METADATA

The dataset was originally published by Skybox as part of their CS: GO AI Challenge, running from Spring to Fall 2020. The data set consists of ~700 demos from high-level tournament play in 2019 and 2020. Warmup rounds and restarts have been filtered, and for the remaining live rounds, a round snapshot has been recorded every 20 seconds until the round is decided. Following the initial publication, It has been pre-processed and flattened to improve readability and make it easier for algorithms to process. The total number of snapshots is 122411.

Skybox website: https://skybox.gg/
Learn more about CS:GO: https://en.wikipedia.org/wiki/Counter-Strike:_Global_Offensive
View CS: GO on Steam
Store: https://store.steampowered.com/app/730/CounterStrike_Global_Offensive/
Find in-depth information on competitive CS:GO: https://www.hltv.org/

Variable	Definition	Key
time_left	The time left in the current round.	
ct_score	The current score of the Counter-Terrorist team.	
t_score	The current score of the Terrorist team.	
map	The map the round is being played on.	E.g. de_dust2, de_inferno and de_overpass
bomb_planted	If the bomb has been planted or not.	False = No, True = Yes
ct_health	The total health of all Counter-Terrorist players.	Player health in range 0-100.
t_health	The total health of all Terrorist players.	Player health in range 0-100.
ct_armor	The total armor of all Counter-Terrorist players.	
t_armor	The total armor of all Terrorist players.	
ct_money	The total bankroll of all Counter-Terrorist players.	Amount in USD.
t_money	The total bankroll of all Terrorist players.	Amount in USD.
ct_helmets	Number of helmets on the Counter-Terrorist team.	
t_helmets	Number of helmets on the Terrorist team.	
ct_defuse_kits	Number of defuse kits on the Counter-Terrorist team.	
ct_players_alive	Number of alive players on the Counter-Terrorist team.	Range 0 to 5.

Variable	Definition	Key
t_players_alive	Number of alive players on the Terrorist team.	Range 0 to 5.
ct_weapon_X	Weapon X count on Counter-Terrorist team.	E.g. Ak47, Deagle and UMP45.
t_weapon_X	Weapon X count on Terrorist team.	E.g. Ak47, Deagle and UMP45.
ct_grenade_X	Grenade X count on Counter-Terrorist team.	E.g. HeGrenade, Flashbang.
t_grenade_X	Grenade X count on Terrorist team.	E.g. HeGrenade, Flashbang.
round_winner	Winner.	CT = Counter-Terrorist, T = Terrorist

REQUIREMENTS

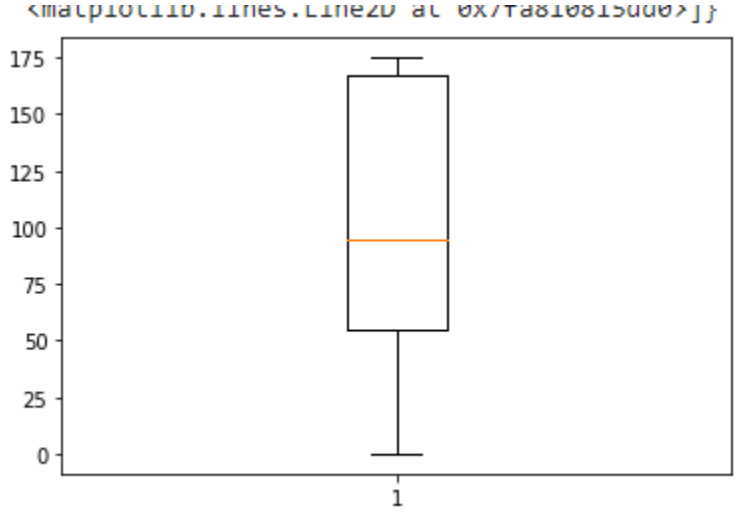
- What types of machine learning models perform best on this dataset?
- Which features are most indicative of which team wins the round?
- How often does the team with the most money win?
- Are some weapons favorable to others?
- What attributes should your team have to win? Health, armor, or money?

