000      18000    16000     16000
15860  de_dust2competitive     29     ct     18000      18000    16000     16000
15861  de_dust2competitive     30     ct     18000      18000    16000     16000

       t_helmets  ct_helmets  t_defuse  ct_defuse  t_players  ct_players  \
0              0           0         0          0          5           5
1              0           0         0          0          5           5
2              0           0         0          0          5           5
3              0           0         0          0          5           5
4              0           0         0          0          5           5
...          ...         ...       ...        ...        ...         ...
15857          5           5         0          0          5           5
15858          5           5         0          0          5           5
15859          5           5         0          0          5           5
15860          5           5         0          0          5           5
15861          5           5         0          0          5           5

       t_weapons  ct_weapons  t_grenades  ct_grenades  t_score  ct_score
0              0           0           0            0        0         0
1              0           0           0            0        1         0
2              0           0           0            0        2         0
3              0           0           0            0        3         0
4              0           0           0            0        4         0
...          ...         ...         ...          ...      ...       ...
15857          5           5           0            0       12        13
15858          5           5           0            0       13        13
15859          5           5           0            0       14        13
15860          5           5           0            0       15        13
15861          5           5           0            0       16        13

[15862 rows, 19 columns]>
```

DATA CLEANING

```
df.isnull()
```

	map	round	winner	t_eq_val	ct_eq_val	t_money	ct_money	t_helmets	ct_helmets	t_defuse	ct_defuse	t_players	ct_players	t_weapons	ct_weapons	t_grenades	ct_grenades	t_score	ct_score
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15857	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
15858	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
15859	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
15860	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False
15861	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False	False

```
df.isnull().sum()
```

```
map            0
round          0
winner         0
t_eq_val       0
ct_eq_val      0
t_money        0
ct_money       0
t_helmets      0
ct_helmets     0
t_defuse       0
ct_defuse      0
t_players      0
ct_players     0
t_weapons      0
ct_weapons     0
t_grenades     0
ct_grenades    0
t_score        0
ct_score       0
dtype: int, object
```

```
df.duplicated()
```

```
0        False
1        False
2        False
3        False
4        False
...      ...
15857    False
15858    False
15859    False
15860    False
15861    False
Length: 15862, dtype: int, object
```

```
df.duplicated().sum()
```

```
0
```

```
df.columns
```

```
Index(['map', 'round', 'winner', 't_eq_val', 'ct_eq_val', 't_money',
       'ct_money', 't_helmets', 'ct_helmets', 't_defuse', 'ct_defuse',
       't_players', 'ct_players', 't_weapons', 'ct_weapons', 't_grenades',
       'ct_grenades', 't_score', 'ct_score'],
      dtype: int, object)
```

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

```
array['de_cache_competitive', 'de_dust2competitive',
       'de_mirage competitive', 'de_inferno competitive',
       'de_train competitive', 'de_cobble competitive',
       'de_overpass competitive', 'de_nuke competitive'],
      dtype: object, object)
```

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

```
array['ct', 't'],
      dtype: object, object)
```

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

```
array[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30],
      dtype: int, object)
```

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

```
array[4000, 3900, 3800, 3700, 3600, 3500, 3400, 3300, 3200, 3100, 3000,
       2900, 2800, 2700, 2600, 2500, 2400, 2300, 2200, 2100, 2000, 1900,
       1800, 1700, 1600, 1500, 1400, 1300, 1200, 1100, 1000,  900,  800,
        700,  600,  500,  400,  300,  200,  100,    0],
      dtype: int, object)
```

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

```
array[4000, 3900, 3800, 3700, 3600, 3500, 3400, 3300, 3200, 3100, 3000,
       2900, 2800, 2700, 2600, 2500, 2400, 2300, 2200, 2100, 2000, 1900,
       1800, 1700, 1600, 1500, 1400, 1300, 1200, 1100, 1000,  900,  800,
        700,  600,  500,  400,  300,  200,  100,    0],
      dtype: int, object)
```

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

