Importing Necessary Packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Preprocessing

```
df=pd.read_csv(r"C:\Users\Arigala.Adarsh\Downloads\
csgo_round_snapshots.csv")

df.head()

   time_left  ct_score  t_score       map  bomb_planted  ct_health
t_health  \
0     175.00       0.0      0.0  de_dust2         False      500.0
500.0
1     156.03       0.0      0.0  de_dust2         False      500.0
500.0
2      96.03       0.0      0.0  de_dust2         False      391.0
400.0
3      76.03       0.0      0.0  de_dust2         False      391.0
400.0
4     174.97       1.0      0.0  de_dust2         False      500.0
500.0

   ct_armor  t_armor  ct_money  ...  t_grenade_flashbang  \
0       0.0      0.0    4000.0  ...                  0.0
1     400.0    300.0     600.0  ...                  0.0
2     294.0    200.0     750.0  ...                  0.0
3     294.0    200.0     750.0  ...                  0.0
4     192.0      0.0   18350.0  ...                  0.0

   ct_grenade_smokegrenade  t_grenade_smokegrenade  \
0                      0.0                     0.0
1                      0.0                     2.0
2                      0.0                     2.0
3                      0.0                     0.0
4                      0.0                     0.0

   ct_grenade_incendiarygrenade  t_grenade_incendiarygrenade  \
0                           0.0                          0.0
1                           0.0                          0.0
2                           0.0                          0.0
3                           0.0                          0.0
```

```
4                           0.0                          0.0

   ct_grenade_molotovgrenade  t_grenade_molotovgrenade  \
0                        0.0                       0.0
1                        0.0                       0.0
2                        0.0                       0.0
3                        0.0                       0.0
4                        0.0                       0.0

   ct_grenade_decoygrenade  t_grenade_decoygrenade  round_winner
0                      0.0                     0.0            CT
1                      0.0                     0.0            CT
2                      0.0                     0.0            CT
3                      0.0                     0.0            CT
4                      0.0                     0.0            CT

[5 rows x 97 columns]
```

```
df.shape
```

```
(122410, 97)
```

```
df.info()
```

```
<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
 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
 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
 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
dtypes: bool(1), float64(94), object(2)
memory usage: 89.8+ MB
```

```
df.describe()
```

```
           time_left       ct_score        t_score      ct_health  \
count  122410.000000  122410.000000  122410.000000  122410.000000
mean       97.886922       6.709239       6.780435     412.106568
std        54.465238       4.790362       4.823543     132.293290
min         0.010000       0.000000       0.000000       0.000000
25%        54.920000       3.000000       3.000000     350.000000
50%        94.910000       6.000000       6.000000     500.000000
75%       166.917500      10.000000      10.000000     500.000000
max       175.000000      32.000000      33.000000     500.000000

            t_health       ct_armor        t_armor       ct_money  \
count  122410.000000  122410.000000  122410.000000  122410.000000
mean      402.714500     314.142121     298.444670    9789.023773
std       139.919033     171.029736     174.576545   11215.042286
min         0.000000       0.000000       0.000000       0.000000
25%       322.000000     194.000000     174.000000    1300.000000
50%       500.000000     377.000000     334.000000    5500.000000
75%       500.000000     486.000000     468.000000   14600.000000
```

```
max       600.000000     500.000000     500.000000   80000.000000

             t_money     ct_helmets  ...  ct_grenade_flashbang  \
count  122410.000000  122410.000000  ...         122410.000000
mean    11241.036680       2.053901  ...              1.853157
std     12162.806759       1.841470  ...              1.772791
min         0.000000       0.000000  ...              0.000000
25%      1550.000000       0.000000  ...              0.000000
50%      7150.000000       2.000000  ...              1.000000
75%     18000.000000       4.000000  ...              3.000000
max     80000.000000       5.000000  ...              7.000000

       t_grenade_flashbang  ct_grenade_smokegrenade
t_grenade_smokegrenade  \
count        122410.000000            122410.000000
122410.000000
mean              1.858100                 1.540814
1.627146
std               1.794473                 1.737804
1.829147
min               0.000000                 0.000000
0.000000
25%               0.000000                 0.000000
0.000000
50%               1.000000                 1.000000
1.000000
75%               3.000000                 3.000000
3.000000
max               7.000000                 6.000000
9.000000

       ct_grenade_incendiarygrenade  t_grenade_incendiarygrenade  \
count                 122410.000000                122410.000000
mean                       1.001969                     0.019819
std                        1.458084                     0.143933
min                        0.000000                     0.000000
25%                        0.000000                     0.000000
50%                        0.000000                     0.000000
75%                        2.000000                     0.000000
max                        5.000000                     3.000000

       ct_grenade_molotovgrenade  t_grenade_molotovgrenade  \
count              122410.000000             122410.000000
mean                    0.048011                  1.352095
std                     0.227669                  1.663246
min                     0.000000                  0.000000
25%                     0.000000                  0.000000
50%                     0.000000                  1.000000
75%                     0.000000                  2.000000
max                     3.000000                  5.000000
```

```
       ct_grenade_decoygrenade  t_grenade_decoygrenade
count            122410.000000           122410.000000
mean                  0.027694                0.025750
std                   0.169531                0.164162
min                   0.000000                0.000000
25%                   0.000000                0.000000
50%                   0.000000                0.000000
75%                   0.000000                0.000000
max                   3.000000                2.000000

[8 rows x 94 columns]
```

