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

```
data=pd.read_csv("/content/sample_data/csgo_round_snapshots (1).csv")
```

```
data.head()
```

```
   time_left  ct_score  t_score      map  bomb_planted  ct_health  t_health  ct_armor  t_armor  ct_money  ...  t_grenade_flashbang  ct_gr
0     175.00       0.0      0.0  de_dust2         False      500.0     500.0       0.0      0.0    4000.0  ...                  0.0
1     156.03       0.0      0.0  de_dust2         False      500.0     500.0     400.0    300.0     600.0  ...                  0.0
2      96.03       0.0      0.0  de_dust2         False      391.0     400.0     294.0    200.0     750.0  ...                  0.0
3      76.03       0.0      0.0  de_dust2         False      391.0     400.0     294.0    200.0     750.0  ...                  0.0
4     174.97       1.0      0.0  de_dust2         False      500.0     500.0     192.0      0.0   18350.0  ...                  0.0
5 rows × 97 columns
```

```
data.shape
```

```
(122410, 97)
```

```
data.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
```

```
data.isna().sum().sum()
```

```
0
```

```
data.round_winner.value_counts()
```

	count
round_winner	
T	62406
CT	60004

```
dtype: int64
```

```
counts=data['map'].value_counts()
```

	count
map	
2	23811
1	22144
4	19025
3	18576
5	14081
6	13491
7	11137
0	145

```
dtype: int64
```

```
total=sum(counts)
total
```

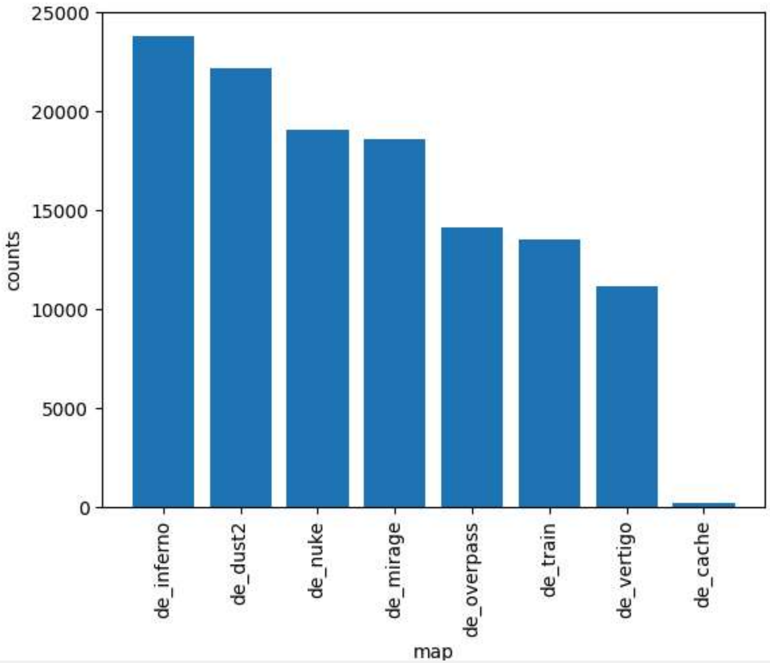
```
122410
```

```
percentage=round((counts/total)*100,2)
percentage
```

	count
map	
de_inferno	19.45
de_dust2	18.09
de_nuke	15.54
de_mirage	15.18
de_overpass	11.50
de_train	11.02
de_vertigo	9.10
de_cache	0.12

```
dtype: float64
```

```
plt.bar(counts.index,counts.values)
plt.xticks(rotation=90)
plt.xlabel('map')
plt.ylabel('counts')
plt.show()
```



```
# label encoder
```

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
data['map']=le.fit_transform(data['map'])
data['bomb_planted']=le.fit_transform(data['bomb_planted'])
data['round_winner']=le.fit_transform(data['round_winner'])
```

