```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remoun
```

```
path = '/content/drive/MyDrive/Round Winner Prediction/csgo_round_snapshots.csv'
```

```
df = pd.read_csv(path)
```

```
df.head()
```

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...	t_grenade_fla
0	175.00	0.0	0.0	de_dust2	False	500.0	500.0	0.0	0.0	4000.0	...	
1	156.03	0.0	0.0	de_dust2	False	500.0	500.0	400.0	300.0	600.0	...	
2	96.03	0.0	0.0	de_dust2	False	391.0	400.0	294.0	200.0	750.0	...	
3	76.03	0.0	0.0	de_dust2	False	391.0	400.0	294.0	200.0	750.0	...	
4	174.97	1.0	0.0	de_dust2	False	500.0	500.0	192.0	0.0	18350.0	...	

5 rows × 97 columns

```
df.info()
```

```
 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
 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.columns
```

```
Index(['time_left', 'ct_score', 't_score', 'map', 'bomb_planted', 'ct_health',
       't_health', 'ct_armor', 't_armor', 'ct_money', 't_money', 'ct_helmets',
       't_helmets', 'ct_defuse_kits', 'ct_players_alive', 't_players_alive',
       'ct_weapon_ak47', 't_weapon_ak47', 'ct_weapon_aug', 't_weapon_aug',
       'ct_weapon_awp', 't_weapon_awp', 'ct_weapon_bizon', 't_weapon_bizon',
       'ct_weapon_cz75auto', 't_weapon_cz75auto', 'ct_weapon_elite',
       't_weapon_elite', 'ct_weapon_famas', 't_weapon_famas',
       'ct_weapon_g3sg1', 't_weapon_g3sg1', 'ct_weapon_galilar',
       't_weapon_galilar', 'ct_weapon_glock', 't_weapon_glock',
       'ct_weapon_m249', 't_weapon_m249', 'ct_weapon_m4a1s', 't_weapon_m4a1s',
       'ct_weapon_m4a4', 't_weapon_m4a4', 'ct_weapon_mac10', 't_weapon_mac10',
       'ct_weapon_mag7', 't_weapon_mag7', 'ct_weapon_mp5sd', 't_weapon_mp5sd',
       'ct_weapon_mp7', 't_weapon_mp7', 'ct_weapon_mp9', 't_weapon_mp9',
       'ct_weapon_negev', 't_weapon_negev', 'ct_weapon_nova', 't_weapon_nova',
       'ct_weapon_p90', 't_weapon_p90', 'ct_weapon_r8revolver',
       't_weapon_r8revolver', 'ct_weapon_sawedoff', 't_weapon_sawedoff',
       'ct_weapon_scar20', 't_weapon_scar20', 'ct_weapon_sg553',
       't_weapon_sg553', 'ct_weapon_ssg08', 't_weapon_ssg08',
       'ct_weapon_ump45', 't_weapon_ump45', 'ct_weapon_xm1014',
       't_weapon_xm1014', 'ct_weapon_deagle', 't_weapon_deagle',
       'ct_weapon_fiveseven', 't_weapon_fiveseven', 'ct_weapon_usps',
       't_weapon_usps', 'ct_weapon_p250', 't_weapon_p250', 'ct_weapon_p2000',
       't_weapon_p2000', 'ct_weapon_tec9', 't_weapon_tec9',
       'ct_grenade_hegrenade', 't_grenade_hegrenade', 'ct_grenade_flashbang',
       't_grenade_flashbang', 'ct_grenade_smokegrenade',
       't_grenade_smokegrenade', 'ct_grenade_incendiarygrenade',
       't_grenade_incendiarygrenade', 'ct_grenade_molotovgrenade',
       't_grenade_molotovgrenade', 'ct_grenade_decoygrenade',
       't_grenade_decoygrenade', 'round_winner'],
      dtype='object')
```

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

```
np.int64(0)
```

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

```
np.int64(4962)
```

```
df.drop_duplicates(inplace=True)
```