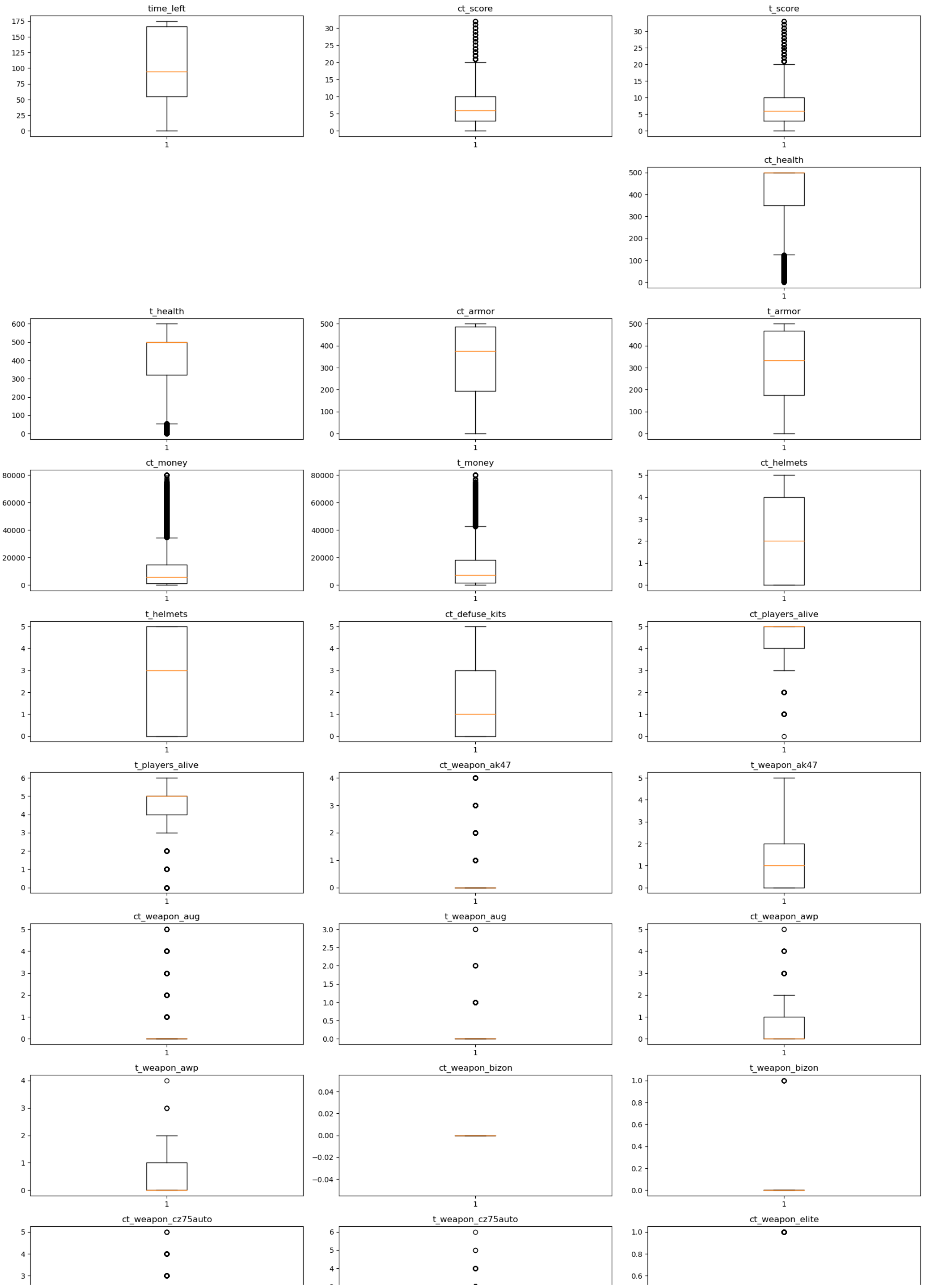
```
df.isnull().sum()
```

```
time_left                    0
ct_score                     0
t_score                      0
map                          0
bomb_planted                 0
                            ..
ct_grenade_molotovgrenade    0
t_grenade_molotovgrenade     0
ct_grenade_decoygrenade      0
t_grenade_decoygrenade       0
round_winner                 0
Length: 97, dtype: int64
```

```
df.duplicated()
```

```
0         False
1         False
2         False
3         False
4         False
          ...
122405    False
122406    False
122407    False
122408    False
122409    False
Length: 122410, dtype: bool
```

```
plt.subplots(figsize=(18,140) )
length=len(df.columns)
for i,j in zip(range(length),df.columns):
    if(df[j].dtypes!="object" and df[j].dtypes!="bool") :
        plt.subplot(length//2,3,i+1)
        plt.boxplot(df[j])
        plt.title(j)
plt.tight_layout()
plt.show()
```


