```
import pandas as pd

df_train = pd.read_csv("Finance/train_data.csv")
df_train
```

```
            Time        V1        V2        V3        V4        V5
V6  \
0        38355.0  1.043949  0.318555  1.045810  2.805989 -0.561113 -
0.367956
1        22555.0 -1.665159  0.808440  1.805627  1.903416 -0.821627
0.934790
2         2431.0 -0.324096  0.601836  0.865329 -2.138000  0.294663 -
1.251553
3        86773.0 -0.258270  1.217501 -0.585348 -0.875347  1.222481 -
0.311027
4       127202.0  2.142162 -0.494988 -1.936511 -0.818288 -0.025213 -
1.027245
...          ...       ...       ...       ...       ...       ...
...
227840   62074.0 -1.993953  1.734986 -1.108037 -2.671817  1.605319
3.041992
227841   32193.0 -0.440396  1.062920  1.582218 -0.029761  0.040967 -
0.903627
227842  163864.0  0.827820 -2.649344 -3.161235  0.209209 -0.561331 -
1.570386
227843  122571.0 -1.523903 -6.287060 -2.638246  1.330015 -1.672002
1.957509
227844   43440.0 -1.608560  0.132746  2.075995 -1.937332 -1.822305 -
0.429669

              V7        V8        V9  ...       V21       V22
V23  \
0       0.032736 -0.042333 -0.322674  ... -0.240105 -0.680315
0.085328
1      -0.824802  0.975890  1.747469  ... -0.335332 -0.510994
0.035839
2       1.072114 -0.334896  1.071268  ...  0.012220  0.352856 -
0.341505
3       1.073860 -0.161408  0.200665  ... -0.424626 -0.781158
0.019316
4      -0.151627 -0.305750 -0.869482  ...  0.010115  0.021722
0.079463
...          ...       ...       ...  ...       ...       ...       ..
.
227840 -0.417771  1.438007  0.945437  ... -0.303532 -0.708199
0.047110
227841  0.730326 -0.108175 -0.513163  ... -0.215794 -0.532224 -
0.024762
227842  1.612531 -0.930219 -1.318562  ...  0.349915  0.002268 -
0.746698
```

```
227843  1.359226  0.081727  0.753151  ...  1.329127  0.001210 -
1.360187
227844  0.247042  0.684452  1.177470  ...  0.465181  1.017280
0.173478

             V24       V25       V26       V27       V28   Amount
Class
0       0.684812  0.318620 -0.204963  0.001662  0.037894    49.67
0
1       0.147565 -0.529358 -0.566950 -0.595998 -0.220086    16.94
0
2      -0.145791  0.094194 -0.804026  0.229428 -0.021623     1.00
0
3       0.178614 -0.315616  0.096665  0.269740 -0.020635    10.78
0
4      -0.480899  0.023846 -0.279076 -0.030121 -0.043888    39.96
0
...          ...       ...       ...       ...       ...      ...    .
..
227840  1.008409  0.234363  0.768581  0.697625  0.354542    14.83
0
227841  0.382581 -0.164620  0.068836  0.269144  0.123483     2.58
0
227842  0.171847  0.247576  0.936557 -0.258164  0.037868   748.04
0
227843 -1.507703 -1.183927  0.578076 -0.328557  0.229935  1771.50
0
227844  0.570107  0.504597 -0.659853  0.175060  0.092039   191.80
0

[227845 rows x 31 columns]
```

```
df_test = pd.read_csv("Finance/test_data.csv")
df_test
```

```
           Time        V1        V2        V3        V4        V5
V6  \
0      113050.0  0.114697  0.796303 -0.149553 -0.823011  0.878763 -
0.553152
1       26667.0 -0.039318  0.495784 -0.810884  0.546693  1.986257
4.386342
2      159519.0  2.275706 -1.531508 -1.021969 -1.602152 -1.220329 -
0.462376
3      137545.0  1.940137 -0.357671 -1.210551  0.382523  0.050823 -
0.171322
4       63369.0  1.081395 -0.502615  1.075887 -0.543359 -1.472946 -
1.065484
...         ...       ...       ...       ...       ...       ...
...
56957  136579.0  2.030797 -0.825073 -0.729555 -0.519187 -0.639893 -
```

```
0.169482
56958  150070.0 -0.263947  1.119700 -0.639394 -0.880567  1.194120 -
0.310693
56959  138634.0  2.206867 -0.748559 -1.443015 -1.101542 -0.332197 -
0.646931
56960   53907.0  1.430579 -0.842354  0.415998 -1.328439 -1.284654 -
0.888110
56961   66373.0 -7.792712  5.599937  0.258943  0.061360 -2.586555
4.770837

             V7         V8        V9  ...       V20        V21
V22  \
0      0.939259  -0.108502  0.111137  ... -0.042711  -0.335776 -
0.807853
1     -1.344891  -1.743736 -0.563103  ...  0.926255  -1.377003 -
0.072200
2     -1.196485  -0.147058 -0.950224  ... -0.408289  -0.193271 -
0.103533
3     -0.109124  -0.002115  0.869258  ... -0.199280   0.157994
0.650355
4     -0.443231  -0.143374  1.659826  ...  0.059880   0.224157
0.821209
...         ...        ...       ...  ...       ...        ...       .
..
56957 -0.619049  -0.017902 -0.578643  ... -0.458358  -0.790167 -
1.825357
56958  0.962087  -0.088880  0.386664  ...  0.294948  -0.448081 -
0.893010
56959 -0.536272  -0.129437 -0.712381  ... -0.065957   0.471336
1.314052
56960 -0.653237  -0.238164 -2.220845  ... -0.289844  -0.577415 -
1.323989
56961 -8.221863 -20.298380  2.028566  ... -2.136013  12.641459 -
4.187308

            V23       V24       V25       V26       V27       V28
Amount
0     -0.055940 -1.025281 -0.369557  0.204653  0.242724  0.085713
0.89
1     -0.197573  1.014807  1.011293 -0.167684  0.113136  0.256836
85.00
2      0.150945 -0.811083 -0.197913 -0.128446  0.014197 -0.051289
42.70
3      0.034206  0.739535  0.223605 -0.195509 -0.012791 -0.056841
29.99
4     -0.137223  0.986259  0.563228 -0.574206  0.089673  0.052036
68.00
...         ...       ...       ...       ...       ...       ...
...
56957  0.600083  0.702623 -0.782688 -0.007105 -0.041057 -0.038601
```

```
42.42
56958  0.004678  0.062555 -0.347536  0.106510  0.274117 -0.036263
7.99
56959  0.038930  0.747315  0.158017  0.021897 -0.046200 -0.072586
1.00
56960  0.336843  0.329714 -0.007425 -0.636401  0.037095  0.029180
30.00
56961  2.655058  0.350225  0.462620  0.455248  0.166157  0.099680
8.39

[56962 rows x 30 columns]
```

```
df_train.shape
```

```
(227845, 31)
```

```
df_train['Class'].unique()
```

```
array([0, 1])
```

```
df_train['Class'].value_counts()
```

```
0    227451
1       394
Name: Class, dtype: int64
```

```
df_test.shape
```

```
(56962, 30)
```

```
df_train.isnull().sum()
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
```

```
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227845 entries, 0 to 227844
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    227845 non-null  float64
 1   V1      227845 non-null  float64
 2   V2      227845 non-null  float64
 3   V3      227845 non-null  float64
 4   V4      227845 non-null  float64
 5   V5      227845 non-null  float64
 6   V6      227845 non-null  float64
 7   V7      227845 non-null  float64
 8   V8      227845 non-null  float64
 9   V9      227845 non-null  float64
 10  V10     227845 non-null  float64
 11  V11     227845 non-null  float64
 12  V12     227845 non-null  float64
 13  V13     227845 non-null  float64
 14  V14     227845 non-null  float64
 15  V15     227845 non-null  float64
 16  V16     227845 non-null  float64
 17  V17     227845 non-null  float64
 18  V18     227845 non-null  float64
 19  V19     227845 non-null  float64
 20  V20     227845 non-null  float64
 21  V21     227845 non-null  float64
 22  V22     227845 non-null  float64
 23  V23     227845 non-null  float64
 24  V24     227845 non-null  float64
 25  V25     227845 non-null  float64
 26  V26     227845 non-null  float64
 27  V27     227845 non-null  float64
 28  V28     227845 non-null  float64
 29  Amount  227845 non-null  float64
 30  Class   227845 non-null  int64
```

```
dtypes: float64(30), int64(1)
memory usage: 53.9 MB
```

```
df_train.describe()
```

```
                Time             V1             V2             V3  \
count  227845.000000  227845.000000  227845.000000  227845.000000
mean    94752.853076      -0.003321      -0.001652       0.001066
std     47500.410602       1.963028       1.661178       1.516107
min         0.000000     -56.407510     -72.715728     -32.965346
25%     54182.000000      -0.922851      -0.598040      -0.889246
50%     84607.000000       0.012663       0.066665       0.182170
75%    139340.000000       1.314821       0.804401       1.029449
max    172792.000000       2.454930      22.057729       9.382558

                 V4             V5             V6             V7  \
count  227845.000000  227845.000000  227845.000000  227845.000000
mean       -0.000374       0.000877       0.000770      -0.000035
std         1.415061       1.367074       1.325341       1.220384
min        -5.683171     -42.147898     -26.160506     -43.557242
25%        -0.848884      -0.690811      -0.767803      -0.554761
50%        -0.019309      -0.055243      -0.273025       0.040409
75%         0.744822       0.610852       0.400298       0.570631
max        16.875344      34.801666      22.529298      36.877368

                 V8             V9  ...            V21            V22
\
count  227845.000000  227845.000000  ...  227845.000000  227845.000000

mean        0.001625      -0.000391  ...       0.000563       0.001234

std         1.192648       1.097367  ...       0.734187       0.724544

min       -73.216718     -13.434066  ...     -34.830382     -10.933144

25%        -0.207838      -0.643365  ...      -0.228031      -0.540792

50%         0.022928      -0.050932  ...      -0.028807       0.008697

75%         0.327854       0.596671  ...       0.186852       0.529535

max        20.007208      15.594995  ...      27.202839      10.503090


                V23            V24            V25            V26  \
count  227845.000000  227845.000000  227845.000000  227845.000000
mean       -0.001002       0.000254       0.000218      -0.001128
std         0.625165       0.606012       0.521348       0.482314
min       -44.807735      -2.836627     -10.295397      -2.604551
```

```
25%        -0.162264      -0.354099      -0.317450      -0.327910
50%        -0.011614       0.041212       0.016221      -0.053257
75%         0.147067       0.440051       0.351214       0.239885
max        22.528412       4.022866       6.070850       3.463246

                 V27            V28         Amount          Class
count  227845.000000  227845.000000  227845.000000  227845.000000
mean       -0.000346       0.000498      88.522327       0.001729
std         0.400286       0.331184     248.100141       0.041548
min       -22.565679     -11.710896       0.000000       0.000000
25%        -0.070986      -0.053117       5.590000       0.000000
50%         0.001315       0.011216      22.000000       0.000000
75%         0.091105       0.078458      77.070000       0.000000
max        12.152401      33.847808   19656.530000       1.000000

[8 rows x 31 columns]
```

```
#sns.boxplot( x="Time", y="Amount", data=df_train, )
#plt.show()
```