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

```
np.int64(0)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 117448 entries, 0 to 122409
Data columns (total 97 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   time_left                     117448 non-null  float64
 1   ct_score                      117448 non-null  float64
 2   t_score                       117448 non-null  float64
 3   map                           117448 non-null  object
 4   bomb_planted                  117448 non-null  bool
 5   ct_health                     117448 non-null  float64
 6   t_health                      117448 non-null  float64
 7   ct_armor                      117448 non-null  float64
 8   t_armor                       117448 non-null  float64
 9   ct_money                      117448 non-null  float64
 10  t_money                       117448 non-null  float64
 11  ct_helmets                    117448 non-null  float64
 12  t_helmets                     117448 non-null  float64
 13  ct_defuse_kits                117448 non-null  float64
 14  ct_players_alive              117448 non-null  float64
 15  t_players_alive               117448 non-null  float64
 16  ct_weapon_ak47                117448 non-null  float64
 17  t_weapon_ak47                 117448 non-null  float64
 18  ct_weapon_aug                 117448 non-null  float64
 19  t_weapon_aug                  117448 non-null  float64
 20  ct_weapon_awp                 117448 non-null  float64
 21  t_weapon_awp                  117448 non-null  float64
```

```
     22  ct_weapon_bizon               117448 non-null  float64
     23  t_weapon_bizon                117448 non-null  float64
     24  ct_weapon_cz75auto            117448 non-null  float64
     25  t_weapon_cz75auto             117448 non-null  float64
     26  ct_weapon_elite               117448 non-null  float64
     27  t_weapon_elite                117448 non-null  float64
     28  ct_weapon_famas               117448 non-null  float64
     29  t_weapon_famas                117448 non-null  float64
     30  ct_weapon_g3sg1               117448 non-null  float64
     31  t_weapon_g3sg1                117448 non-null  float64
     32  ct_weapon_galilar             117448 non-null  float64
     33  t_weapon_galilar              117448 non-null  float64
     34  ct_weapon_glock               117448 non-null  float64
     35  t_weapon_glock                117448 non-null  float64
     36  ct_weapon_m249                117448 non-null  float64
     37  t_weapon_m249                 117448 non-null  float64
     38  ct_weapon_m4a1s               117448 non-null  float64
     39  t_weapon_m4a1s                117448 non-null  float64
     40  ct_weapon_m4a4                117448 non-null  float64
     41  t_weapon_m4a4                 117448 non-null  float64
     42  ct_weapon_mac10               117448 non-null  float64
     43  t_weapon_mac10                117448 non-null  float64
     44  ct_weapon_mag7                117448 non-null  float64
     45  t_weapon_mag7                 117448 non-null  float64
     46  ct_weapon_mp5sd               117448 non-null  float64
     47  t_weapon_mp5sd                117448 non-null  float64
     48  ct_weapon_mp7                 117448 non-null  float64
     49  t_weapon_mp7                  117448 non-null  float64
     50  ct_weapon_mp9                 117448 non-null  float64
     51  t_weapon_mp9                  117448 non-null  float64
     52  ct_weapon_negev               117448 non-null  float64
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()


for k in df.columns:
  if df[k].dtype=='object' or df[k].dtype =='bool':
    df[k]=le.fit_transform(df[k])


# 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'])


df.info()
```

```
 83  t_weapon_tec9                117448 non-null  float64
 84  ct_grenade_hegrenade         117448 non-null  float64
 85  t_grenade_hegrenade          117448 non-null  float64
 86  ct_grenade_flashbang         117448 non-null  float64
 87  t_grenade_flashbang          117448 non-null  float64
 88  ct_grenade_smokegrenade      117448 non-null  float64
 89  t_grenade_smokegrenade       117448 non-null  float64
 90  ct_grenade_incendiarygrenade  117448 non-null  float64
 91  t_grenade_incendiarygrenade  117448 non-null  float64
 92  ct_grenade_molotovgrenade    117448 non-null  float64
 93  t_grenade_molotovgrenade     117448 non-null  float64
 94  ct_grenade_decoygrenade      117448 non-null  float64
 95  t_grenade_decoygrenade       117448 non-null  float64
 96  round_winner                 117448 non-null  int64
dtypes: float64(94), int64(3)
memory usage: 87.8 MB
```

```
# data is splited here in terms of dependent and independent feature
X = df.iloc[:,:-1]
y = df['round_winner']
```

```
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, random_state=34)
```

```
X_train
```