time_left
0 25 50 75 100 125 150 175
1
ct_score
0 5 10 15 20 25 30
1
t_score
0 5 10 15 20 25 30
1
ct_health
0 100 200 300 400 500
1
t_health
0 100 200 300 400 500 600
1
ct_armor
0 100 200 300 400 500
1
t_armor
0 100 200 300 400 500
1
ct_money
0 20000 40000 60000 80000
1
t_money
0 20000 40000 60000 80000
1
ct_helmets
0 1 2 3 4 5
1
t_helmets
0 1 2 3 4 5
1
ct_defuse_kits
0 1 2 3 4 5
1
ct_players_alive
0 1 2 3 4 5
1
t_players_alive
0 1 2 3 4 5 6
1
ct_weapon_ak47
0 1 2 3 4
1
t_weapon_ak47
0 1 2 3 4 5
1
ct_weapon_aug
0 1 2 3 4 5
1
t_weapon_aug
0.0 0.5 1.0 1.5 2.0 2.5 3.0
1
ct_weapon_awp
0 1 2 3 4 5
1
t_weapon_awp
0 1 2 3 4
1
ct_weapon_bizon
−0.04 −0.02 0.00 0.02 0.04
1
t_weapon_bizon
0.0 0.2 0.4 0.6 0.8 1.0
1
ct_weapon_cz75auto
3 4 5
t_weapon_cz75auto
4 5 6
ct_weapon_elite
0.6 0.8 1.0

```
df.describe(include='O')
```

```
              map round_winner
count      122410       122410
unique          8            2
top     de_inferno            T
freq        23811        62406
```