```
array[3900, 3800, 3700, 3600, 3500, 3400, 3300, 3200, 3100, 3000,
       2900, 2800, 2700, 2600, 2500, 2400, 2300, 2200, 2100, 2000, 1900,
       1800, 1700, 1600, 1500, 1400, 1300, 1200, 1100, 1000,  900,  800,
        700,  600,  500,  400,  300,  200,  100,    0],
      dtype: int, object)
```

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

```
array[3900, 3800, 3700, 3600, 3500, 3400, 3300, 3200, 3100, 3000,
       2900, 2800, 2700, 2600, 2500, 2400, 2300, 2200, 2100, 2000, 1900,
       1800, 1700, 1600, 1500, 1400, 1300, 1200, 1100, 1000,  900,  800,
        700,  600,  500,  400,  300,  200,  100,    0],
      dtype: int, object)
```

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

```
array[0, 1, 2, 3, 4, 5],
      dtype: int, object)
```

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

```
array[0, 1, 2, 3, 4, 5],
      dtype: int, object)
```

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

```
array[0, 1],
      dtype: int, object)
```

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

```
array[0, 1],
      dtype: int, object)
```

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

```
array[5],
      dtype: int, object)
```

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

```
array[5],
      dtype: int, object)
```

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

```
array[0, 1, 2, 3, 4, 5],
      dtype: int, object)
```

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

```
array[0, 1, 2, 3, 4, 5],
      dtype: int, object)
```

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

```
array[0, 1, 2, 3, 4, 5],
      dtype: int, object)
```

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

```
array[0, 1, 2, 3, 4, 5],
      dtype: int, object)
```

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

```
array[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16],
      dtype: int, object)
```

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

```
array[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13],
      dtype: int, object)
```

```
df['winner'].value_counts()
```

```
winner
ct    8966
t     6896
dtype: int, object, dtype: int, object
```

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

```
map
de_cache_competitive      2000
de_dust2competitive       1900
de_mirage competitive     1900
de_inferno competitive    1900
de_train competitive      1900
de_cobble competitive     1900
de_overpass competitive   1900
de_nuke competitive       1900
dtype: int, object, dtype: int, object
```

```
df['round'].value_counts()
```

```
round
1    1000
2    1000
3    1000
4    1000
5    1000
6    1000
7    1000
8    1000
9    1000
10   1000
11   1000
12   1000
13   1000
14   1000
15   1000
16   1000
17   1000
18   1000
19   1000
20   1000
21   1000
22   1000
23   1000
24   1000
25   1000
26   1000
27   1000
28   1000
29   1000
30   1000
dtype: int, object, dtype: int, object
```

```
df['t_eq_val'].value_counts()
```

```
t_eq_val
4000    1000
3900    1000
3800    1000
3700    1000
3600    1000
3500    1000
3400    1000
3300    1000
3200    1000
3100    1000
3000    1000
2900    1000
2800    1000
2700    1000
2600    1000
2500    1000
2400    1000
2300    1000
2200    1000
2100    1000
2000    1000
1900    1000
1800    1000
1700    1000
1600    1000
1500    1000
1400    1000
1300    1000
1200    1000
1100    1000
1000    1000
900     1000
800     1000
700     1000
600     1000
500     1000
400     1000
300     1000
200     1000
100     1000
0       1000
dtype: int, object, dtype: int, object
```

```
df['ct_eq_val'].value_counts()
```

```
ct_eq_val
4000    1000
3900    1000
3800    1000
3700    1000
3600    1000
3500    1000
3400    1000
3300    1000
3200    1000
3100    1000
3000    1000
2900    1000
2800    1000
2700    1000
2600    1000
2500    1000
2400    1000
2300    1000
2200    1000
2100    1000
2000    1000
1900    1000
1800    1000
1700    1000
1600    1000
1500    1000
1400    1000
1300    1000
1200    1000
1100    1000
1000    1000
900     1000
800     1000
700     1000
600     1000
500     1000
400     1000
300     1000
200     1000
100     1000
0       1000
dtype: int, object, dtype: int, object
```