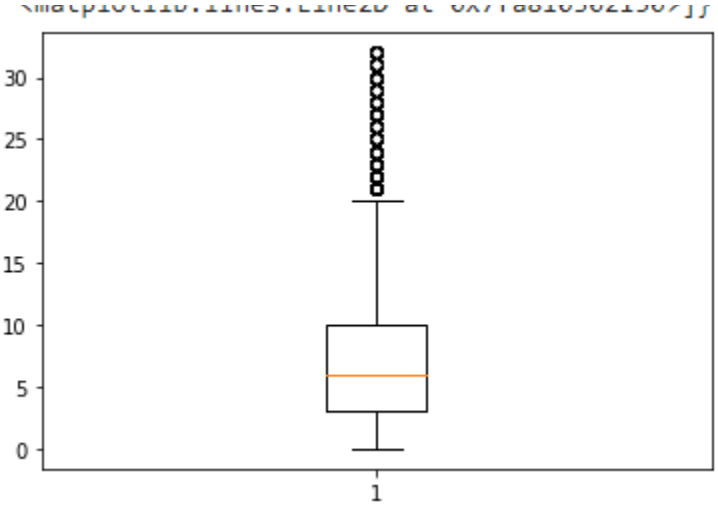
VISUALIZATIONS

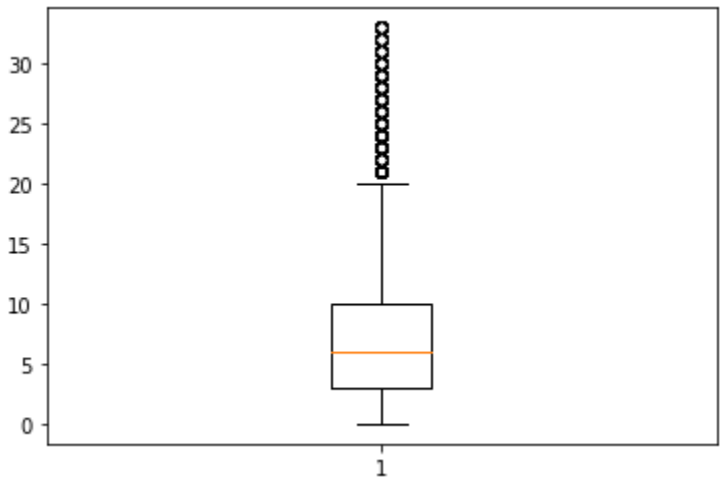
Importing Important Libraries

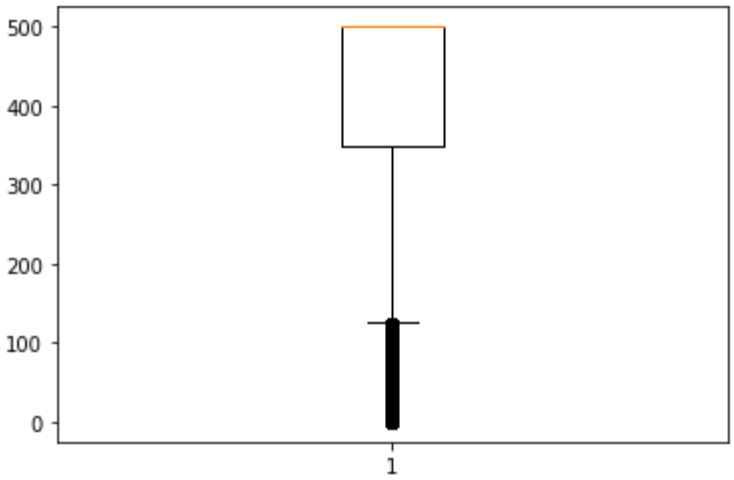
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,mean_squared_error
import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from imblearn.over_sampling import SMOTE
from scipy.stats import sem
from sklearn.model_selection import RepeatedKFold, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split, RepeatedKFold, cross_val_score
from sklearn.metrics import classification_report