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

```
de_inferno     23811
de_dust2       22144
de_nuke        19025
de_mirage      18576
de_overpass    14081
de_train       13491
de_vertigo     11137
de_cache         145
Name: map, dtype: int64
```

```
# lets see teams how they are successful in winning rounds
counts = df['map'].value_counts()
total = counts.sum()
percentages = counts / total * 100

counts.index
```

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

```
for map_name, count, percent in zip(counts.index, counts.values,
percentages.values):
    print(f'{map_name}: {percent:.2f}%','/',count)
```

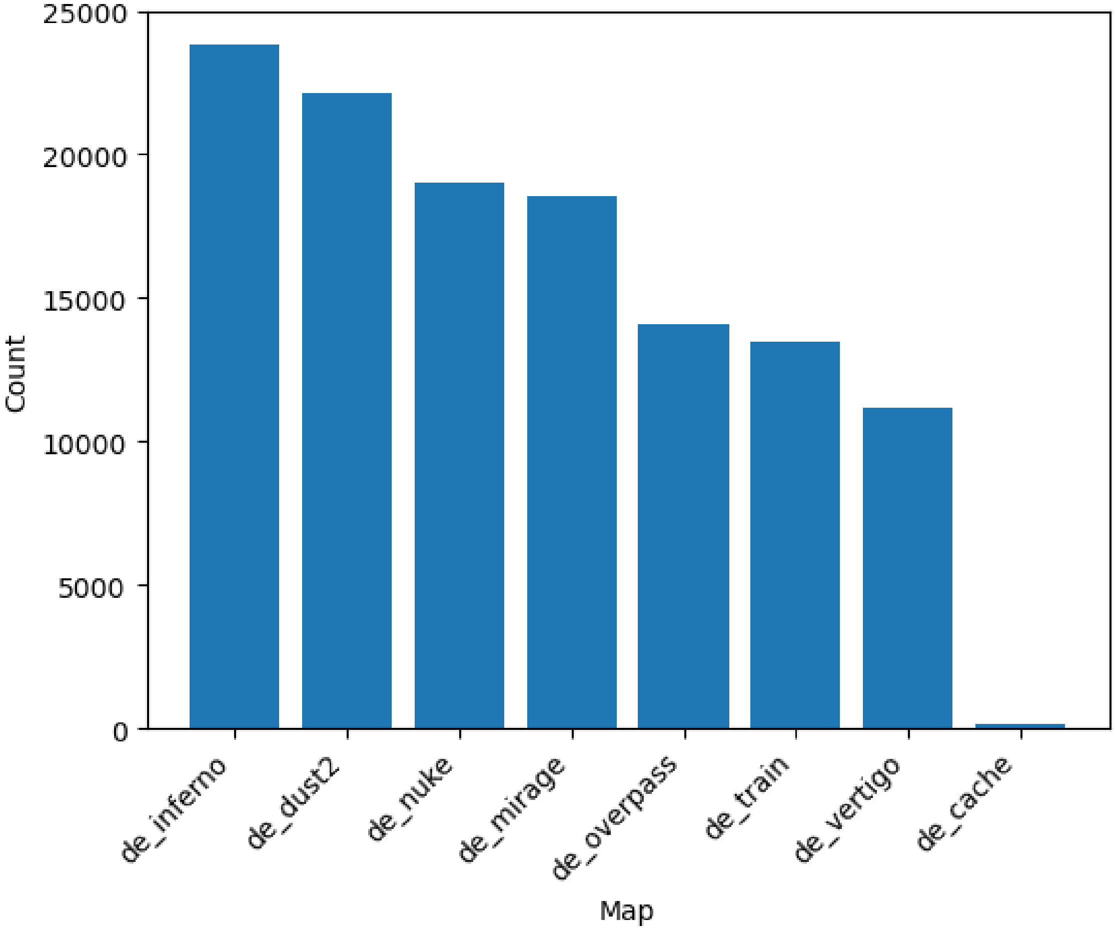
```
de_inferno: 19.45% / 23811
de_dust2: 18.09% / 22144
de_nuke: 15.54% / 19025
de_mirage: 15.18% / 18576
de_overpass: 11.50% / 14081
de_train: 11.02% / 13491
de_vertigo: 9.10% / 11137
de_cache: 0.12% / 145
```

```
plt.bar(counts.index, counts.values)

plt.xticks(rotation=45, ha='right')
plt.xlabel('Map')

plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