```
df['t_money'].value_counts()
```

```
t_money
3900    1000
3800    1000
3700    1000
3600    1000
3500    1000
3400    1000
3300    1000
3200    1000
3100    1000
3000    1000
2900    1000
2800    1000
2700    1000
2600    1000
2500    1000
2400    1000
2300    1000
2200    1000
2100    1000
2000    1000
1900    1000
1800    1000
1700    1000
1600    1000
1500    1000
1400    1000
1300    1000
1200    1000
1100    1000
1000    1000
900     1000
800     1000
700     1000
600     1000
500     1000
400     1000
300     1000
200     1000
100     1000
0       1000
dtype: int, object, dtype: int, object
```

```
df['ct_money'].value_counts()
```

```
ct_money
3900    1000
3800    1000
3700    1000
3600    1000
3500    1000
3400    1000
3300    1000
3200    1000
3100    1000
3000    1000
2900    1000
2800    1000
2700    1000
2600    1000
2500    1000
2400    1000
2300    1000
2200    1000
2100    1000
2000    1000
1900    1000
1800    1000
1700    1000
1600    1000
1500    1000
1400    1000
1300    1000
1200    1000
1100    1000
1000    1000
900     1000
800     1000
700     1000
600     1000
500     1000
400     1000
300     1000
200     1000
100     1000
0       1000
dtype: int, object, dtype: int, object
```

```
df['t_helmets'].value_counts()
```

```
t_helmets
0    3172
1    3172
2    3172
3    3172
4    3172
5    3172
dtype: int, object, dtype: int, object
```

```
df['ct_helmets'].value_counts()
```

```
ct_helmets
0    3172
1    3172
2    3172
3    3172
4    3172
5    3172
dtype: int, object, dtype: int, object
```

```
df['t_defuse'].value_counts()
```

```
t_defuse
0    8862
1    7000
dtype: int, object, dtype: int, object
```

```
df['ct_defuse'].value_counts()
```

```
ct_defuse
0    8862
1    7000
dtype: int, object, dtype: int, object
```

```
df['t_players'].value_counts()
```

```
t_players
5    15862
dtype: int, object, dtype: int, object
```

```
df['ct_players'].value_counts()
```

```
ct_players
5    15862
dtype: int, object, dtype: int, object
```

```
df['t_weapons'].value_counts()
```

```
t_weapons
0    3172
1    3172
2    3172
3    3172
4    3172
5    3172
dtype: int, object, dtype: int, object
```

```
df['ct_weapons'].value_counts()
```

```
ct_weapons
0    3172
1    3172
2    3172
3    3172
4    3172
5    3172
dtype: int, object, dtype: int, object
```

```
df['t_grenades'].value_counts()
```

```
t_grenades
0    3172
1    3172
2    3172
3    3172
4    3172
5    3172
dtype: int, object, dtype: int, object
```

```
df['ct_grenades'].value_counts()
```

```
ct_grenades
0    3172
1    3172
2    3172
3    3172
4    3172
5    3172
dtype: int, object, dtype: int, object
```

```
df['t_score'].value_counts()
```

```
t_score
0    1000
1    1000
2    1000
3    1000
4    1000
5    1000
6    1000
7    1000
8    1000
9    1000
10   1000
11   1000
12   1000
13   1000
14   1000
15   1000
16   1000
dtype: int, object, dtype: int, object
```

```
df['ct_score'].value_counts()
```

```
ct_score
0    1000
1    1000
2    1000
3    1000
4    1000
5    1000
6    1000
7    1000
8    1000
9    1000
10   1000
11   1000
12   1000
13   1000
dtype: int, object, dtype: int, object
```