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...	ct_grenade_f
46655	169.96	7.0	0.0	3	0	500.0	500.0	284.0	0.0	45850.0	...	
4931	114.92	2.0	2.0	2	0	500.0	500.0	400.0	481.0	300.0	...	
61390	14.80	11.0	12.0	4	1	100.0	276.0	100.0	198.0	200.0	...	
23759	109.93	2.0	8.0	3	0	500.0	500.0	500.0	487.0	2950.0	...	
25860	174.93	18.0	16.0	7	0	500.0	500.0	200.0	0.0	66000.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
97402	110.05	2.0	9.0	1	0	500.0	500.0	500.0	459.0	1200.0	...	
23141	114.91	8.0	10.0	6	0	500.0	500.0	196.0	442.0	10500.0	...	
44918	109.89	13.0	9.0	5	0	500.0	500.0	500.0	443.0	350.0	...	
107644	94.95	13.0	14.0	4	0	500.0	500.0	500.0	491.0	950.0	...	
79686	174.96	9.0	7.0	1	0	500.0	500.0	0.0	287.0	10700.0	...	

93958 rows × 96 columns

```
X_test
```

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...	ct_grenade_f
86634	94.95	6.0	3.0	4	0	400.0	460.0	371.0	286.0	21400.0	...	
29985	34.80	8.0	9.0	4	0	100.0	68.0	93.0	100.0	4100.0	...	
3264	74.95	6.0	5.0	2	0	413.0	308.0	461.0	266.0	21650.0	...	
51472	94.91	6.0	10.0	5	0	500.0	500.0	500.0	500.0	1850.0	...	
44943	14.95	0.0	0.0	4	0	42.0	40.0	62.0	68.0	750.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
40083	94.95	0.0	0.0	3	0	479.0	486.0	493.0	292.0	750.0	...	
65879	94.93	10.0	12.0	2	0	500.0	500.0	462.0	500.0	29150.0	...	
102343	34.93	8.0	2.0	3	0	346.0	100.0	440.0	100.0	19650.0	...	
25198	175.00	12.0	9.0	7	0	500.0	500.0	298.0	100.0	35850.0	...	
32293	38.27	0.0	1.0	2	1	236.0	339.0	100.0	455.0	1150.0	...	

23490 rows × 96 columns

```
y_train
```

	round_winner
46655	0
4931	1
61390	1
23759	0
25860	0
...	...
97402	0
23141	1
44918	0
107644	1
79686	1

93958 rows × 1 columns

dtype: int64

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
X_train
```

```
array([[ 1.41686578,  0.04867334, -1.40859784, ..., -0.8280369 ,
        -0.1628247 , -0.15382351],
       [ 0.38227138, -0.99065426, -0.99586095, ...,  2.16418228,
        -0.1628247 , -0.15382351],
       [-1.49969795,  0.88013541,  1.06782348, ..., -0.8280369 ,
        -0.1628247 , -0.15382351],
       ...,
       [ 0.28772178,  1.29586645,  0.44871815, ...,  2.16418228,
        -0.1628247 , -0.15382351],
       [ 0.00689256,  1.29586645,  1.48056037, ...,  1.56573845,
        -0.1628247 , -0.15382351],
       [ 1.51085147,  0.46440437,  0.03598127, ..., -0.8280369 ,
        -0.1628247 , -0.15382351]])
```

