In [1]:
```
import os
import pandas as pd
import numpy as np
import matplotlib as plt
import datetime as dt
import seaborn as sns
```

In [2]:
```
import import_ipynb
import matplotlib.pyplot as plt1
```

In [3]:
```
%matplotlib inline
import sklearn
```

In [4]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
```

In [5]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [6]:
```
import statsmodels.api as sm
```

In [7]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [8]:
```
#SETTING WORKING DIRECTORY
os.chdir("E:/data science and machine learning/santander/python file")
```

In [9]:
```
os.getcwd()
```

Out[9]:
```
'E:\\data science and machine learning\\santander\\python file'
```

In [10]:
```
#GETTING THE FILE FROM HDD
sdf=pd.read_csv("train.csv",sep=',')
```

In [11]:
```
type(sdf)
```

Out[11]:
```
pandas.core.frame.DataFrame
```

In [12]:
```
sdf.columns
```

Out[12]:
```
Index(['ID_code', 'target', 'var_0', 'var_1', 'var_2', 'var_3', 'var_4',
       'var_5', 'var_6', 'var_7',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=202)
```

In [13]:
```
sdf.dtypes
```

Out[13]:

```
ID_code     object
target       int64
var_0      float64
var_1      float64
var_2      float64
var_3      float64
var_4      float64
var_5      float64
var_6      float64
var_7      float64
var_8      float64
var_9      float64
var_10     float64
var_11     float64
var_12     float64
var_13     float64
var_14     float64
var_15     float64
var_16     float64
var_17     float64
var_18     float64
var_19     float64
var_20     float64
var_21     float64
var_22     float64
var_23     float64
var_24     float64
var_25     float64
var_26     float64
var_27     float64
            ...
var_170    float64
var_171    float64
var_172    float64
var_173    float64
var_174    float64
var_175    float64
var_176    float64
var_177    float64
var_178    float64
var_179    float64
var_180    float64
var_181    float64
var_182    float64
var_183    float64
var_184    float64
var_185    float64
var_186    float64
var_187    float64
var_188    float64
var_189    float64
var_190    float64
var_191    float64
var_192    float64
var_193    float64
var_194    float64
var_195    float64
```

```
var_196    float64
var_197    float64
var_198    float64
var_199    float64
Length: 202, dtype: object
```

In [14]:
```
missing_val=pd.DataFrame(sdf.isnull().sum())
```

In [15]:
```
missing_val
```

Out[15]:

	0
ID_code	0
target	0
var_0	0
var_1	0
var_2	0
var_3	0
var_4	0
var_5	0
var_6	0
var_7	0
var_8	0
var_9	0
var_10	0
var_11	0
var_12	0
var_13	0
var_14	0
var_15	0
var_16	0
var_17	0
var_18	0
var_19	0
var_20	0
var_21	0
var_22	0
var_23	0
var_24	0
var_25	0
var_26	0
var_27	0
...	...
var_170	0
var_171	0
var_172	0
var_173	0

	0
var_174	0
var_175	0
var_176	0
var_177	0
var_178	0
var_179	0
var_180	0
var_181	0
var_182	0
var_183	0
var_184	0
var_185	0
var_186	0
var_187	0
var_188	0
var_189	0
var_190	0
var_191	0
var_192	0
var_193	0
var_194	0
var_195	0
var_196	0
var_197	0
var_198	0
var_199	0

202 rows × 1 columns

In [16]:
```
del sdf['ID_code']
```

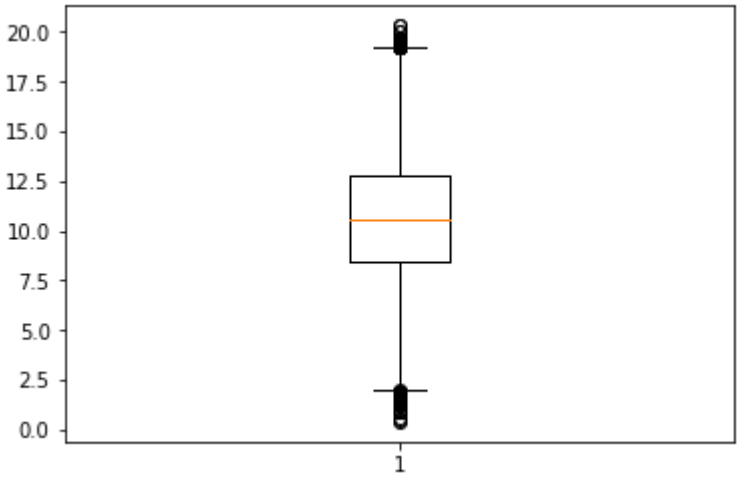
In [17]:
```
sdf.columns
```

Out[17]:
```
Index(['target', 'var_0', 'var_1', 'var_2', 'var_3', 'var_4', 'var_5', 'var_
6',
       'var_7', 'var_8',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=201)
```

In [17]:
```
#BOX PLOT TO CHECK OUTLIERS OF EVERY VARIABLE
cnames=['var_0','var_1','var_5','var_8','var_23','var_45','var_56','var_67','v
ar_76','var_87','var_98','var_123','var_156' ]
cnames1=['var_0','var_1','var_2','var_3','var_4','var_5','var_6','var_7','var_
8','var_9','var_10','var_11','var_12','var_13','var_14','var_15','var_16','var
_17','var_18','var_19','var_20','var_21','var_22','var_23','var_24','var_25',
'var_26','var_27','var_28','var_29','var_30']
```

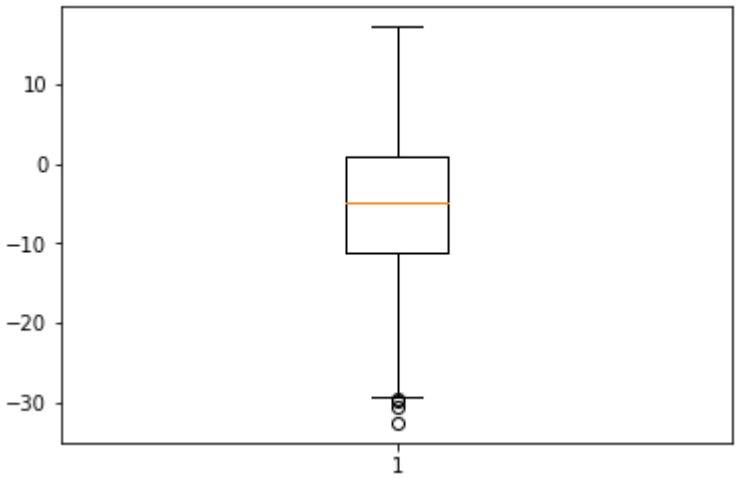
In [19]:
```
plt1.boxplot(sdf['var_0'])
```

Out[19]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x1b0fbb01ef0>,
  <matplotlib.lines.Line2D at 0x1b0fbb01fd0>],
 'caps': [<matplotlib.lines.Line2D at 0x1b0fbae85c0>,
  <matplotlib.lines.Line2D at 0x1b0fbae8908>],
 'boxes': [<matplotlib.lines.Line2D at 0x1b0fbb01a90>],
 'medians': [<matplotlib.lines.Line2D at 0x1b0fbae8c50>],
 'fliers': [<matplotlib.lines.Line2D at 0x1b0fbae8f98>],
 'means': []}
```