```
df['winner'].value_counts(normalize=True)
```

```
winner
ct    0.565
t     0.435
dtype: int, object, dtype: int, object
```

```
df['map'].value_counts(normalize=True)
```

```
map
de_cache_competitive      0.126
de_dust2competitive       0.120
de_mirage competitive     0.120
de_inferno competitive    0.120
de_train competitive      0.120
de_cobble competitive     0.120
de_overpass competitive   0.120
de_nuke competitive       0.120
dtype: int, object, dtype: int, object
```

```
df['round'].value_counts(normalize=True)
```

```
round
1    0.063
2    0.063
3    0.063
4    0.063
5    0.063
6    0.063
7    0.063
8    0.063
9    0.063
10   0.063
11   0.063
12   0.063
13   0.063
14   0.063
15   0.063
16   0.063
17   0.063
18   0.063
19   0.063
20   0.063
21   0.063
22   0.063
23   0.063
24   0.063
25   0.063
26   0.063
27   0.063
28   0.063
29   0.063
30   0.063
dtype: int, object, dtype: int, object
```

```
df['t_eq_val'].value_counts(normalize=True)
```

```
t_eq_val
4000    0.063
3900    0.063
3800    0.063
3700    0.063
3600    0.063
3500    0.063
3400    0.063
3300    0.063
3200    0.063
3100    0.063
3000    0.063
2900    0.063
2800    0.063
2700    0.063
2600    0.063
2500    0.063
2400    0.063
2300    0.063
2200    0.063
2100    0.063
2000    0.063
1900    0.063
1800    0.063
1700    0.063
1600    0.063
1500    0.063
1400    0.063
1300    0.063
1200    0.063
1100    0.063
1000    0.063
900     0.063
800     0.063
700     0.063
600     0.063
500     0.063
400     0.063
300     0.063
200     0.063
100     0.063
0       0.063
dtype: int, object, dtype: int, object
```

```
df['ct_eq_val'].value_counts(normalize=True)
```

```
ct_eq_val
4000    0.063
3900    0.063
3800    0.063
3700    0.063
3600    0.063
3500    0.063
3400    0.063
3300    0.063
3200    0.063
3100    0.063
3000    0.063
2900    0.063
2800    0.063
2700    0.063
2600    0.063
2500    0.063
2400    0.063
2300    0.063
2200    0.063
2100    0.063
2000    0.063
1900    0.063
1800    0.063
1700    0.063
1600    0.063
1500    0.063
1400    0.063
1300    0.063
1200    0.063
1100    0.063
1000    0.063
900     0.063
800     0.063
700     0.063
600     0.063
500     0.063
400     0.063
300     0.063
200     0.063
100     0.063
0       0.063
dtype: int, object, dtype: int, object
```

```
df['t_money'].value_counts(normalize=True)
```

```
t_money
3900    0.063
3800    0.063
3700    0.063
3600    0.063
3500    0.063
3400    0.063
3300    0.063
3200    0.063
3100    0.063
3000    0.063
2900    0.063
2800    0.063
2700    0.063
2600    0.063
2500    0.063
2400    0.063
2300    0.063
2200    0.063
2100    0.063
2000    0.063
1900    0.063
1800    0.063
1700    0.063
1600    0.063
1500    0.063
1400    0.063
1300    0.063
1200    0.063
1100    0.063
1000    0.063
900     0.063
800     0.063
700     0.063
600     0.063
500     0.063
400     0.063
300     0.063
200     0.063
100     0.063
0       0.063
dtype: int, object, dtype: int, object
```