warnings.filterwarnings('ignore')
```

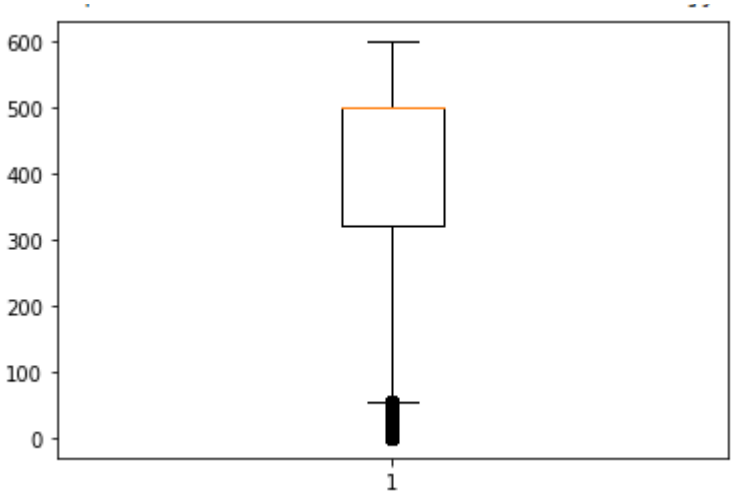
Box Plots for Data



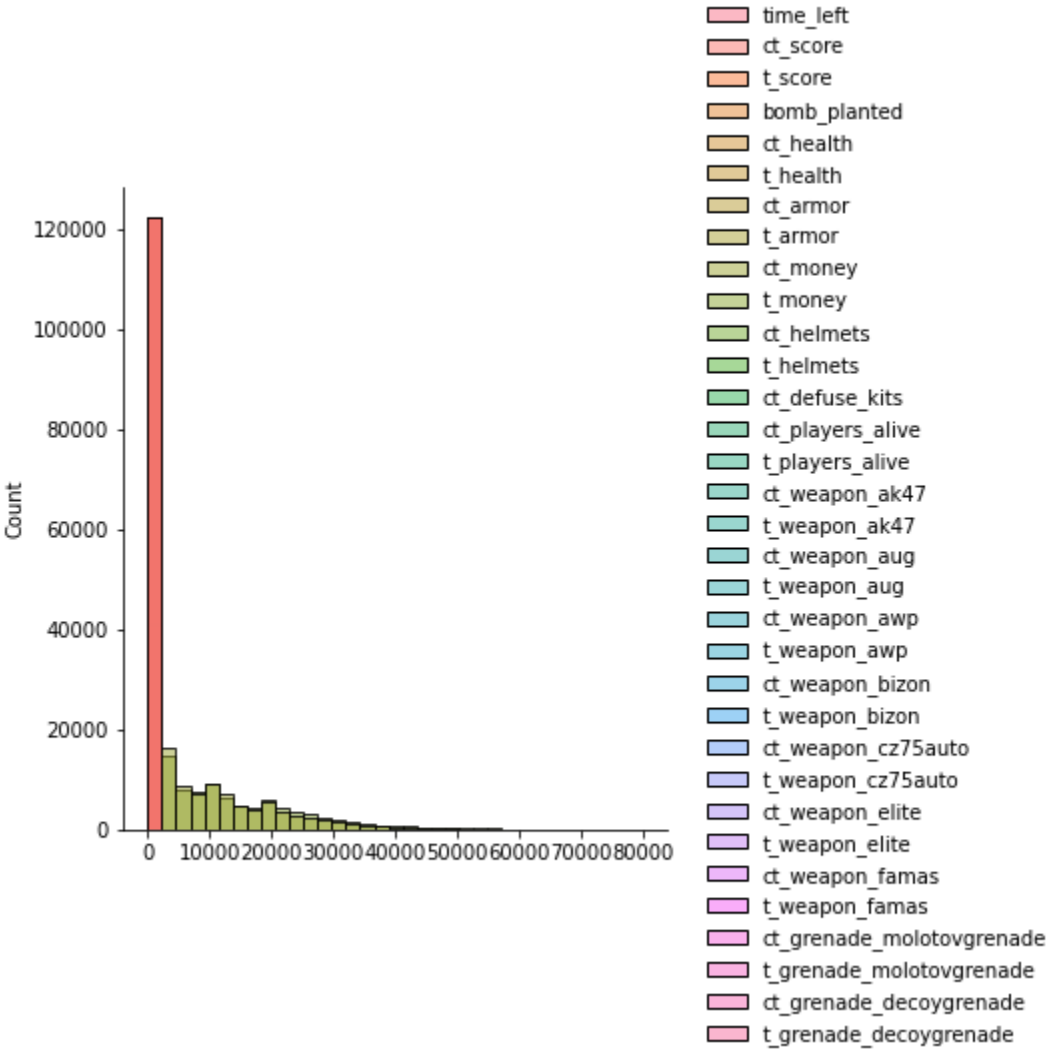

30
25
20
15
10
5
0
1


30
25
20
15
10
5
0
1

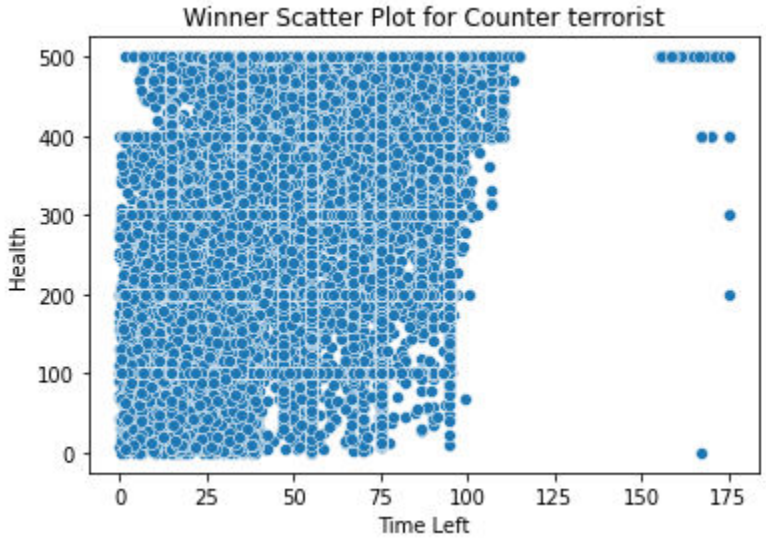

500
400
300
200
100
0
1

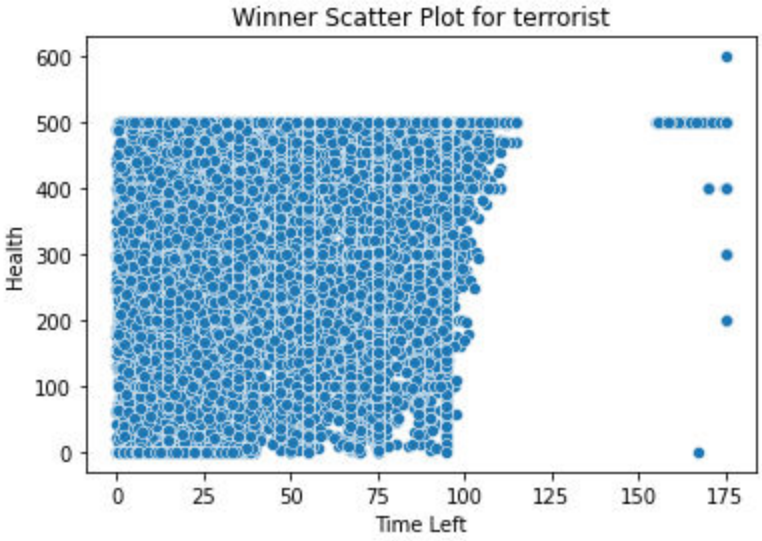


Histograms for Data



Scatter Plots





As we above plots showing there are outliers and histograms showing data is not normalized.