```
x=data.drop(columns=['round_winner'])
y=data['round_winner']
```

```
x
```

	time_left	ct_score	t_score	map	bomb_planted	ct_health	t_health	ct_armor	t_armor	ct_money	...	ct_grenade_flashbang	t_g
0	175.00	0.0	0.0	1	0	500.0	500.0	0.0	0.0	4000.0	...	0.0	
1	156.03	0.0	0.0	1	0	500.0	500.0	400.0	300.0	600.0	...	0.0	
2	96.03	0.0	0.0	1	0	391.0	400.0	294.0	200.0	750.0	...	0.0	
3	76.03	0.0	0.0	1	0	391.0	400.0	294.0	200.0	750.0	...	0.0	
4	174.97	1.0	0.0	1	0	500.0	500.0	192.0	0.0	18350.0	...	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
122405	15.41	11.0	14.0	6	1	200.0	242.0	195.0	359.0	100.0	...	1.0	
122406	174.93	11.0	15.0	6	0	500.0	500.0	95.0	175.0	11500.0	...	1.0	
122407	114.93	11.0	15.0	6	0	500.0	500.0	495.0	475.0	1200.0	...	4.0	
122408	94.93	11.0	15.0	6	0	500.0	500.0	495.0	475.0	1200.0	...	1.0	
122409	74.93	11.0	15.0	6	0	375.0	479.0	395.(	[illegible]	[illegible]	[illegible]	[illegible]	

122410 rows × 96 columns

```
y
```

```
        round_winner
     0             0
     1             0
     2             0
     3             0
     4             0
   ...           ...
122405             1
122406             1
122407             1
122408             1
122409             1
122410 rows × 1 columns

dtype: int64
```

```
from sklearn.model_selection import train_test_split
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=42)
```

```
xtrain.shape
```

```
(97928, 96)
```

```
from sklearn.preprocessing import StandardScaler
```

```
sc=StandardScaler()
```

```
xtrain=sc.fit_transform(xtrain)
xtest=sc.transform(xtest)
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
lda=LinearDiscriminantAnalysis()
```

```
lda.fit(xtrain,ytrain)
lda.transform(xtest)
```

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

```
lda.coef_
```

```
array([[ 1.31478821e-01, -2.32108474e-02,  2.33357149e-02,
        -8.35243221e-02,  2.34627291e-01, -4.50415709e-01,
         3.52248201e-01, -5.37515044e-01,  5.43192391e-01,
        -2.04538266e-01,  1.62367665e-01,  2.88354236e-02,
         1.60360148e-01, -1.22563615e-02, -1.62979039e-01,
         2.38141174e-01, -1.98647978e-01,  5.91007575e-01,
        -2.05979791e-01,  3.07624231e-02, -3.16329105e-01,
         2.33895928e-01,  1.49471699e-16,  1.29929334e-03,
         9.92695503e-04,  3.34950500e-02, -1.07406492e-02,
         1.57686764e-02, -1.40746762e-01,  4.70335749e-02,
        -1.21204873e-16,  1.87370526e-02, -4.45839869e-02,
         1.30264587e-01,  2.35829139e-02, -1.00482852e-02,
         6.18270834e-16,  6.70006739e-16, -1.02181571e-01,
         4.99304639e-02, -4.84155344e-01,  1.03711254e-01,
        -1.39832256e-02,  1.10944795e-01, -2.28060850e-02,
         5.92355109e-03, -1.37414356e-02,  3.93801978e-02,
         5.44902094e-03, -2.96537404e-03, -1.16096906e-01,
         3.44142810e-02,  3.11201301e-17, -7.54819910e-03,
        -2.06885892e-02, -4.04722536e-03, -2.49060574e-02,
         1.94375837e-02, -1.83741826e-16, -4.91896923e-03,
        -1.74982761e-16,  1.81117490e-02,  4.00952990e-03,
        -1.17276337e-02, -2.19505655e-01,  5.40283270e-01,
        -8.24619420e-02,  1.90970034e-02, -5.23921112e-02,
         7.85612409e-02, -2.20996974e-02, -3.91970104e-03,
        -7.84645185e-03,  6.35447141e-02,  2.42901449e-02,
        -2.55256370e-02,  7.68433721e-02,  7.35997283e-02,
         3.77787467e-02,  2.90036746e-02,  4.51456248e-02,
         2.02957226e-02, -9.32469259e-03,  2.02662240e-02,
        -6.73389335e-03,  2.37488231e-03,  3.09474977e-02,
        -1.96564105e-01,  5.89771532e-02, -2.34058310e-01,
         1.18903021e-01, -1.00128731e-02,  8.48634510e-03,
        -9.91582374e-02, -2.31382826e-03,  3.24012912e-02]])
```

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