```
df['ct_money'].value_counts(normalize=True)
```

```
ct_money
3900    0.063
3800    0.063
3700    0.063
3600    0.063
3500    0.063
3400    0.063
3300    0.063
3200    0.063
3100    0.063
3000    0.063
2900    0.063
2800    0.063
2700    0.063
2600    0.063
2500    0.063
2400    0.063
2300    0.063
2200    0.063
2100    0.063
2000    0.063
1900    0.063
1800    0.063
1700    0.063
1600    0.063
1500    0.063
1400    0.063
1300    0.063
1200    0.063
1100    0.063
1000    0.063
900     0.063
800     0.063
700     0.063
600     0.063
500     0.063
400     0.063
300     0.063
200     0.063
100     0.063
0       0.063
dtype: int, object, dtype: int, object
```

```
df['t_helmets'].value_counts(normalize=True)
```

```
t_helmets
0    0.200
1    0.200
2    0.200
3    0.200
4    0.200
5    0.200
dtype: int, object, dtype: int, object
```

```
df['ct_helmets'].value_counts(normalize=True)
```

```
ct_helmets
0    0.200
1    0.200
2    0.200
3    0.200
4    0.200
5    0.200
dtype: int, object, dtype: int, object
```

```
df['t_defuse'].value_counts(normalize=True)
```

```
t_defuse
0    0.559
1    0.441
dtype: int, object, dtype: int, object
```

```
df['ct_defuse'].value_counts(normalize=True)
```

```
ct_defuse
0    0.559
1    0.441
dtype: int, object, dtype: int, object
```

```
df['t_players'].value_counts(normalize=True)
```

```
t_players
5    1.000
dtype: int, object, dtype: int, object
```

```
df['ct_players'].value_counts(normalize=True)
```

```
ct_players
5    1.000
dtype: int, object, dtype: int, object
```

```
df['t_weapons'].value_counts(normalize=True)
```

```
t_weapons
0    0.200
1    0.200
2    0.200
3    0.200
4    0.200
5    0.200
dtype: int, object, dtype: int, object
```

```
df['ct_weapons'].value_counts(normalize=True)
```

```
ct_weapons
0    0.200
1    0.200
2    0.200
3    0.200
4    0.200
5    0.200
dtype: int, object, dtype: int, object
```

```
df['t_grenades'].value_counts(normalize=True)
```

```
t_grenades
0    0.200
1    0.200
2    0.200
3    0.200
4    0.200
5    0.200
dtype: int, object, dtype: int, object
```

```
df['ct_grenades'].value_counts(normalize=True)
```

```
ct_grenades
0    0.200
1    0.200
2    0.200
3    0.200
4    0.200
5    0.200
dtype: int, object, dtype: int, object
```

```
df['t_score'].value_counts(normalize=True)
```

```
t_score
0    0.063
1    0.063
2    0.063
3    0.063
4    0.063
5    0.063
6    0.063
7    0.063
8    0.063
9    0.063
10   0.063
11   0.063
12   0.063
13   0.063
14   0.063
15   0.063
16   0.063
dtype: int, object, dtype: int, object
```

```
df['ct_score'].value_counts(normalize=True)
```

```
ct_score
0    0.063
1    0.063
2    0.063
3    0.063
4    0.063
5    0.063
6    0.063
7    0.063
8    0.063
9    0.063
10   0.063
11   0.063
12   0.063
13   0.063
dtype: int, object, dtype: int, object
```

```
df.describe
```

```
<bound_method_DataFrame.describe of                map  round winner  t_eq_val  ct_eq_val  t_money  ct_money  \
0      de_cache_competitive      1     ct      4000       4000     3900      3900
1      de_cache_competitive      2     ct      4000       4000     3900      3900
2      de_cache_competitive      3     ct      4000       4000     3900      3900
3      de_cache_competitive      4     ct      4000       4000     3900      3900
4      de_cache_competitive      5     ct      4000       4000     3900      3900
...                     ...    ...    ...       ...        ...      ...       ...
15857  de_dust2competitive     26     ct     18000      18000    16000     16000
15858  de_dust2competitive     27     ct     18000      18000    16000     16000
15859  de_dust2competitive     28     ct     18000      18000    16000     16000
15860  de_dust2competitive     29     ct     18000      18000    16000     16000
15861  de_dust2competitive     30     ct     18000      18000    16000     16000