Over Sampling

oversampling involves randomly selecting examples from the minority class, with replacement, and adding them to the training dataset.

```
df_test = pd.read_csv("Finance/test_data_hidden.csv")
df_test
```

```
           Time        V1        V2        V3        V4        V5
V6  \
0      113050.0  0.114697  0.796303 -0.149553 -0.823011  0.878763 -
0.553152
1       26667.0 -0.039318  0.495784 -0.810884  0.546693  1.986257
4.386342
2      159519.0  2.275706 -1.531508 -1.021969 -1.602152 -1.220329 -
0.462376
3      137545.0  1.940137 -0.357671 -1.210551  0.382523  0.050823 -
0.171322
4       63369.0  1.081395 -0.502615  1.075887 -0.543359 -1.472946 -
1.065484
...         ...       ...       ...       ...       ...       ...
...
56957  136579.0  2.030797 -0.825073 -0.729555 -0.519187 -0.639893 -
0.169482
56958  150070.0 -0.263947  1.119700 -0.639394 -0.880567  1.194120 -
0.310693
56959  138634.0  2.206867 -0.748559 -1.443015 -1.101542 -0.332197 -
0.646931
56960   53907.0  1.430579 -0.842354  0.415998 -1.328439 -1.284654 -
```

```
0.888110
56961   66373.0 -7.792712  5.599937  0.258943  0.061360 -2.586555
4.770837

             V7         V8        V9  ...        V21       V22
V23  \
0      0.939259  -0.108502  0.111137  ...  -0.335776 -0.807853 -
0.055940
1     -1.344891  -1.743736 -0.563103  ...  -1.377003 -0.072200 -
0.197573
2     -1.196485  -0.147058 -0.950224  ...  -0.193271 -0.103533
0.150945
3     -0.109124  -0.002115  0.869258  ...   0.157994  0.650355
0.034206
4     -0.443231  -0.143374  1.659826  ...   0.224157  0.821209 -
0.137223
...         ...        ...       ...  ...        ...       ...       .
..
56957 -0.619049  -0.017902 -0.578643  ...  -0.790167 -1.825357
0.600083
56958  0.962087  -0.088880  0.386664  ...  -0.448081 -0.893010
0.004678
56959 -0.536272  -0.129437 -0.712381  ...   0.471336  1.314052
0.038930
56960 -0.653237  -0.238164 -2.220845  ...  -0.577415 -1.323989
0.336843
56961 -8.221863 -20.298380  2.028566  ...  12.641459 -4.187308
2.655058

            V24       V25       V26       V27       V28  Amount  Class

0     -1.025281 -0.369557  0.204653  0.242724  0.085713    0.89      0

1      1.014807  1.011293 -0.167684  0.113136  0.256836   85.00      0

2     -0.811083 -0.197913 -0.128446  0.014197 -0.051289   42.70      0

3      0.739535  0.223605 -0.195509 -0.012791 -0.056841   29.99      0

4      0.986259  0.563228 -0.574206  0.089673  0.052036   68.00      0

...         ...       ...       ...       ...       ...     ...    ...

56957  0.702623 -0.782688 -0.007105 -0.041057 -0.038601   42.42      0

56958  0.062555 -0.347536  0.106510  0.274117 -0.036263    7.99      0

56959  0.747315  0.158017  0.021897 -0.046200 -0.072586    1.00      0
```

```
56960  0.329714 -0.007425 -0.636401  0.037095  0.029180   30.00      0

56961  0.350225  0.462620  0.455248  0.166157  0.099680    8.39      0


[56962 rows x 31 columns]
```

```
from numpy import mean
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.tree import DecisionTreeClassifier
from imblearn.pipeline import Pipeline
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler

# define dataset
X, y = make_classification(n_samples=df_train.shape[0],
weights=[0.99], flip_y=0)
# define pipeline
steps = [('over', RandomOverSampler()), ('model',
DecisionTreeClassifier())]
pipeline = Pipeline(steps=steps)
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='f1_micro', cv=cv,
n_jobs=-1)
score = mean(scores)
print('F1 Score on Training dataset: %.3f' % score)
```

```
F1 Score on Training dataset: 0.987
```

```
# define dataset
X, y = make_classification(n_samples=df_test.shape[0], weights=[0.99],
flip_y=0)
# define pipeline
steps = [('over', RandomOverSampler()), ('model',
DecisionTreeClassifier())]
pipeline = Pipeline(steps=steps)
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='f1_micro', cv=cv,
n_jobs=-1)
score = mean(scores)
print('F1 Score on Testing Dataset: %.3f' % score)
```

```
F1 Score on Testing Dataset: 0.998
```