```
lda_coefficient
```

```
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]])
```

```
lda_coefficient=lda_coefficient.flatten()
```

```
lda_coefficient
```

```
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_of_feature=x.shape[1]
num_of_feature
```

```
96
```

```
feature_index=np.arange(num_of_feature)
feature_index
```

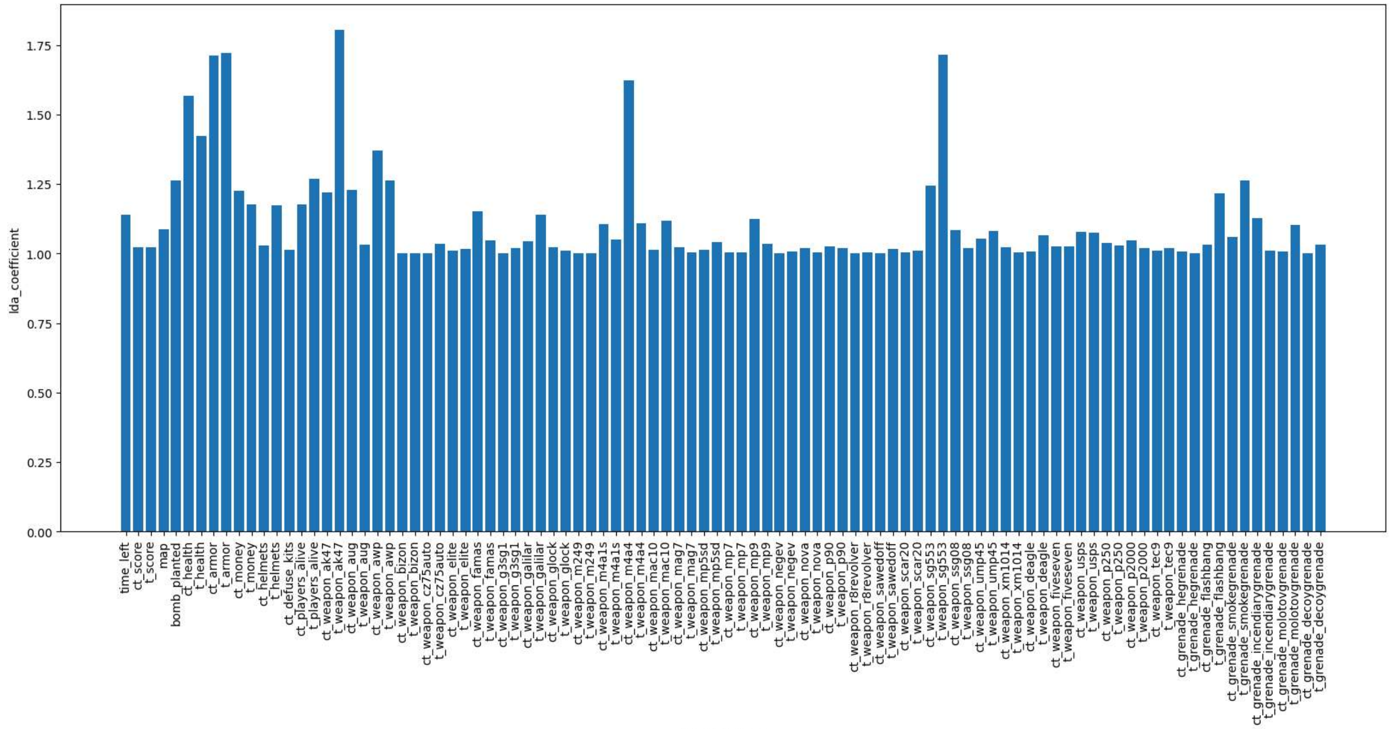
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95])
```

```
feature_name=list(x.columns)
```

```
feature_name
```

```
['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',
```

```
plt.figure(figsize=(20,8))
plt.bar(feature_index,lda_coefficient)
plt.xlabel('feature_name')
plt.ylabel('lda_coefficient')
plt.xticks(feature_index,feature_name,rotation=90)
plt.show()
```



```
data_feature_score=pd.DataFrame({"feature_name":feature_name,"feature_score":lda_coefficient})
data_feature_score
```