```
[6]  df.shape

     (122410, 35)
```

This showing the shape of data

As there are some columns that are categorical which needs to be converted into numerical

map	bomb_planted	round_winner
de_dust2	False	CT
de_dust2	False	CT
de_dust2	False	CT
de_dust2	False	CT
de_dust2	False	CT

After Converting the columns

```
[7]  le = LabelEncoder()
     df['map'] = le.fit_transform(df['map'])
     df['bomb_planted'] = le.fit_transform(df['bomb_planted'])
     df['round_winner'] = le.fit_transform(df['round_winner'])
```

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...	t_weapon_cz75auto	ct_weapon_elite	t_weapon_elite	ct_weapon_famas	t_weapon_famas	ct_grenade_molotovgrenade	t_grenade_m
0	175.00	0	0	1	0	500	500	0	0	4000	...	0	0	0	0	0	0	
1	156.03	0	0	1	0	500	500	400	300	600	...	0	0	0	0	0	0	
2	96.03	0	0	1	0	391	400	294	200	750	...	0	0	0	0	0	0	
3	76.03	0	0	1	0	391	400	294	200	750	...	0	0	0	0	0	0	
4	174.97	1	0	1	0	500	500	192	0	18350	...	0	0	0	0	0	0	

5 rows × 35 columns

Training Model Before Preprocessing And Checking Accuracy

```
[8]  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)
     classifier = KNeighborsClassifier(n_neighbors=7)
     classifier.fit(X_train, y_train)
     y_pred = classifier.predict(X_test)
     print('accuracy score is: '+str(accuracy_score(y_test, y_pred)))

     accuracy score is: 0.7407074585409689
```

Performing Preprocessing

Removing null values which this data set does not have

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

```
time_left                   0
ct_score                    0
t_score                     0
map                         0
bomb_planted                0
ct_health                   0
t_health                    0
ct_armor                    0
t_armor                     0
ct_money                    0
t_money                     0
ct_helmets                  0
t_helmets                   0
ct_defuse_kits              0
ct_players_alive            0
t_players_alive             0
ct_weapon_ak47              0
t_weapon_ak47               0
ct_weapon_aug               0
t_weapon_aug                0
ct_weapon_awp               0
t_weapon_awp                0
ct_weapon_bizon             0
```

Removing duplicate Values

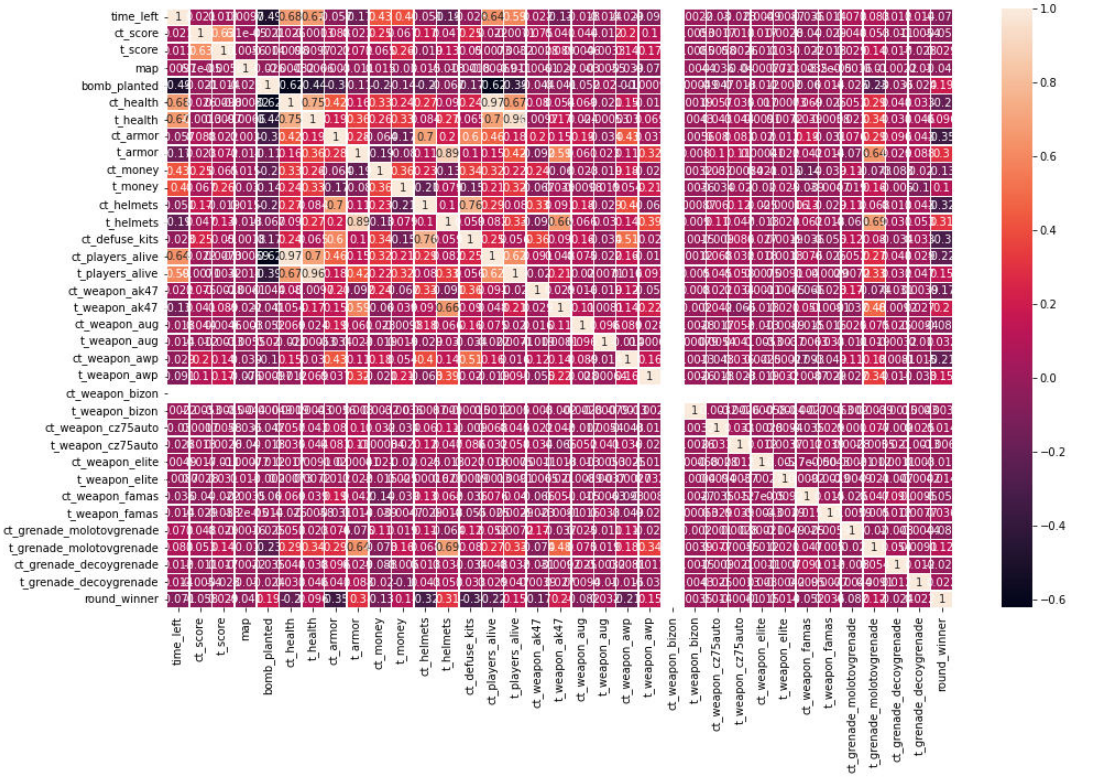
```
df.dropna(inplace=True)
df.isnull().sum()
```