```
for i in df.columns:
    if (df[i].dtypes=="object") | (df[i].dtypes=="bool"):
        print("Columns which have categorical values",i)

Columns which have categorical values map
Columns which have categorical values bomb_planted
Columns which have categorical values round_winner

df["bomb_planted"].value_counts()

False    108726
True      13684
Name: bomb_planted, dtype: int64

df["round_winner"].value_counts()

T     62406
CT    60004
Name: round_winner, dtype: int64
```

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

```
de_inferno     23811
de_dust2       22144
de_nuke        19025
de_mirage      18576
de_overpass    14081
de_train       13491
de_vertigo     11137
de_cache         145
Name: map, dtype: int64
```

```
# Converting categorical features into a integer column
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df["bomb_planted"]=le.fit_transform(df["bomb_planted"])
df["map"]=le.fit_transform(df["map"])
df["round_winner"]=le.fit_transform(df["round_winner"])
```

Segregation of the data into dependent and independent columns

```
X=df.drop(columns=["round_winner"])
y=df[["round_winner"]]
```

Spliting the dataset into Train and Test datasets

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,rando
m_state=42)
```

```
# Scaling the data
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

Choosing the model

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda=LinearDiscriminantAnalysis()
```

Training the model

```
lda.fit(x_train,y_train)
```

```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\sklearn\utils\
validation.py:1183: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
LinearDiscriminantAnalysis()
```

```
lda.transform(x_test)
```

```
array([[-0.31066701],
       [ 0.16160545],
       [-2.19522227],
       ...,
       [ 2.91456775],
       [-1.65102466],
       [ 0.99528168]])
```