Undersampling

```
# define dataset
X, y = make_classification(n_samples=df_train.shape[0],
weights=[0.99], flip_y=0)
# define pipeline
steps = [('under', RandomUnderSampler()), ('model',
DecisionTreeClassifier())]
pipeline = Pipeline(steps=steps)
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='f1_micro', cv=cv,
n_jobs=-1)
score = mean(scores)
print('F1 Score on Training Dataset: %.3f' % score)
```

```
F1 Score on Training Dataset: 0.925
```

```
# define dataset
X, y = make_classification(n_samples=df_test.shape[0], weights=[0.99],
flip_y=0)
# define pipeline
steps = [('under', RandomUnderSampler()), ('model',
DecisionTreeClassifier())]
pipeline = Pipeline(steps=steps)
# evaluate pipeline
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='f1_micro', cv=cv,
n_jobs=-1)
score = mean(scores)
print('F1 Score on Testing Dataset: %.3f' % score)
```

```
F1 Score on Testing Dataset: 0.905
```

Liner Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

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

```
dataset1 = pd.read_csv("Finance/train_data.csv")
dataset2 = pd.read_csv("Finance/test_data_hidden.csv")

dataset1.shape

(227845, 31)

dataset2.shape

(56962, 31)

dataset = dataset1.append(dataset2)
dataset.shape

(284807, 31)

corr_var=dataset.corr()
print(corr_var)
plt.figure(figsize=(20,30))
sns.heatmap(corr_var, annot=True, cmap='BuPu')
```

```
            Time            V1            V2            V3
V4  \
Time    1.000000  1.173963e-01 -1.059333e-02 -4.196182e-01 -1.052602e-
01
V1      0.117396  1.000000e+00 -6.568731e-17 -5.332665e-16 -3.109935e-
16
V2     -0.010593 -6.568731e-17  1.000000e+00 -7.274769e-18 -2.274135e-
16
V3     -0.419618 -5.332665e-16 -7.274769e-18  1.000000e+00 -3.590752e-
16
V4     -0.105260 -3.109935e-16 -2.274135e-16 -3.590752e-16
1.000000e+00
V5      0.173072  2.612367e-16  1.540214e-16 -5.016223e-16 -1.850561e-
15
V6     -0.063016  1.147932e-16  3.407670e-16  1.490639e-15 -4.829386e-
16
V7      0.084714 -2.238366e-17 -2.823933e-16  1.681293e-16 -1.759757e-
17
V8     -0.036949 -3.079914e-17  2.593188e-17  1.668496e-17  6.402592e-
16
V9     -0.008660  6.644345e-17 -5.249762e-17  9.001567e-17  5.940802e-
16
V10     0.030617  5.732546e-17 -1.361335e-16  3.063001e-16 -1.218780e-
16
V11    -0.247689  2.960454e-16  3.480859e-16  7.476534e-18 -2.281390e-
16
V12     0.124348  2.381275e-16 -3.059475e-16  2.040316e-16 -1.783419e-
16
V13    -0.065902 -1.080252e-16 -4.934619e-17 -3.702347e-17  1.214180e-
17
V14    -0.098757  3.765465e-16 -4.715701e-16  6.584840e-16 -8.348175e-
```

```
17
V15    -0.183453 -1.898719e-16  1.124884e-16 -3.139739e-17  2.111781e-
16
V16     0.011903  2.870095e-16  1.448723e-16  5.206459e-16 -1.193152e-
16
V17    -0.073297 -9.072968e-18 -5.555087e-16  2.670633e-16 -2.263165e-
16
V18     0.090438  2.176172e-16  2.107043e-16  2.245032e-16  2.141898e-
17
V19     0.028975  1.929888e-16  8.289147e-18  3.454114e-16 -2.970904e-
16
V20    -0.050866  4.004679e-16  2.912728e-16  6.421726e-17 -2.420843e-
16
V21     0.044736 -1.432347e-16  4.496318e-17 -1.270991e-16 -7.166526e-
17
V22     0.144059  1.225977e-16  2.834994e-16 -2.106097e-16  2.500354e-
16
V23     0.051142  1.634933e-16 -4.510491e-16 -6.212022e-17  1.564425e-
16
V24    -0.016182 -5.872307e-17 -1.788168e-16  1.440958e-17  1.823204e-
16
V25    -0.233083 -2.623325e-16  9.228550e-17  1.079754e-16  5.839668e-
16
V26    -0.041407 -1.843730e-16  2.894134e-16 -2.720852e-16 -4.017489e-
16
V27    -0.005135  1.311170e-16 -7.691893e-16  6.687037e-16  8.794217e-
17
V28    -0.009413  4.363514e-16 -5.223624e-16  6.188946e-16 -8.482090e-
17
Amount -0.010596 -2.277087e-01 -5.314089e-01 -2.108805e-01  9.873167e-
02
Class  -0.012323 -1.013473e-01  9.128865e-02 -1.929608e-01  1.334475e-
01

                  V5            V6            V7            V8
V9  \
Time    1.730721e-01 -6.301647e-02  8.471437e-02 -3.694943e-02 -
8.660434e-03
V1      2.612367e-16  1.147932e-16 -2.238366e-17 -3.079914e-17
6.644345e-17
V2      1.540214e-16  3.407670e-16 -2.823933e-16  2.593188e-17 -
5.249762e-17
V3     -5.016223e-16  1.490639e-15  1.681293e-16  1.668496e-17
9.001567e-17
V4     -1.850561e-15 -4.829386e-16 -1.759757e-17  6.402592e-16
5.940802e-16
V5      1.000000e+00  9.049727e-16  2.556132e-16  4.805930e-16
4.801088e-16
V6      9.049727e-16  1.000000e+00 -3.880122e-16 -4.236754e-16 -
1.045368e-16
```

```
V7      2.556132e-16 -3.880122e-16  1.000000e+00 -3.019071e-16
1.438403e-16
V8      4.805930e-16 -4.236754e-16 -3.019071e-16  1.000000e+00
3.783254e-16