```
time_left                0
ct_score                 0
t_score                  0
map                      0
bomb_planted             0
ct_health                0
t_health                 0
ct_armor                 0
t_armor                  0
ct_money                 0
t_money                  0
ct_helmets               0
t_helmets                0
ct_defuse_kits           0
ct_players_alive         0
t_players_alive          0
ct_weapon_ak47           0
t_weapon_ak47            0
ct_weapon_aug            0
t_weapon_aug             0
ct_weapon_awp            0
t_weapon_awp             0
ct_weapon_bizon          0
t_weapon_bizon           0
ct_weapon_cz75auto       0
t_weapon_cz75auto        0
ct_weapon_elite          0
```

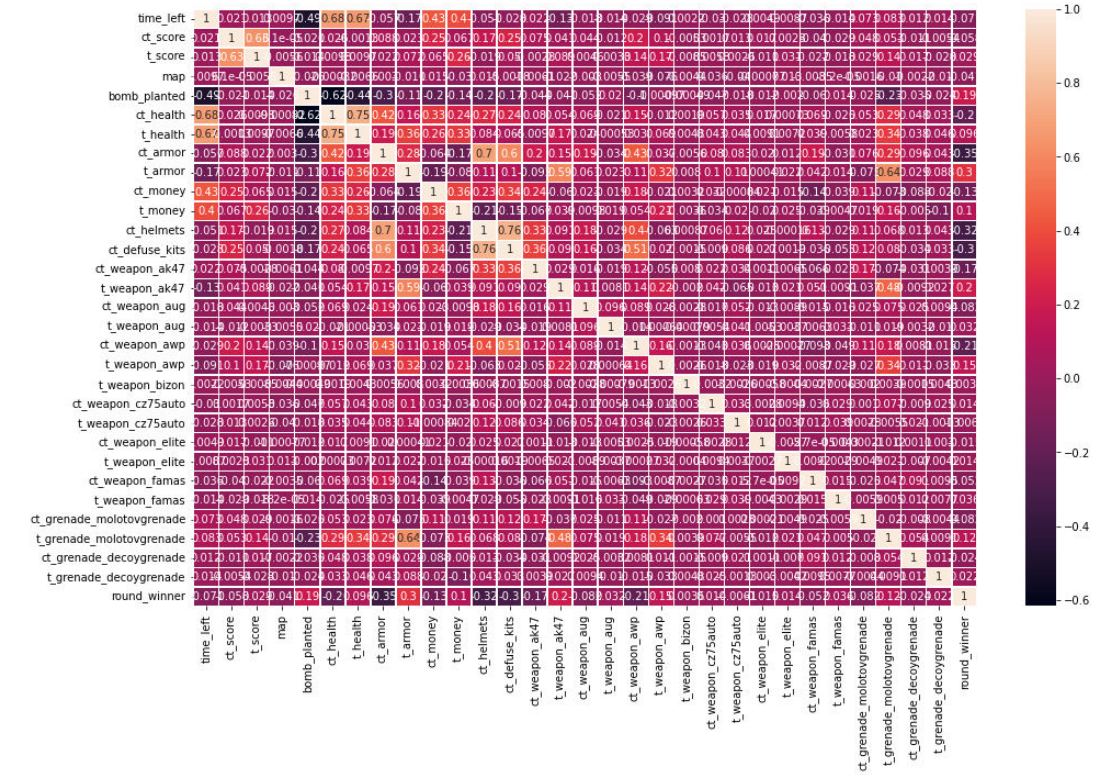
This Shows the shape of data set indicates the duplicate values removed

```
[13] df.drop_duplicates(subset=None, keep='first', inplace=True) #(117001, 35)

     df.shape

     (117001, 35)
```

Plotting a Heat Map to See the Correlation and by looking it, visible, errors in plot



After Dealing with Errors



Normalizing data by Max Method

```
[20] df[df.columns[0]] = df[df.columns[0]] / df[df.columns[0]].max()
     df[df.columns[1]] = df[df.columns[1]] / df[df.columns[1]].max()
     df[df.columns[2]] = df[df.columns[2]] / df[df.columns[2]].max()
     df[df.columns[5]] = df[df.columns[5]] / df[df.columns[5]].max()
     df[df.columns[6]] = df[df.columns[6]] / df[df.columns[6]].max()
     df[df.columns[7]] = df[df.columns[7]] / df[df.columns[7]].max()
     df[df.columns[8]] = df[df.columns[8]] / df[df.columns[8]].max()
     df[df.columns[9]] = df[df.columns[9]] / df[df.columns[8]].max()
     df[df.columns[10]] = df[df.columns[10]] / df[df.columns[10]].max()
     df.head()
```

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...	t_weap
0	1.000000	0.00000	0.0	1	0	1.000	0.833333	0.000	0.0	4000.0	...	
1	0.891600	0.00000	0.0	1	0	1.000	0.833333	0.800	0.6	600.0	...	
2	0.548743	0.00000	0.0	1	0	0.782	0.666667	0.588	0.4	750.0	...	
3	0.434457	0.00000	0.0	1	0	0.782	0.666667	0.588	0.4	750.0	...	
4	0.999829	0.03125	0.0	1	0	1.000	0.833333	0.384	0.0	18350.0	...	

5 rows × 31 columns

Removing Outliers and Printing number of outliers in each columns

```
[21] cols = df.select_dtypes(['int64','float64']).columns
     for column in cols:
         q1 = df[column].quantile(0.25)     # First Quartile
         q3 = df[column].quantile(0.75)     # Third Quartile
         IQR = q3 - q1                              # Inter Quartile Range

         ll= q1 - 1.5*IQR                         # Lower Limit
         ul = q3 + 1.5*IQR                          # Upper Limit

         outliers = df[(df[column] < ll) | (df[column] > ul)]
         print('Number of outliers in "' + column + '" : ' + str(len(outliers)))
```