```
X_test
```

```
array([[ 0.00689256, -0.15919218, -0.78949251, ..., -0.8280369 ,
        -0.1628247 , -0.15382351],
       [-1.12375522,  0.25653885,  0.44871815, ..., -0.8280369 ,
        -0.1628247 , -0.15382351],
       [-0.36905018, -0.15919218, -0.37675562, ..., -0.22959306,
        -0.1628247 , -0.15382351],
       ...,
       [-1.12131159,  0.25653885, -0.99586095, ..., -0.8280369 ,
        -0.1628247 , -0.15382351],
       [ 1.51160335,  1.08800093,  0.44871815, ..., -0.22959306,
        -0.1628247 , -0.15382351],
       [-1.05852915, -1.40638529, -1.20222939, ..., -0.8280369 ,
        -0.1628247 , -0.15382351]])
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
lda.fit(X_train, y_train)
```

```
▾ LinearDiscriminantAnalysis ⓘ ?
LinearDiscriminantAnalysis()
```

```
lda.transform(X_test)
```

```
array([[-0.81926606],
       [-0.20646063],
       [-1.2027105 ],
       ...,
       [-3.11199387],
       [-0.58398431],
       [ 3.15241551]])
```

```
lda.coef_
```

```
array([[ 1.40798218e-01, -1.63779539e-02,  1.65392901e-02,
        -8.44119919e-02,  2.45841039e-01, -4.25778801e-01,
         3.25206002e-01, -6.05044865e-01,  6.42755761e-01,
        -2.14088433e-01,  1.63286872e-01,  4.99608671e-02,
         7.13517526e-02, -3.50006793e-02, -2.23093635e-01,
         3.81277147e-01, -1.92314735e-01,  6.25316797e-01,
        -2.12493284e-01,  3.78495199e-02, -3.11983894e-01,
         2.44962696e-01, -2.18621069e-16, -1.56787590e-03,
         1.38840808e-02,  2.70750360e-04, -8.16982190e-03,
         1.45702604e-02, -1.41816668e-01,  4.89809549e-02,
         5.76767164e-16,  2.09674722e-02, -5.31379540e-02,
         1.37751078e-01,  1.92835708e-02, -1.71303471e-01,
         1.14972095e-02, -1.72407505e-15, -9.06436363e-02,
         4.98581963e-02, -4.67518881e-01,  1.13211531e-01,
        -1.40641808e-02,  1.08913917e-01, -1.83591480e-02,
         4.22382388e-03, -1.03658699e-02,  3.68348304e-02,
         7.66327721e-03, -1.10885542e-02, -1.13307614e-01,
         3.97381250e-02, -5.85499953e-16,  9.44560366e-16,
        -1.27126380e-02, -2.51179351e-02, -2.58388323e-02,
         1.70787449e-02,  1.54680513e-15, -6.42861996e-03,
         1.41525137e-15,  1.43328160e-02,  8.33175815e-03,
         1.49108163e-15, -2.08865481e-01,  5.68169961e-01,
        -8.25881264e-02,  1.81067895e-02, -4.30754793e-02,
         9.39543415e-02, -2.03907205e-02,  2.32238959e-03,
         2.69303940e-02, -1.49312297e-02,  3.37016610e-02,
        -3.69002175e-02,  1.24240245e-01,  3.58209412e-02,
         5.63204531e-02, -1.94758378e-02,  5.69087054e-02,
         7.31847564e-03, -4.46916184e-03,  1.96270505e-03,
        -9.92870062e-03, -2.55954486e-03,  1.71511066e-02,
        -1.92516460e-01,  8.11853597e-02, -2.30364069e-01,
         9.48200884e-02, -1.43216491e-02,  1.10580825e-02,
        -1.07292562e-01, -1.34194642e-04,  1.82069910e-02]])
```