V9      4.801088e-16 -1.045368e-16  1.438403e-16  3.783254e-16
1.000000e+00
V10    -4.550919e-18  1.406497e-16 -9.611507e-17 -4.146154e-17 -
2.381227e-16
V11     7.066448e-16  8.762967e-16 -3.596580e-16  7.752017e-17
3.095044e-16
V12     2.964200e-16  2.877651e-16  7.621904e-16  1.260587e-17 -
1.267878e-15
V13    -2.794638e-16 -1.666685e-16 -3.654871e-17 -3.705123e-16
9.661516e-16
V14     2.525963e-16  3.858452e-16  2.089100e-17 -2.629747e-16
9.720337e-16
V15     1.839756e-16 -1.424793e-16 -1.020003e-16  1.503457e-16 -
8.443815e-16
V16     7.243059e-16 -1.952121e-16  4.700217e-16  2.081757e-16 -
5.830182e-16
V17     6.013967e-16  1.355398e-16  6.523970e-16 -2.983738e-16
7.782034e-16
V18     3.954471e-16  6.413279e-17  1.734726e-16 -3.725256e-16
1.317471e-16
V19    -1.003894e-16  7.054593e-17 -7.084995e-17 -3.505972e-16
1.167258e-16
V20    -1.119966e-16 -1.735920e-16  3.461662e-16  5.763201e-17 -
2.750288e-16
V21     1.782612e-16 -9.471119e-17  1.692918e-16  1.258107e-16
2.402789e-16
V22     7.833339e-18 -8.290890e-17 -5.620145e-16  1.560090e-17 -
1.866478e-16
V23     7.782694e-17  5.837112e-17 -2.328622e-16  3.284846e-16 -
4.103339e-17
V24    -9.779101e-16 -1.017336e-15 -6.154313e-17 -2.668076e-16 -
2.759459e-16
V25    -1.174113e-16  5.717903e-16  8.291426e-17 -1.455408e-16
2.619596e-16
V26     3.373739e-16 -2.701271e-16 -7.800136e-16  4.893721e-17 -
1.000314e-16
V27     6.353219e-16 -1.037262e-16 -5.187221e-16  4.210334e-16 -
3.963412e-17
V28    -3.553727e-16  6.991654e-16 -1.532296e-16 -5.508323e-16
8.082647e-16
Amount -3.863563e-01  2.159812e-01  3.973113e-01 -1.030791e-01 -
4.424560e-02
Class  -9.497430e-02 -4.364316e-02 -1.872566e-01  1.987512e-02 -
9.773269e-02

        ...           V21           V22           V23           V24  \
```

```
Time    ...  4.473573e-02  1.440591e-01  5.114236e-02 -1.618187e-02
V1      ... -1.432347e-16  1.225977e-16  1.634933e-16 -5.872307e-17
V2      ...  4.496318e-17  2.834994e-16 -4.510491e-16 -1.788168e-16
V3      ... -1.270991e-16 -2.106097e-16 -6.212022e-17  1.440958e-17
V4      ... -7.166526e-17  2.500354e-16  1.564425e-16  1.823204e-16
V5      ...  1.782612e-16  7.833339e-18  7.782694e-17 -9.779101e-16
V6      ... -9.471119e-17 -8.290890e-17  5.837112e-17 -1.017336e-15
V7      ...  1.692918e-16 -5.620145e-16 -2.328622e-16 -6.154313e-17
V8      ...  1.258107e-16  1.560090e-17  3.284846e-16 -2.668076e-16
V9      ...  2.402789e-16 -1.866478e-16 -4.103339e-17 -2.759459e-16
V10     ...  1.077236e-15 -2.817806e-16  2.059390e-16 -9.266032e-17
V11     ...  2.327658e-17  3.351216e-17  1.277124e-16  1.642064e-15
V12     ...  6.321910e-16 -3.585931e-17  2.911165e-16  4.438099e-16
V13     ...  1.496961e-16 -5.437684e-17 -7.552734e-16 -6.281499e-16
V14     ... -2.385105e-16  6.175645e-16  2.364425e-16  2.461719e-17
V15     ...  8.506940e-17 -3.438792e-16  4.600508e-17 -4.623591e-16
V16     ... -3.581023e-16  2.628305e-16  8.526701e-16 -3.501789e-16
V17     ... -1.031391e-15 -2.886278e-16  4.032753e-16 -1.512009e-16
V18     ... -1.195484e-15 -5.529936e-16 -2.998763e-16 -1.898616e-16
V19     ...  5.732189e-16 -9.819138e-16  6.615934e-16 -4.030007e-17
V20     ... -1.137837e-15  9.503417e-16  1.009969e-16  1.747400e-16
V21     ...  1.000000e+00  3.743426e-15  6.715706e-16  1.343139e-16
V22     ...  3.743426e-15  1.000000e+00  3.367295e-16  4.301374e-17
V23     ...  6.715706e-16  3.367295e-16  1.000000e+00  7.143132e-17
V24     ...  1.343139e-16  4.301374e-17  7.143132e-17  1.000000e+00
V25     ... -1.725294e-16 -9.593905e-16 -6.181584e-16  1.236255e-15
V26     ... -4.947650e-16 -2.983061e-17  1.220224e-15  1.652930e-16
V27     ... -1.163418e-15  8.965419e-17  4.106698e-16 -3.334188e-16
V28     ...  2.776732e-16 -5.376112e-16  1.364354e-15 -2.465020e-16
Amount  ...  1.059989e-01 -6.480065e-02 -1.126326e-01  5.146217e-03
Class   ...  4.041338e-02  8.053175e-04 -2.685156e-03 -7.220907e-03

                 V25           V26           V27           V28
Amount  \
Time   -2.330828e-01 -4.140710e-02 -5.134591e-03 -9.412688e-03 -
0.010596
V1     -2.623325e-16 -1.843730e-16  1.311170e-16  4.363514e-16 -
0.227709
V2      9.228550e-17  2.894134e-16 -7.691893e-16 -5.223624e-16 -
0.531409
V3      1.079754e-16 -2.720852e-16  6.687037e-16  6.188946e-16 -
0.210880
V4      5.839668e-16 -4.017489e-16  8.794217e-17 -8.482090e-17
0.098732
V5     -1.174113e-16  3.373739e-16  6.353219e-16 -3.553727e-16 -
0.386356
V6      5.717903e-16 -2.701271e-16 -1.037262e-16  6.991654e-16
0.215981
V7      8.291426e-17 -7.800136e-16 -5.187221e-16 -1.532296e-16
0.397311
```