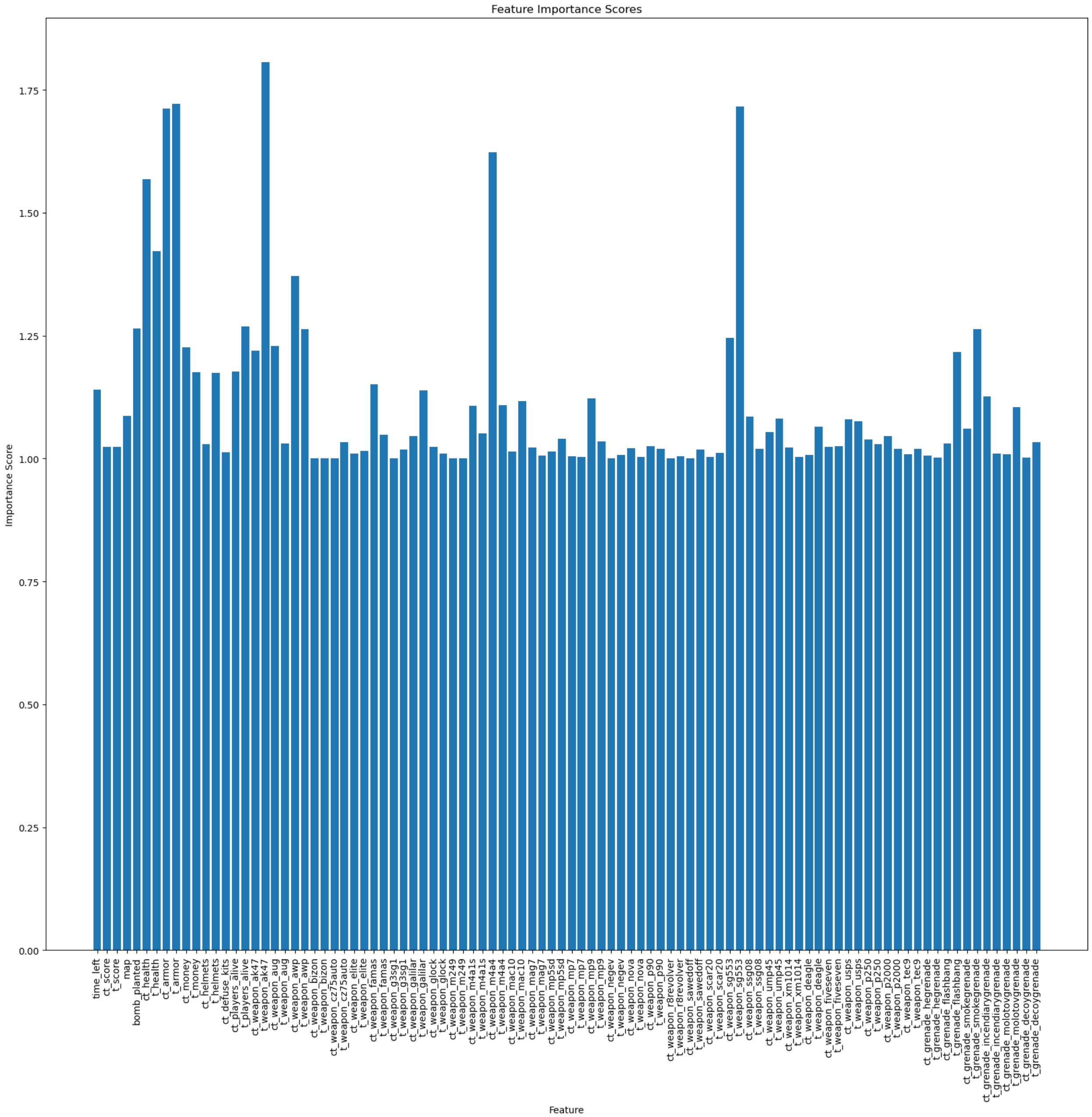
```
# Obtaining the LDA coefficients.This will give the importance scores
associated with each feature.
lda_coefficients=np.exp(np.abs(lda.coef_))
lda_coefficients= lda_coefficients.flatten()
lda_coefficients
```

```
array([1.14051375, 1.02348232, 1.02361012, 1.08711166, 1.26443741,
       1.56896428, 1.42226149, 1.71174795, 1.72149378, 1.22695841,
       1.17629264, 1.02925519, 1.17393359, 1.01233178, 1.17701202,
       1.26888831, 1.21975251, 1.80580698, 1.22872837, 1.03124048,
       1.37208174, 1.26351299, 1.        , 1.00130014, 1.00099319,
       1.03406233, 1.01079854, 1.01589366, 1.1511331 , 1.0481572 ,
       1.        , 1.01891369, 1.04559279, 1.13912974, 1.02386319,
       1.01009894, 1.        , 1.        , 1.10758456, 1.051198  ,
       1.62280372, 1.10928011, 1.01408145, 1.11733322, 1.02306813,
       1.00594113, 1.01383628, 1.04016588, 1.00546389, 1.00296978,
       1.1231047 , 1.0350133 , 1.        , 1.00757676, 1.02090408,
       1.00405543, 1.0252188 , 1.01962772, 1.        , 1.00493109,
       1.        , 1.01827676, 1.00401758, 1.01179667, 1.24546089,
       1.71649302, 1.08595734, 1.01928052, 1.05378886, 1.0817296 ,
       1.0223457 , 1.00392739, 1.00787732, 1.06560713, 1.02458755,
       1.02585421, 1.07987292, 1.07637588, 1.03850144, 1.02942838,
       1.0461802 , 1.02050308, 1.0093683 , 1.02047298, 1.00675662,
       1.0023777 , 1.03143135, 1.21721335, 1.06075101, 1.26371818,
       1.12626069, 1.01006317, 1.00852246, 1.10424102, 1.00231651,
       1.03293193])
```

```
num_features=X.shape[1]
feature_indices=np.arange(num_features)
feature_indices

feature_names=list(X.columns)

plt.figure(figsize=(20,18))
plt.bar(feature_indices,lda_coefficients)
plt.xticks(feature_indices,feature_names,rotation="vertical")
plt.xlabel('Feature')
plt.ylabel('Importance Score')
plt.title('Feature Importance Scores')
plt.show()
```



```
df_feature_score=pd.DataFrame({"Feature_names":feature_names,"feature_scores":lda_coefficients})

top_20_values=df_feature_score.nlargest(20,'feature_scores')
top_20_values.head(20)
```

```
          Feature_names  feature_scores
17        t_weapon_ak47        1.805807
8               t_armor        1.721494
65       t_weapon_sg553        1.716493
7              ct_armor        1.711748
40       ct_weapon_m4a4        1.622804
```

```
5                 ct_health        1.568964
6                  t_health        1.422261
20            ct_weapon_awp        1.372082
15          t_players_alive        1.268888
4              bomb_planted        1.264437
89   t_grenade_smokegrenade        1.263718
21             t_weapon_awp        1.263513
64          ct_weapon_sg553        1.245461
18            ct_weapon_aug        1.228728
9                  ct_money        1.226958
16           ct_weapon_ak47        1.219753
87      t_grenade_flashbang        1.217213
14         ct_players_alive        1.177012
10                  t_money        1.176293
12                t_helmets        1.173934
```

```
top_20_values.index
```

```
Int64Index([17, 8, 65, 7, 40, 5, 6, 20, 15, 4, 89, 21, 64, 18, 9, 16,
87, 14,
            10, 12],
           dtype='int64')
```

```
x_train=x_train[:,[17, 8, 65, 7, 40, 5, 6, 20, 15, 4, 89, 21, 64, 18,
9, 16, 87, 14,
            10, 12]]
```

```
x_test=x_test[:,[17, 8, 65, 7, 40, 5, 6, 20, 15, 4, 89, 21, 64, 18, 9,
16, 87, 14,
            10, 12]]
```

```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
```

```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\sklearn\utils\
validation.py:1183: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Evolution of Model