       t_helmets  ct_helmets  t_defuse  ct_defuse  t_players  ct_players  \
0              0           0         0          0          5           5
1              0           0         0          0          5           5
2              0           0         0          0          5           5
3              0           0         0          0          5           5
4              0           0         0          0          5           5
...          ...         ...       ...        ...        ...         ...
15857          5           5         0          0          5           5
15858          5           5         0          0          5           5
15859          5           5         0          0          5           5
15860          5           5         0          0          5           5
15861          5           5         0          0          5           5

       t_weapons  ct_weapons  t_grenades  ct_grenades  t_score  ct_score
0              0           0           0            0        0         0
1              0           0           0            0        1         0
2              0           0           0            0        2         0
3              0           0           0            0        3         0
4              0           0           0            0        4         0
...          ...         ...         ...          ...      ...       ...
15857          5           5           0            0       12        13
15858          5           5           0            0       13        13
15859          5           5           0            0       14        13
15860          5           5           0            0       15        13
15861          5           5           0            0       16        13

[15862 rows, 19 columns]>
```

DATA VISUALIZATION

```
df.columns
```

```
Index(['map', 'round', 'winner', 't_eq_val', 'ct_eq_val', 't_money',
       'ct_money', 't_helmets', 'ct_helmets', 't_defuse', 'ct_defuse',
       't_players', 'ct_players', 't_weapons', 'ct_weapons', 't_grenades',
       'ct_grenades', 't_score', 'ct_score'],
      dtype: int, object)
```

```
sns.set_style("whitegrid")
plt.rcParams['figure.size'] = (12, 6)
plt.rcParams['figure.dpi'] = 100
```

```
plt.figure(figsize=(12, 6))
sns.countplot(x='round', hue='winner', palette='Set2', data=df)
plt.title('Round Outcomes')
plt.xlabel('Round')
plt.ylabel('Count')
plt.legend(title='Winner')
plt.tight_legend()
plt.show
```

The Counter-Side is ct

The Counter-Side is ct (Counter-Terrorists) wins more than the Terrorists (t) in almost all rounds. Counter-Terrorists win more than Terrorists in almost all rounds. The Counter-Terrorists have a strong advantage in most rounds, with consistently higher win counts across the majority of rounds. The Terrorists (t) win fewer rounds overall, with their wins distributed across the rounds but at lower frequencies. The Counter-Terrorists (ct) dominate the game, winning most rounds and maintaining a significant advantage throughout the match.

Analysis on the total time the maps are played according to the round time is 1:45

The Total time Maps played is 1:45 for Round 30

```
df[df['round'] == 30].shape[0] * 1:45 / 1000
```

```
1:45
```

```
df[df['round'] == 30].shape[0]
```

```
1000
```

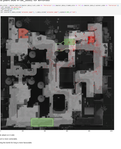
```
df[df['round'] == 30]['map'].value_counts
```

```
map
de_cache_competitive      125
de_dust2competitive       125
de_mirage competitive     125
de_inferno competitive    125
de_train competitive      125
de_cobble competitive     125
de_overpass competitive   125
de_nuke competitive       125
dtype: int, object, dtype: int, object
```

```
df[df['round'] == 30].groupby('winner')['map'].count
```

```
winner
ct    565
t     435
dtype: int, object, dtype: int, object
```

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='bar', color=['blue', 'red'], rot=0)
```



Terrorist_vs_CounterTerrorist

Terrorist_vs_CounterTerrorist

```
df[df['round'] == 30].groupby('map')['winner'].count.plot(kind='bar', color=['blue', 'red'], rot=0)
```