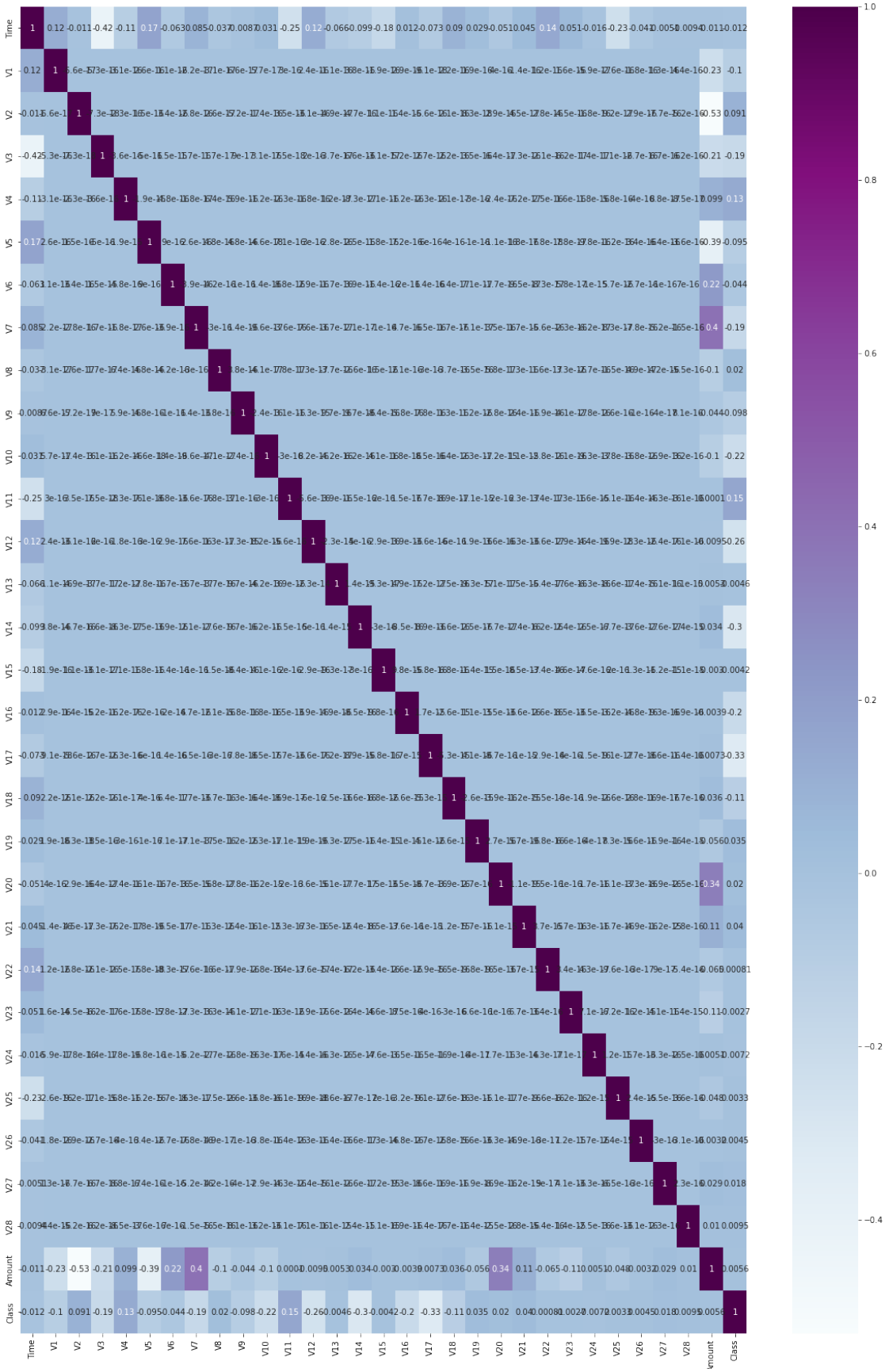
```
V8     -1.455408e-16  4.893721e-17  4.210334e-16 -5.508323e-16 -
0.103079
V9      2.619596e-16 -1.000314e-16 -3.963412e-17  8.082647e-16 -
0.044246
V10    -3.782206e-16 -3.839450e-16 -2.896641e-16  3.235169e-16 -
0.101502
V11    -6.081670e-16 -1.429259e-16 -4.285748e-16 -3.061550e-16
0.000104
V12     9.855041e-18  2.322358e-16 -2.439481e-16  7.076925e-16 -
0.009542
V13    -8.637726e-17 -1.418114e-16 -5.064226e-16  1.081446e-15
0.005293
V14    -6.695280e-17 -3.618160e-17 -2.632492e-17  2.391432e-15
0.033751
V15     1.966864e-16  1.261962e-16 -1.209832e-15 -1.147542e-15 -
0.002986
V16    -3.201784e-16 -4.765546e-16  9.325334e-16  6.850197e-16 -
0.003910
V17     9.088853e-17  2.702516e-16  8.603326e-16 -1.421158e-16
0.007309
V18    -2.557015e-16  2.837130e-16  1.855542e-16  7.696346e-16
0.035650
V19     8.280445e-16  5.636014e-16 -1.931479e-16 -1.363588e-15 -
0.056151
V20    -1.131622e-17 -3.250766e-16 -8.946970e-16 -2.470720e-16
0.339403
V21    -1.725294e-16 -4.947650e-16 -1.163418e-15  2.776732e-16
0.105999
V22    -9.593905e-16 -2.983061e-17  8.965419e-17 -5.376112e-16 -
0.064801
V23    -6.181584e-16  1.220224e-15  4.106698e-16  1.364354e-15 -
0.112633
V24     1.236255e-15  1.652930e-16 -3.334188e-16 -2.465020e-16
0.005146
V25     1.000000e+00  2.430404e-15 -6.467898e-16  3.576794e-16 -
0.047837
V26     2.430404e-15  1.000000e+00 -3.046841e-16 -3.061907e-16 -
0.003208
V27    -6.467898e-16 -3.046841e-16  1.000000e+00  2.339586e-16
0.028825
V28     3.576794e-16 -3.061907e-16  2.339586e-16  1.000000e+00
0.010258
Amount -4.783686e-02 -3.208037e-03  2.882546e-02  1.025822e-02
1.000000
Class   3.307706e-03  4.455398e-03  1.757973e-02  9.536041e-03
0.005632

           Class
Time   -0.012323
V1     -0.101347
```

```
V2      0.091289
V3     -0.192961
V4      0.133447
V5     -0.094974
V6     -0.043643
V7     -0.187257
V8      0.019875
V9     -0.097733
V10    -0.216883
V11     0.154876
V12    -0.260593
V13    -0.004570
V14    -0.302544
V15    -0.004223
V16    -0.196539
V17    -0.326481
V18    -0.111485
V19     0.034783
V20     0.020090
V21     0.040413
V22     0.000805
V23    -0.002685
V24    -0.007221
V25     0.003308
V26     0.004455
V27     0.017580
V28     0.009536
Amount  0.005632
Class   1.000000

[31 rows x 31 columns]
```

```
<AxesSubplot:>
```


Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

```
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
```

```
X
```

```
array([[ 3.83550000e+04,  1.04394859e+00,  3.18555351e-01, ...,
         1.66177594e-03,  3.78938133e-02,  4.96700000e+01],
       [ 2.25550000e+04, -1.66515873e+00,  8.08440107e-01, ...,
        -5.95998497e-01, -2.20085621e-01,  1.69400000e+01],
       [ 2.43100000e+03, -3.24095611e-01,  6.01835927e-01, ...,
         2.29428222e-01, -2.16228931e-02,  1.00000000e+00],
       ...,
       [ 1.38634000e+05,  2.20686736e+00, -7.48559259e-01, ...,
        -4.62001550e-02, -7.25858535e-02,  1.00000000e+00],
       [ 5.39070000e+04,  1.43057893e+00, -8.42353562e-01, ...,
         3.70949313e-02,  2.91797404e-02,  3.00000000e+01],
       [ 6.63730000e+04, -7.79271230e+00,  5.59993720e+00, ...,
         1.66156929e-01,  9.96801592e-02,  8.39000000e+00]])
```

```
y
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
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=0)
print('Total no. of samples: Training and Testing dataset
separately!')
print('X_train:', np.shape(X_train))
print('y_train:', np.shape(y_train))
print('X_test:', np.shape(X_test))
print('y_test:', np.shape(y_test))
```

```
Total no. of samples: Training and Testing dataset separately!
X_train: (227845, 30)
y_train: (227845,)
X_test: (56962, 30)
y_test: (56962,)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from sklearn.linear_model import LogisticRegression
classifier7 = LogisticRegression()
classifier7.fit(X_train,y_train)
```

```
LogisticRegression()
```

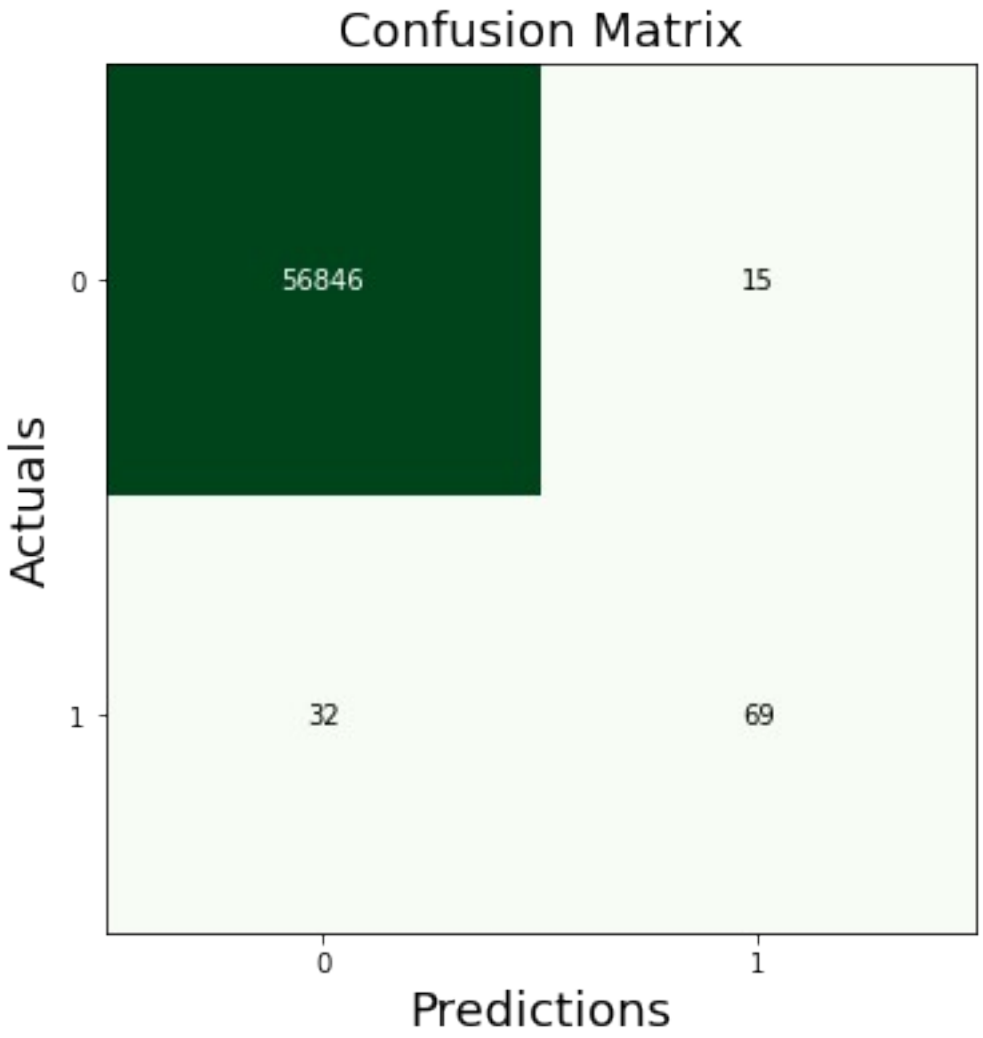
```
y_pred7 = classifier7.predict(X_test)
print(np.concatenate((y_pred7.reshape(len(y_pred7),1),
y_test.reshape(len(y_test),1)),1))
```

```
[[0 0]
 [0 0]
 [0 0]
 ...
 [0 0]
 [0 0]
 [0 0]]
```

```
from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score
cm7 = confusion_matrix(y_test, y_pred7)
print(cm7)
```

```
[[56846    15]
 [   32    69]]
```

```
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(conf_mat=cm7, figsize=(6, 6),
cmap=plt.cm.Greens)
plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```




```
logreg=accuracy_score(y_test,y_pred7)
logreg
```

```
0.9991748885221726
```

```
roc_auc_score(y_test, y_pred7)
```

```
0.8414522578161335
```

```
import sklearn.metrics as metrics
print(metrics.classification_report(y_test, y_pred7))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56861
           1       0.82      0.68      0.75       101

    accuracy                           1.00     56962
   macro avg       0.91      0.84      0.87     56962
```