```
lda_coef = np.exp(np.abs(lda.coef_))
lda_coef
```

```
array([[1.15119233, 1.01651281, 1.01667682, 1.08807708, 1.27869629,
        1.53078213, 1.38431579, 1.83133437, 1.90171433, 1.23873219,
        1.1773744 , 1.05122996, 1.07395893, 1.03562041, 1.24993761,
        1.46415333, 1.21205193, 1.86883791, 1.23675781, 1.03857494,
        1.36613269, 1.27757365, 1.        , 1.00156911, 1.01398091,
        1.00027079, 1.00820329, 1.01467692, 1.15236536, 1.05020035,
        1.        , 1.02118883, 1.05457512, 1.14768983, 1.0194707 ,
        1.18685087, 1.01156356, 1.        , 1.09487876, 1.05112203,
        1.59602934, 1.1198688 , 1.01416355, 1.11506636, 1.01852871,
        1.00423276, 1.01041978, 1.03752164, 1.00769272, 1.01115026,
        1.1199764 , 1.04053825, 1.        , 1.        , 1.01279379,
        1.02543605, 1.02617555, 1.01722542, 1.        , 1.00644933,
        1.        , 1.01443602, 1.00836656, 1.        , 1.23227922,
        1.76503401, 1.08609438, 1.01827171, 1.04401669, 1.09850959,
        1.02060003, 1.00232509, 1.02729629, 1.01504326, 1.034276  ,
        1.03758948, 1.13228786, 1.03647024, 1.05793665, 1.01966673,
        1.05855917, 1.00734532, 1.00447916, 1.00196463, 1.00997815,
        1.00256282, 1.01729903, 1.21229646, 1.08457191, 1.25905831,
        1.09946103, 1.0144247 , 1.01111945, 1.1132599 , 1.0001342 ,
        1.01837375]])
```

```
lda_coef=lda_coef.flatten()  #used to convert any dimention array into 1d array
lda_coef
```

```
array([1.15119233, 1.01651281, 1.01667682, 1.08807708, 1.27869629,
       1.53078213, 1.38431579, 1.83133437, 1.90171433, 1.23873219,
       1.1773744 , 1.05122996, 1.07395893, 1.03562041, 1.24993761,
       1.46415333, 1.21205193, 1.86883791, 1.23675781, 1.03857494,
       1.36613269, 1.27757365, 1.        , 1.00156911, 1.01398091,
       1.00027079, 1.00820329, 1.01467692, 1.15236536, 1.05020035,
       1.        , 1.02118883, 1.05457512, 1.14768983, 1.0194707 ,
       1.18685087, 1.01156356, 1.        , 1.09487876, 1.05112203,
       1.59602934, 1.1198688 , 1.01416355, 1.11506636, 1.01852871,
       1.00423276, 1.01041978, 1.03752164, 1.00769272, 1.01115026,
       1.1199764 , 1.04053825, 1.        , 1.        , 1.01279379,
       1.02543605, 1.02617555, 1.01722542, 1.        , 1.00644933,
       1.        , 1.01443602, 1.00836656, 1.        , 1.23227922,
       1.76503401, 1.08609438, 1.01827171, 1.04401669, 1.09850959,
       1.02060003, 1.00232509, 1.02729629, 1.01504326, 1.034276  ,
       1.03758948, 1.13228786, 1.03647024, 1.05793665, 1.01966673,
       1.05855917, 1.00734532, 1.00447916, 1.00196463, 1.00997815,
       1.00256282, 1.01729903, 1.21229646, 1.08457191, 1.25905831,
       1.09946103, 1.0144247 , 1.01111945, 1.1132599 , 1.0001342 ,
       1.01837375])
```