```
Number of outliers in "time_left" : 0
Number of outliers in "ct_score" : 598
Number of outliers in "t_score" : 369
Number of outliers in "map" : 0
Number of outliers in "bomb_planted" : 13684
Number of outliers in "ct_health" : 2782
Number of outliers in "t_health" : 1924
Number of outliers in "ct_armor" : 0
Number of outliers in "t_armor" : 0
Number of outliers in "ct_money" : 4171
Number of outliers in "t_money" : 2934
Number of outliers in "ct_helmets" : 0
Number of outliers in "ct_defuse_kits" : 0
Number of outliers in "ct_weapon_ak47" : 25809
Number of outliers in "t_weapon_ak47" : 0
Number of outliers in "ct_weapon_aug" : 12198
Number of outliers in "t_weapon_aug" : 891
Number of outliers in "ct_weapon_awp" : 170
Number of outliers in "t_weapon_awp" : 6
Number of outliers in "t_weapon_bizon" : 10
Number of outliers in "ct_weapon_cz75auto" : 14833
Number of outliers in "t_weapon_cz75auto" : 10459
Number of outliers in "ct_weapon_elite" : 455
Number of outliers in "t_weapon_elite" : 215
Number of outliers in "ct_weapon_famas" : 10686
Number of outliers in "t_weapon_famas" : 557
Number of outliers in "ct_grenade_molotovgrenade" : 546
Number of outliers in "t_grenade_molotovgrenade" : 0
Number of outliers in "ct_grenade_decoygrenade" : 3141
Number of outliers in "t_grenade_decoygrenade" : 2833
Number of outliers in "round_winner" : 0
```

Training Model After Preprocessing And Checking Accuracy

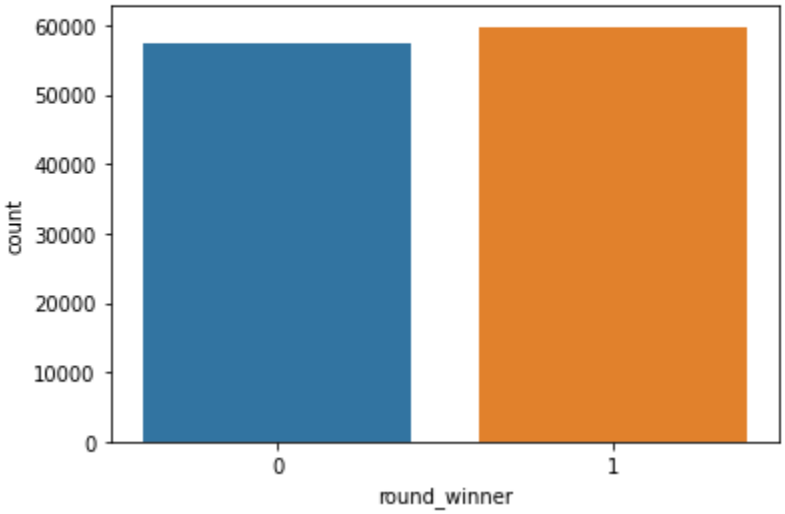
```
[24] X=df.drop(["round_winner"],axis=1)
     Y=df['round_winner']
     X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3)
     classifier = KNeighborsClassifier(n_neighbors=7)
     classifier.fit(X_train, y_train)
     y_pred = classifier.predict(X_test)
     print('accuracy score is: '+str(accuracy_score(y_test, y_pred)))
```

```
accuracy score is: 0.7307484117261617
```

The data set Does not Have Overfitting, however have Class Imbalance

```
sns.countplot(Y,label='round_winner')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5a84b56ad0>
```

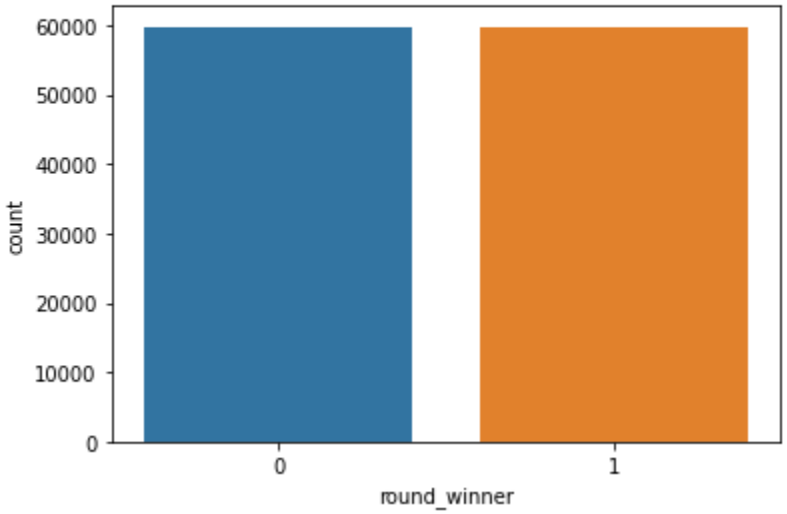


After Adding dummy Values