```
weighted avg       1.00      1.00      1.00     56962
```

```
import warnings
warnings.simplefilter("ignore")
from sklearn.model_selection import GridSearchCV
parameters_lr = [{'penalty':['l1','l2'],'C': [0.001, 0.01, 0.1, 1, 10,
100, 1000]}]
grid_search_lr = GridSearchCV(estimator = classifier7,
                              param_grid = parameters_lr,
                              scoring = 'accuracy',
                              cv = 10,
                              n_jobs = -1)
grid_search_lr.fit(X_train, y_train)
best_accuracy_lr = grid_search_lr.best_score_
best_paramaeter_lr = grid_search_lr.best_params_
print("Best Accuracy of LR: {:.2f}
%".format(best_accuracy_lr.mean()*100))
print("Best Parameter of LR:", best_paramaeter_lr)
```

```
Best Accuracy of LR: 99.93 %
Best Parameter of LR: {'C': 1, 'penalty': 'l2'}
```

Feature selection

```
X
```

```
array([[ 3.83550000e+04,  1.04394859e+00,  3.18555351e-01, ...,
         1.66177594e-03,  3.78938133e-02,  4.96700000e+01],
       [ 2.25550000e+04, -1.66515873e+00,  8.08440107e-01, ...,
        -5.95998497e-01, -2.20085621e-01,  1.69400000e+01],
       [ 2.43100000e+03, -3.24095611e-01,  6.01835927e-01, ...,
         2.29428222e-01, -2.16228931e-02,  1.00000000e+00],
       ...,
       [ 1.38634000e+05,  2.20686736e+00, -7.48559259e-01, ...,
        -4.62001550e-02, -7.25858535e-02,  1.00000000e+00],
       [ 5.39070000e+04,  1.43057893e+00, -8.42353562e-01, ...,
         3.70949313e-02,  2.91797404e-02,  3.00000000e+01],
       [ 6.63730000e+04, -7.79271230e+00,  5.59993720e+00, ...,
         1.66156929e-01,  9.96801592e-02,  8.39000000e+00]])
```

```
y
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
X[X < 0] = 0
X
```

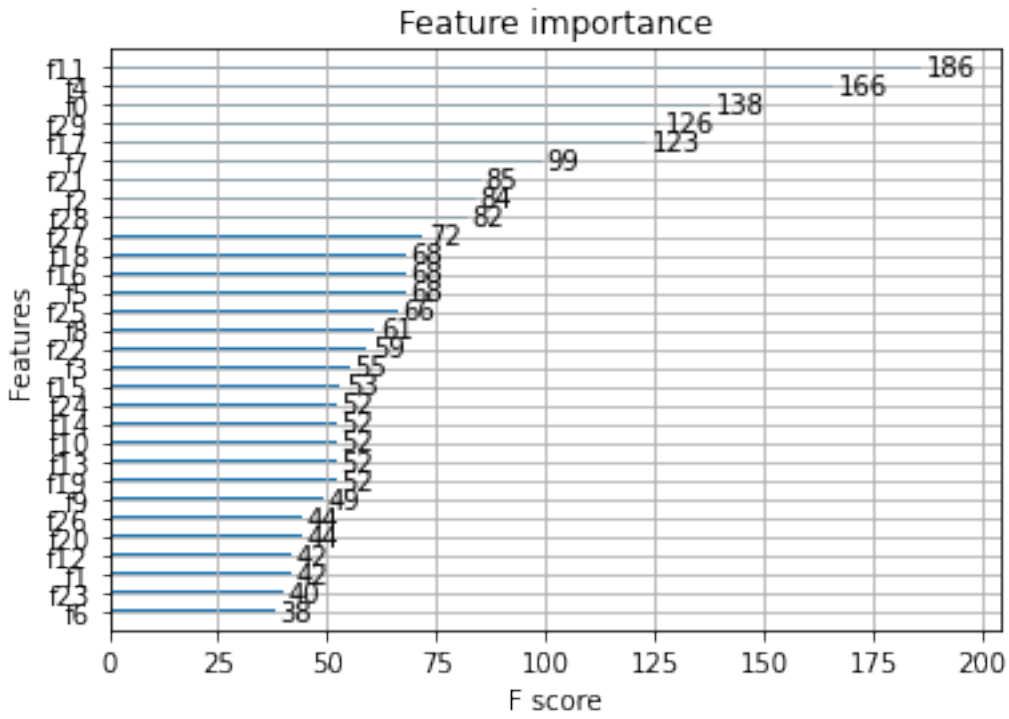
```
array([[3.83550000e+04, 1.04394859e+00, 3.18555351e-01, ...,
        1.66177594e-03, 3.78938133e-02, 4.96700000e+01],
       [2.25550000e+04, 0.00000000e+00, 8.08440107e-01, ...,
        0.00000000e+00, 0.00000000e+00, 1.69400000e+01],
       [2.43100000e+03, 0.00000000e+00, 6.01835927e-01, ...,
        2.29428222e-01, 0.00000000e+00, 1.00000000e+00],
       ...,
       [1.38634000e+05, 2.20686736e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 1.00000000e+00],
       [5.39070000e+04, 1.43057893e+00, 0.00000000e+00, ...,
        3.70949313e-02, 2.91797404e-02, 3.00000000e+01],
       [6.63730000e+04, 0.00000000e+00, 5.59993720e+00, ...,
        1.66156929e-01, 9.96801592e-02, 8.39000000e+00]])
```

```
from sklearn.ensemble import
GradientBoostingClassifier,RandomForestClassifier
gb = GradientBoostingClassifier()
rf = RandomForestClassifier()
rf.fit(X,y)
gb.fit(X,y)
print(gb.feature_importances_)
print(rf.feature_importances_)
```

```
[3.64633276e-04 0.00000000e+00 1.51358929e-04 0.00000000e+00
 1.59590394e-03 4.40769899e-19 0.00000000e+00 2.06862685e-03
 3.84479906e-04 0.00000000e+00 0.00000000e+00 6.56312393e-01
 2.10637566e-05 1.54684581e-05 0.00000000e+00 1.87466253e-04
 0.00000000e+00 5.93915709e-03 2.93746613e-04 1.31153999e-06
 0.00000000e+00 2.87885851e-01 0.00000000e+00 0.00000000e+00
 0.00000000e+00 0.00000000e+00 1.29867812e-04 4.43303461e-02
 2.12682066e-04 1.05643757e-04]
[0.02837775 0.01344033 0.05786517 0.00868204 0.1106635  0.0190326
 0.01193106 0.02878274 0.0373489  0.02353931 0.02744445 0.27590368
 0.01609382 0.01061878 0.01497391 0.01349619 0.02878912 0.03951694
 0.01575701 0.02182735 0.01844161 0.03614555 0.01351837 0.00857968
 0.01040161 0.01378099 0.01497803 0.02795423 0.02409873 0.02801655]
```