```
feature_names=X.columns
feature_names
```

```
Index(['time_left', 'ct_score', 't_score', 'map', 'bomb_planted', 'ct_health',
       't_health', 'ct_armor', 't_armor', 'ct_money', 't_money', 'ct_helmets',
       't_helmets', 'ct_defuse_kits', 'ct_players_alive', 't_players_alive',
       'ct_weapon_ak47', 't_weapon_ak47', 'ct_weapon_aug', 't_weapon_aug',
       'ct_weapon_awp', 't_weapon_awp', 'ct_weapon_bizon', 't_weapon_bizon',
       'ct_weapon_cz75auto', 't_weapon_cz75auto', 'ct_weapon_elite',
       't_weapon_elite', 'ct_weapon_famas', 't_weapon_famas',
       'ct_weapon_g3sg1', 't_weapon_g3sg1', 'ct_weapon_galilar',
       't_weapon_galilar', 'ct_weapon_glock', 't_weapon_glock',
       'ct_weapon_m249', 't_weapon_m249', 'ct_weapon_m4a1s', 't_weapon_m4a1s',
       'ct_weapon_m4a4', 't_weapon_m4a4', 'ct_weapon_mac10', 't_weapon_mac10',
       'ct_weapon_mag7', 't_weapon_mag7', 'ct_weapon_mp5sd', 't_weapon_mp5sd',
       'ct_weapon_mp7', 't_weapon_mp7', 'ct_weapon_mp9', 't_weapon_mp9',
       'ct_weapon_negev', 't_weapon_negev', 'ct_weapon_nova', 't_weapon_nova',
       'ct_weapon_p90', 't_weapon_p90', 'ct_weapon_r8revolver',
       't_weapon_r8revolver', 'ct_weapon_sawedoff', 't_weapon_sawedoff',
       'ct_weapon_scar20', 't_weapon_scar20', 'ct_weapon_sg553',
       't_weapon_sg553', 'ct_weapon_ssg08', 't_weapon_ssg08',
       'ct_weapon_ump45', 't_weapon_ump45', 'ct_weapon_xm1014',
       't_weapon_xm1014', 'ct_weapon_deagle', 't_weapon_deagle',
       'ct_weapon_fiveseven', 't_weapon_fiveseven', 'ct_weapon_usps',
       't_weapon_usps', 'ct_weapon_p250', 't_weapon_p250', 'ct_weapon_p2000',
       't_weapon_p2000', 'ct_weapon_tec9', 't_weapon_tec9',
       'ct_grenade_hegrenade', 't_grenade_hegrenade', 'ct_grenade_flashbang',
       't_grenade_flashbang', 'ct_grenade_smokegrenade',
       't_grenade_smokegrenade', 'ct_grenade_incendiarygrenade',
       't_grenade_incendiarygrenade', 'ct_grenade_molotovgrenade',
       't_grenade_molotovgrenade', 'ct_grenade_decoygrenade',
       't_grenade_decoygrenade'],
      dtype='object')
```

```
df_features_imp =pd.DataFrame({'Features':feature_names,'imp_value':lda_coef})
df_features_imp
```

	Features	imp_value
0	time_left	1.151192
1	ct_score	1.016513
2	t_score	1.016677
3	map	1.088077
4	bomb_planted	1.278696
...	...	...
91	t_grenade_incendiarygrenade	1.014425
92	ct_grenade_molotovgrenade	1.011119
93	t_grenade_molotovgrenade	1.113260
94	ct_grenade_decoygrenade	1.000134
95	t_grenade_decoygrenade	1.018374

96 rows × 2 columns

Next steps: Generate code with df_features_imp | View recommended plots | New interactive sheet

```
top_20_fea=df_features_imp.nlargest(20,'imp_value')
#nlargest(n,colname) -->used to get top n vaalues based on colname
top_20_fea
```

	Features	imp_value
8	t_armor	1.901714
17	t_weapon_ak47	1.868838
7	ct_armor	1.831334
65	t_weapon_sg553	1.765034
40	ct_weapon_m4a4	1.596029
5	ct_health	1.530782
15	t_players_alive	1.464153
6	t_health	1.384316
20	ct_weapon_awp	1.366133
4	bomb_planted	1.278696
21	t_weapon_awp	1.277574
89	t_grenade_smokegrenade	1.259058
14	ct_players_alive	1.249938
9	ct_money	1.238732
18	ct_weapon_aug	1.236758
64	ct_weapon_sg553	1.232279
87	t_grenade_flashbang	1.212296
16	ct_weapon_ak47	1.212052
35	t_weapon_glock	1.186851
10	t_money	1.177374

Next steps: Generate code with top_20_fea | View recommended plots | New interactive sheet