```
[26]
    oversample = SMOTE()
    x, y = oversample.fit_resample(X,Y)
```

```
[ ] sns.countplot(y,label='round_winner')
```

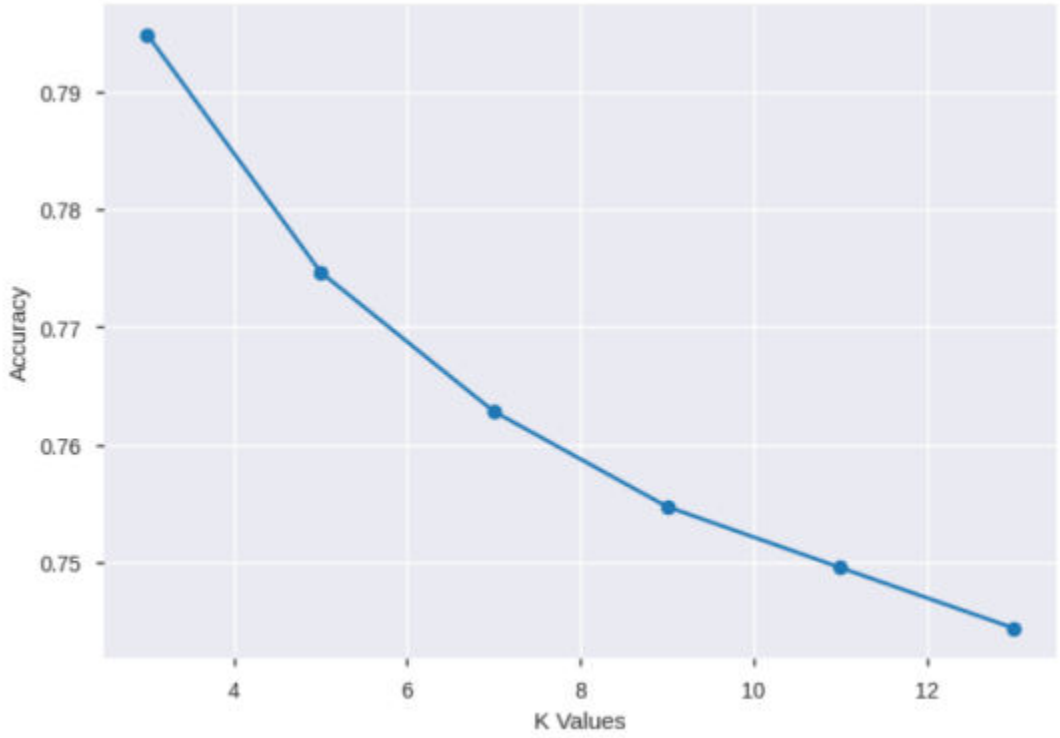
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0203033690>
```



Training Model After Class Imbalance And Checking Accuracy

```
[27] 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)
     classifier = KNeighborsClassifier(n_neighbors=7)
     classifier.fit(X_train, y_train)
     y_pred = classifier.predict(X_test)
     print('accuracy score is: '+str(accuracy_score(y_test, y_pred)))

     accuracy score is: 0.7370212291126175
```

Loss Graph



Precision, Recall, and F-Score

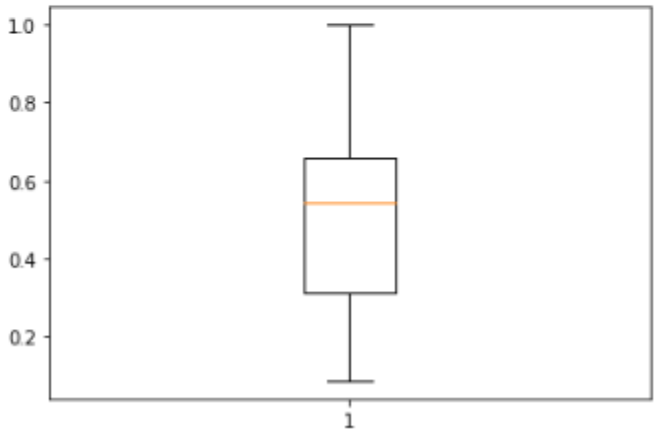
```
[30]
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.75      0.73     17773
           1       0.74      0.71      0.73     18074

    accuracy                           0.73     35847
   macro avg       0.73      0.73      0.73     35847
weighted avg       0.73      0.73      0.73     35847
```

Box Plot after Preprocessing

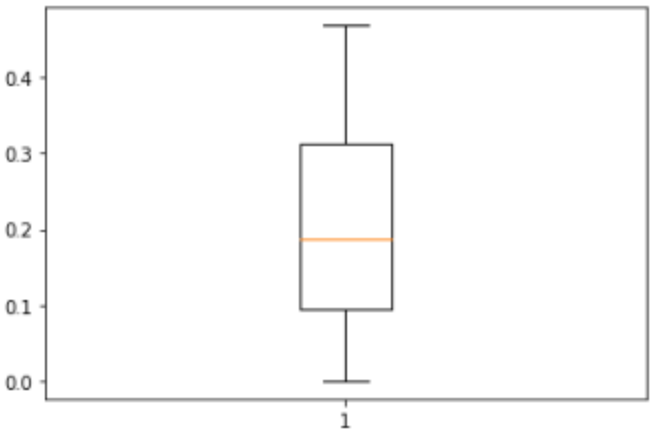
```
[31] plt.boxplot(df.time_left)
```

```
{'boxes': [<matplotlib.lines.Line2D at 0x7f4f948aaa90>],
 'caps': [<matplotlib.lines.Line2D at 0x7f4f94047c90>,
  <matplotlib.lines.Line2D at 0x7f4f93777c50>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f4f94061750>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x7f4f94061210>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7f4f94047250>,
  <matplotlib.lines.Line2D at 0x7f4f94047390>]}
```



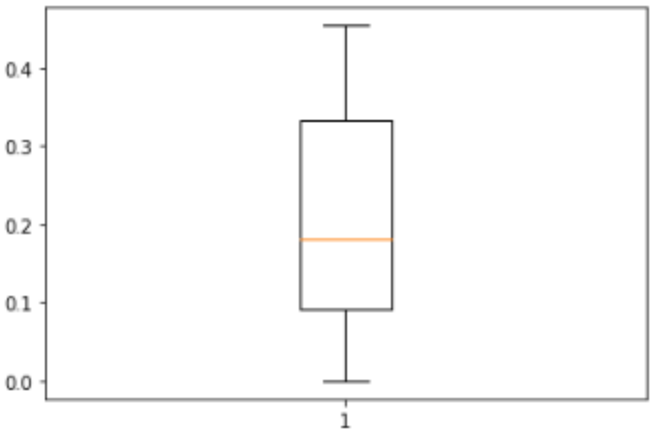
```
[32] plt.boxplot(df.ct_score)
```

```
{'boxes': [<matplotlib.lines.Line2D at 0x7f4f949f2110>],
 'caps': [<matplotlib.lines.Line2D at 0x7f4f949e9150>,
  <matplotlib.lines.Line2D at 0x7f4f949e9690>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f4f949d4190>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x7f4f949e9c10>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7f4f949f2690>,
  <matplotlib.lines.Line2D at 0x7f4f949f2bd0>]}
```