```
from numpy import loadtxt
from xgboost import XGBClassifier
from xgboost import plot_importance
from matplotlib import  pyplot
model = XGBClassifier()
plt.figure(figsize=(50,50))
model.fit(X,y)
plot_importance(model)
pyplot.show()
```

```
<Figure size 3600x3600 with 0 Axes>
```




```
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
clf = ExtraTreesClassifier(n_estimators = 10)
clf = clf.fit(X,y)
clf.feature_importances_
```

```
array([0.02861605, 0.01359411, 0.06563189, 0.02187146, 0.0778679 ,
       0.01391486, 0.0104974 , 0.01842076, 0.05610615, 0.01861837,
       0.02267911, 0.22028062, 0.02092831, 0.01415054, 0.01451976,
       0.01021286, 0.0234859 , 0.05872985, 0.02037552, 0.03715652,
       0.01897273, 0.03821916, 0.01419254, 0.01575347, 0.01647169,
       0.02370166, 0.01726524, 0.03154895, 0.03027836, 0.02593827])
```

```
model = SelectFromModel(clf, prefit = True)
X = model.transform(X)
X.shape
```

```
(284807, 7)
```

```
from sklearn.model_selection import train_test_split
trainX, testX, trainY, testY = train_test_split(X, y,
test_size=0.20,random_state=1)
```

```
trainX.shape
```

```
(227845, 7)
```

```
trainY.shape
```

```
(227845,)
```

```
import tensorflow as tf
from sklearn.preprocessing import Normalizer
transformer = Normalizer()
trainX = transformer.fit_transform(trainX)
testX = transformer.fit_transform(testX)

model = tf.keras.Sequential()
model.add(tf.keras.layers.Reshape((trainX.shape[1],),input_shape=(trai
nX.shape[1],)))

#Add 1st hidden layer
model.add(tf.keras.layers.Dense(2000))
model.add(tf.keras.layers.LeakyReLU())
model.add(tf.keras.layers.BatchNormalization())

#Add 2nd hidden layer
model.add(tf.keras.layers.Dense(1000))
model.add(tf.keras.layers.LeakyReLU())
model.add(tf.keras.layers.BatchNormalization())

#Add 3rd hidden layer
model.add(tf.keras.layers.Dense(1000))
model.add(tf.keras.layers.LeakyReLU())
model.add(tf.keras.layers.BatchNormalization())

#Add 4th hidden layer
model.add(tf.keras.layers.Dense(300))
model.add(tf.keras.layers.LeakyReLU())
model.add(tf.keras.layers.BatchNormalization())

#Add OUTPUT layer
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

Learning rate (Adam)

```
#Create optimizer with non-default learning rate
adam_op = tf.keras.optimizers.Adam(learning_rate=0.05)
#Compile the model
model.compile(optimizer=adam_op, loss='binary_crossentropy',
metrics=['accuracy'])

model.fit(trainX,trainY,
          validation_data=(testX,testY),
          epochs=5,
          batch_size = 32)
```

```
Epoch 1/5
7121/7121 [==============================] - 237s 32ms/step - loss:
0.1736 - accuracy: 0.9969 - val_loss: 0.1495 - val_accuracy: 0.9982
Epoch 2/5
```

```
7121/7121 [==============================] - 218s 31ms/step - loss:
0.2030 - accuracy: 0.9967 - val_loss: 0.0271 - val_accuracy: 0.9982
Epoch 3/5
7121/7121 [==============================] - 210s 29ms/step - loss:
0.1973 - accuracy: 0.9969 - val_loss: 0.2044 - val_accuracy: 0.9982
Epoch 4/5
7121/7121 [==============================] - 209s 29ms/step - loss:
0.1794 - accuracy: 0.9967 - val_loss: 0.3411 - val_accuracy: 0.9982
Epoch 5/5
7121/7121 [==============================] - 209s 29ms/step - loss:
0.1891 - accuracy: 0.9967 - val_loss: 0.0908 - val_accuracy: 0.9982

<keras.callbacks.History at 0x7f9c887acb90>
```

change the batch size to 16

```
 model.fit(trainX,trainY,
          validation_data=(testX,testY),
          epochs=5,
          batch_size = 16)

Epoch 1/5
14241/14241 [==============================] - 356s 25ms/step - loss:
0.1753 - accuracy: 0.9966 - val_loss: 0.0666 - val_accuracy: 0.9982
Epoch 2/5
14241/14241 [==============================] - 375s 26ms/step - loss:
0.1563 - accuracy: 0.9967 - val_loss: 0.2147 - val_accuracy: 0.9907
Epoch 3/5
14241/14241 [==============================] - 366s 26ms/step - loss:
0.1590 - accuracy: 0.9968 - val_loss: 0.0894 - val_accuracy: 0.9982
Epoch 4/5
14241/14241 [==============================] - 355s 25ms/step - loss:
0.1330 - accuracy: 0.9968 - val_loss: 0.1069 - val_accuracy: 0.9982
Epoch 5/5
14241/14241 [==============================] - 352s 25ms/step - loss:
0.1180 - accuracy: 0.9967 - val_loss: 0.3135 - val_accuracy: 0.9982

<keras.callbacks.History at 0x7f9c80574250>
```

Learning rate(SGD)

```
#Create optimizer with non-default learning rate
sgd_op = tf.keras.optimizers.SGD(learning_rate=0.05)
#Compile the model
model.compile(optimizer=sgd_op_op, loss='binary_crossentropy',
metrics=['accuracy'])

model.fit(trainX,trainY,
          validation_data=(testX,testY),
```

```
          epochs=5,
          batch_size = 32)
```

```
Epoch 1/5
7121/7121 [==============================] - 216s 30ms/step - loss:
0.0704 - accuracy: 0.9972 - val_loss: 0.0175 - val_accuracy: 0.9981
Epoch 2/5
7121/7121 [==============================] - 217s 30ms/step - loss:
0.1771 - accuracy: 0.9969 - val_loss: 0.8759 - val_accuracy: 0.9982
Epoch 3/5
7121/7121 [==============================] - 215s 30ms/step - loss:
0.2103 - accuracy: 0.9968 - val_loss: 0.0435 - val_accuracy: 0.9977
Epoch 4/5
7121/7121 [==============================] - 232s 33ms/step - loss:
0.1441 - accuracy: 0.9969 - val_loss: 0.0595 - val_accuracy: 0.9982
Epoch 5/5
7121/7121 [==============================] - 225s 32ms/step - loss:
0.0750 - accuracy: 0.9969 - val_loss: 0.0709 - val_accuracy: 0.9982

<keras.callbacks.History at 0x7f9c805f3cd0>
```

change batch size to 16

```
model.fit(trainX,trainY,
          validation_data=(testX,testY),
          epochs=5,
          batch_size = 16)
```