	feature_name	feature_score
0	time_left	1.140514
1	ct_score	1.023482
2	t_score	1.023610
3	map	1.087112
4	bomb_planted	1.264437
...	...	...
91	t_grenade_incendiarygrenade	1.010063
92	ct_grenade_molotovgrenade	1.008522
93	t_grenade_molotovgrenade	1.104241
94	ct_grenade_decoygrenade	1.002317
95	t_grenade_decoygrenade	1.032932

96 rows × 2 columns

```
top_20=data_feature_score.nlargest(20,'feature_score')
```

```
top_20
```

	feature_name	feature_score
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.index
```

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

```
xtrain
```

```
array([[-1.18271918,  1.31487825,  1.70421317, ..., -0.81273923,
        -0.16466055, -0.15653985],
       [-0.851754  ,  1.7326225 ,  0.04557825, ..., -0.21118431,
        -0.16466055, -0.15653985],
       [-0.05427852,  2.15036674,  1.70421317, ...,  0.99192555,
        -0.16466055, -0.15653985],
       ...,
       [-0.42069114, -0.56497085, -0.78373921, ...,  1.59348047,
         5.69041524, -0.15653985],
       [ 0.31323609, -1.40045934, -1.40572731, ..., -0.81273923,
        -0.16466055, -0.15653985],
       [-0.78820574, -0.56497085,  1.08222507, ..., -0.21118431,
        -0.16466055,  5.9630747 ]])
```

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

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

```
xtrain
```

```
array([[-0.91294032, -1.22331646,  0.1549069 , ..., -0.22768296,
        -0.53827509, -1.38029672],
       [ 1.28715696,  0.24927145, -0.72297059, ..., -1.05769215,
        -0.80581744,  0.60967313],
       [-0.17957456,  1.13740812,  1.91066189, ..., -0.22768296,
         0.81178478,  1.1071656 ],
       ...,
       [-0.17957456,  0.3925193 ,  0.1549069 , ...,  0.60232623,
        -0.34482139,  0.60967313],
       [-0.91294032,  0.58160646, -0.72297059, ...,  0.60232623,
        -0.86755798, -1.38029672],
       [ 0.5537912 ,  0.58160646, -0.72297059, ...,  0.60232623,
        -0.88813816,  0.60967313]])
```

```
#model building LogisticRegression
from sklearn.linear_model import LogisticRegression
```

```
lr=LogisticRegression()
```

```
lr.fit(xtrain,ytrain)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
ypred=lr.predict(xtest)
```

```
from sklearn.metrics import *
```

```
accuracy_score(ytest,ypred)
```

```
0.7516951229474717
```

```
#DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier()
```

```
dt.fit(xtrain,ytrain)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
ypred2=dt.predict(xtest)
```

```
accuracy_score(ytest,ypred2)
```

```
0.814802712196716
```

```
#RandomForestClassifier

from sklearn.ensemble import RandomForestClassifier
```

```
rfc=RandomForestClassifier()
```

```
rfc.fit(xtrain,ytrain)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
ypred3=rfc.predict(xtest)
```

```
accuracy_score(ytest,ypred3)
```

```
0.8550771995751981
```

```
#knearsest neighbor
from sklearn.neighbors import KNeighborsClassifier
```

```
knc=KNeighborsClassifier()
```

```
knc.fit(xtrain,ytrain)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
ypred4=knc.predict(xtest)
```

```
accuarcy_score=(ytest,ypred4)
```

```
78.5551685256235
```

```
78.5551685256235
```

```
Start coding or generate with AI.
```

```
#XGBoost
import xgboost as xgb
```

```
xgbc=xgb.XGBClassifier()
xgbc.fit(xtrain,ytrain)
ypred5=xgbc.predict(xtest)
```

```
accuracy_score(ytest,ypred4)
```

```
0.7880892083980067
```

```
Start coding or generate with AI.
```