```
imp_col=top_20_fea.index
imp_col
```

```
Index([8, 17, 7, 65, 40, 5, 15, 6, 20, 4, 21, 89, 14, 9, 18, 64, 87, 16, 35,
       10],
      dtype='int64')
```

```
X_train[:,[8,17,7]]
```

```
array([[-1.73825231, -0.94029438, -0.19849244],
       [ 1.02521179,  1.2541644 ,  0.48371375],
       [-0.60069328, -0.94029438, -1.28061261],
       ...,
       [ 0.80689238,  1.2541644 ,  1.07182254],
       [ 1.08266427,  0.52267814,  1.07182254],
       [-0.08936625, -0.94029438, -1.86872139]])
```

```
X_train = X_train[:,imp_col]
X_train
```

```
array([[-1.73825231, -0.94029438, -0.19849244, ...,  1.2210558 ,
         1.06731306,  0.50792636],
       [ 1.02521179,  1.2541644 ,  0.48371375, ..., -0.47725904,
         1.06731306, -0.68489092],
       [-0.60069328, -0.94029438, -1.28061261, ...,  1.2210558 ,
        -1.99263947, -0.40302656],
       ...,
       [ 0.80689238,  1.2541644 ,  1.07182254, ..., -0.47725904,
         1.06731306,  0.94093538],
       [ 1.08266427,  0.52267814,  1.07182254, ..., -0.47725904,
         1.06731306, -0.47655639],
       [-0.08936625, -0.94029438, -1.86872139, ..., -0.47725904,
        -0.15666795,  0.56103124]])
```

```
X_test = X_test[:,imp_col]
X_test
```

```
array([[-0.09511149, -0.94029438,  0.3131622 , ..., -0.47725904,
        -1.99263947, -0.00678246],
       [-1.16372755, -0.20880812, -1.32178022, ..., -0.47725904,
        -1.38064897, -0.92590537],
       [-0.21001645,  0.52267814,  0.84246011, ...,  1.2210558 ,
         0.45532255, -0.92590537],
       ...,
       [-1.16372755, -0.94029438,  0.71895726, ...,  2.91937065,
        -1.99263947, -0.67263595],
       [-1.16372755, -0.94029438, -0.11615721, ...,  1.2210558 ,
         1.06731306,  1.12884495],
       [ 0.87583536, -0.20880812, -1.28061261, ..., -0.47725904,
        -0.76865846, -0.80335565]])
```

Logistic regression -75

```
from sklearn.linear_model import LogisticRegression
model_lg=LogisticRegression()
model_lg.fit(X_train,y_train)
```

```
▾ LogisticRegression ⓘ ?
LogisticRegression()
```

```
y_pred=model_lg.predict(X_test)
```

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

```
0.7530438484461472
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[8682, 2818],
       [2983, 9007]])
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
model_dt=DecisionTreeClassifier()
model_dt.fit(X_train,y_train)
```

```
▾ DecisionTreeClassifier ⓘ ?
DecisionTreeClassifier()
```

```
y_pred=model_dt.predict(X_test)
```

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

```
0.8109408258833546
```

Random Forest -->85

```
from sklearn.ensemble import RandomForestClassifier
model_rf=RandomForestClassifier()
model_rf.fit(X_train,y_train)
```

```
▾ RandomForestClassifier ⓘ ?
RandomForestClassifier()
```

```
y_pred=model_rf.predict(X_test)
```

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

```
0.8536398467432951
```