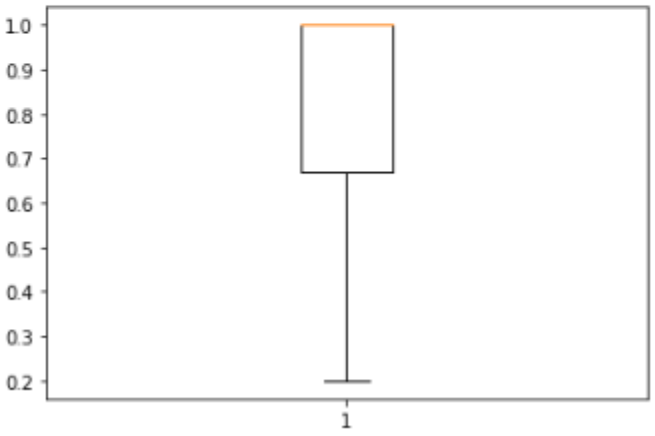

```
plt.boxplot(df.t_score)
```

```
{'boxes': [<matplotlib.lines.Line2D at 0x7f4f949008d0>],
 'caps': [<matplotlib.lines.Line2D at 0x7f4f948d7910>,
  <matplotlib.lines.Line2D at 0x7f4f948d7e50>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f4f94903950>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x7f4f94903410>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7f4f94900e50>,
  <matplotlib.lines.Line2D at 0x7f4f948d73d0>]}
```



```
plt.boxplot(df.ct_health)
```

```
{'boxes': [<matplotlib.lines.Line2D at 0x7f4f949260d0>],
 'caps': [<matplotlib.lines.Line2D at 0x7f4f9494a110>,
  <matplotlib.lines.Line2D at 0x7f4f9494a650>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f4f94943150>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x7f4f9494abd0>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7f4f94926650>,
  <matplotlib.lines.Line2D at 0x7f4f94926b90>]}
```



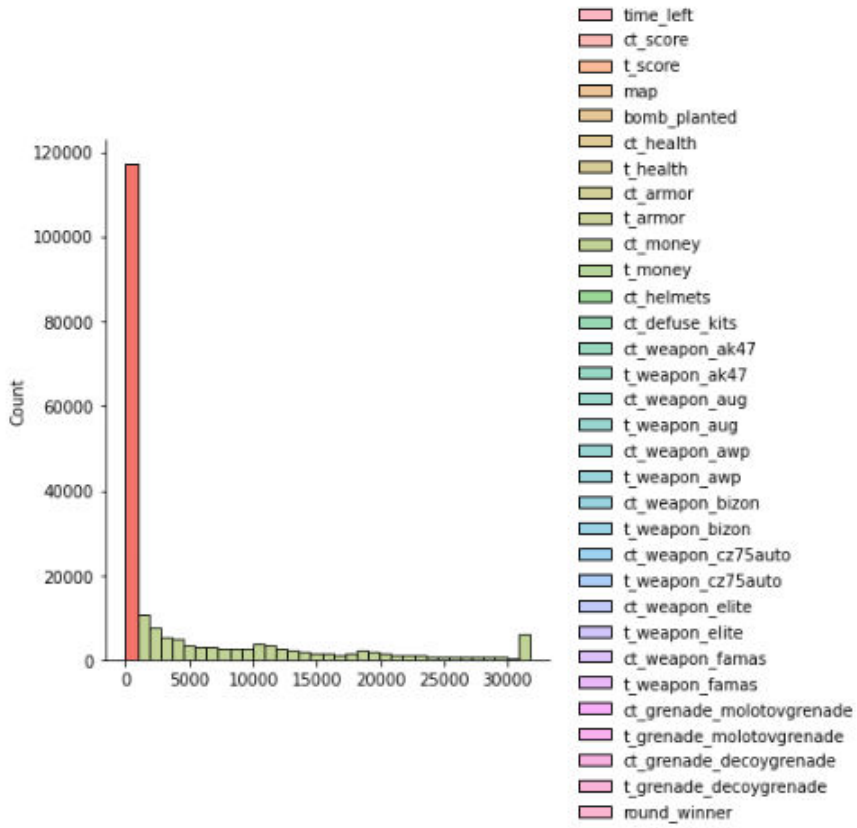

```
[35] plt.boxplot(df.t_health)
```

```
{'boxes': [<matplotlib.lines.Line2D at 0x7f4f94038210>],
 'caps': [<matplotlib.lines.Line2D at 0x7f4f94025250>,
  <matplotlib.lines.Line2D at 0x7f4f94025790>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f4f94031290>],
 'means': [],
 'medians': [<matplotlib.lines.Line2D at 0x7f4f94025d10>],
 'whiskers': [<matplotlib.lines.Line2D at 0x7f4f94038790>,
  <matplotlib.lines.Line2D at 0x7f4f94038cd0>]}
```

Histograms after Preprocessing

```
sns.displot(df, bins=35)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f4f94916ed0>
```



Methaws Coefficient Co relation

```
matthews_corrcoef(y_test, y_pred)
```

```
0.4595912282295406
```