```
from sklearn.metrics import accuracy_score,classification_report
accuracy_score(y_test,y_pred)
```

```
0.7516951229474717
```

```
classification_report(y_test,y_pred)
```

```
'              precision    recall  f1-score   support\n\n           0
0.74      0.76      0.75     12004\n           1       0.77      0.74
0.75     12478\n\n    accuracy                           0.75
24482\n   macro avg       0.75      0.75      0.75     24482\nweighted
avg       0.75      0.75      0.75     24482\n'
```

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
dtc.fit(x_train,y_train)
y_pred=dtc.predict(x_test)
```

```
accuracy_score(y_test,y_pred)
```

```
0.8145984805162977
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.81      0.81     12004
           1       0.82      0.82      0.82     12478

    accuracy                           0.81     24482
   macro avg       0.81      0.81      0.81     24482
weighted avg       0.81      0.81      0.81     24482
```

```
# Random Forest
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred=rfc.predict(x_test)
```

```
C:\Users\Arigala.Adarsh\anaconda3\lib\site-packages\sklearn\
base.py:1152: DataConversionWarning: A column-vector y was passed when
a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

```
accuracy_score(y_test,y_pred)
```

```
0.8559758189690385
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.86      0.85     12004
           1       0.86      0.86      0.86     12478
```

```
    accuracy                           0.86     24482
   macro avg       0.86      0.86      0.86     24482
weighted avg       0.86      0.86      0.86     24482
```