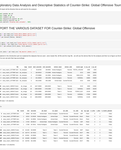

Round Types

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```


Winner Round Types

Winner Round Types

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```



Which maps are player like to be in CSGO with their weapons

```
df[df['round'] == 30].groupby('map')['winner'].count.plot(kind='pie', autopct='%1.1f%%')
```

Which weapon is widely used for killing the opponent

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```

Weapons Types

```
df[df['round'] == 30].groupby('map')['winner'].count.plot(kind='pie', autopct='%1.1f%%')
```

A player are playing with Pistol which body part will they target

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```

A player are playing with Rifle which body part will they target

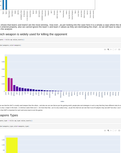
```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```

A player are playing with SMG which body part will they target

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```

A player are playing with Sniper which body part will they target

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```



Total Maps Played Round Type Round Winner Round

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```

Round Winner

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```

Round Winner

```
df[df['round'] == 30].groupby('winner')['map'].count.plot(kind='pie', autopct='%1.1f%%')
```

As Explained above we will analyze de_dust 2 map

```
df[df['map'] == 'de_dust2competitive'].shape[0]
```

```
1900
```

```
df[df['map'] == 'de_dust2competitive'].groupby('winner')['round'].count
```

```
winner
ct    1074
t      826
dtype: int, object, dtype: int, object
```

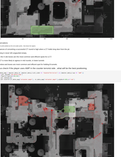
```
df[df['map'] == 'de_dust2competitive'].groupby('winner')['round'].count.plot(kind='bar', color=['blue', 'red'], rot=0)
```



```
df[df['map'] == 'de_dust2competitive'].groupby('round')['winner'].count.plot(kind='bar', color=['blue', 'red'], rot=0)
```

```
df[df['map'] == 'de_dust2competitive'].groupby('round')['winner'].count.plot(kind='pie', autopct='%1.1f%%')
```

```
df[df['map'] == 'de_dust2competitive'].groupby('round')['winner'].count.plot(kind='pie', autopct='%1.1f%%')
```


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

Observations of the Project –

1. de_dust 2 is the most played map , the teams have to be best prepared on this map , as it is well played by all the teams.
2. Teams should focus to end the game before 20 rounds , as the chances of the winning is most for that region , for any map they play .
3. Terrorists have won the more rounds overall , how ever we cannot define the game on that basis , as it is a bit baised , there are some outplays and some out performing teams as well
4. We will need player stats to analyze and predict the winning maps , positioning , as that will play most impact , rest all the rounds we saw were almost the same for both the sides
5. We saw that the normal rounds have more chances of winning the rounds , and teams focus on full eco rather semi eco , as best analysis is , team will have next round to buy the weapons as well
6. Players should practice more to hit the head and the chest area ,as that have more kills in the whole dataset
7. Players should strategize to play with rifles more , as they have more winning sides that smgs and others
8. Terrorists should strategize to focus more on bomb plant , as that have lead to more wins than the other rounds , and the probability of winning increases there.
9. In terms of de_dust 2 , the Counter terrorist should hold the A long region more , as that has the max counter terrorist dominated area
10. Counter terrorist should use the AWP for A long , A site and the Mid doors region in the map , as that is the best region to defend and most Counter terrorist dominated area.
11. Terrorists should plan to plant on A more , as the A short is most weak for counter terrorists , and the other team should focus on making that region stronger .

How ever as per the subject expert , the data is not enough to go for further predictions , as there are some outlier performances from the world best players that can lead to unexpected wins as there have been in past as well , so we will also need , players data to analyze more towards each and every prediction and analysis for winning .

At the end , we would like to share this that we have been playing this gaming series for Counter strike 1 , 2 , GO for 12 years now , this is one of the life we live and were very confident when we got the project as we wanted to analyze the thing that we do from a very professional angle , we would like to thank Sayantani ma'am for giving us the opportunity to do this project , and share subject knowledge so we can do this project effectively and learn from each and every step . I would like to thank Praxis Business School for providing us such a opportunity to present our work with you all.