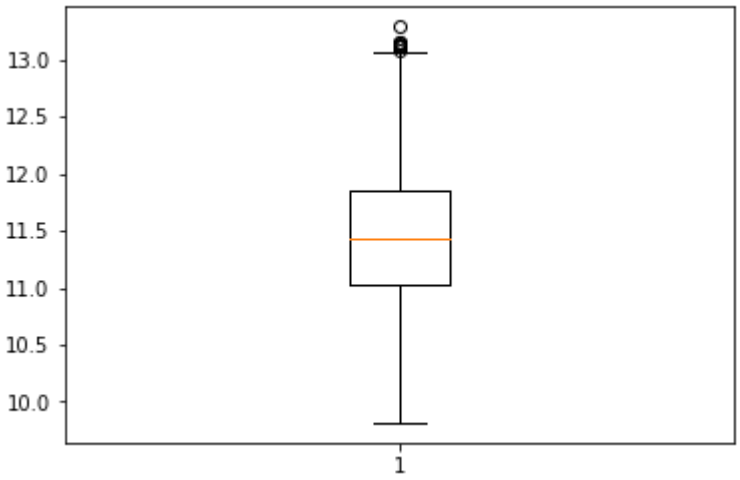
In [25]:
```
plt1.boxplot(sdf['var_5'])
```

Out[25]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x23da7d91668>,
  <matplotlib.lines.Line2D at 0x23da7d919b0>],
 'caps': [<matplotlib.lines.Line2D at 0x23da7d91cf8>,
  <matplotlib.lines.Line2D at 0x23da7886080>],
 'boxes': [<matplotlib.lines.Line2D at 0x23da7d91518>],
 'medians': [<matplotlib.lines.Line2D at 0x23da78863c8>],
 'fliers': [<matplotlib.lines.Line2D at 0x23da7886710>],
 'means': []}
```



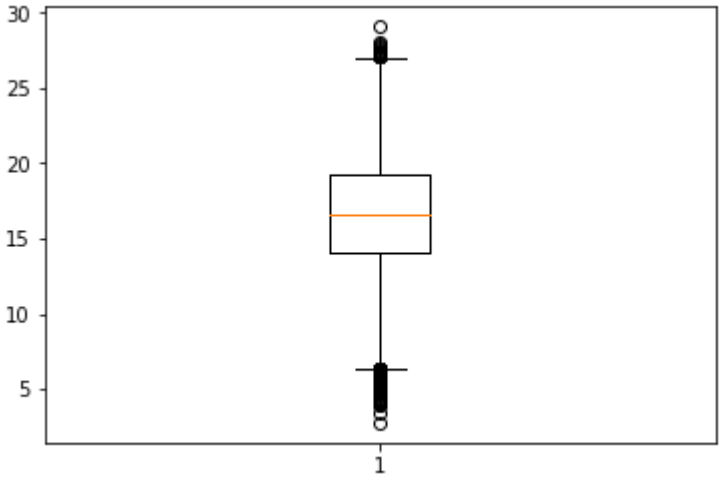
In [26]:
```
plt1.boxplot(sdf['var_34'])
```

Out[26]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x23da8b51ac8>,
  <matplotlib.lines.Line2D at 0x23da8b51e10>],
 'caps': [<matplotlib.lines.Line2D at 0x23da8b47198>,
  <matplotlib.lines.Line2D at 0x23da8b474e0>],
 'boxes': [<matplotlib.lines.Line2D at 0x23da8b51978>],
 'medians': [<matplotlib.lines.Line2D at 0x23da8b47828>],
 'fliers': [<matplotlib.lines.Line2D at 0x23da8b47b70>],
 'means': []}
```



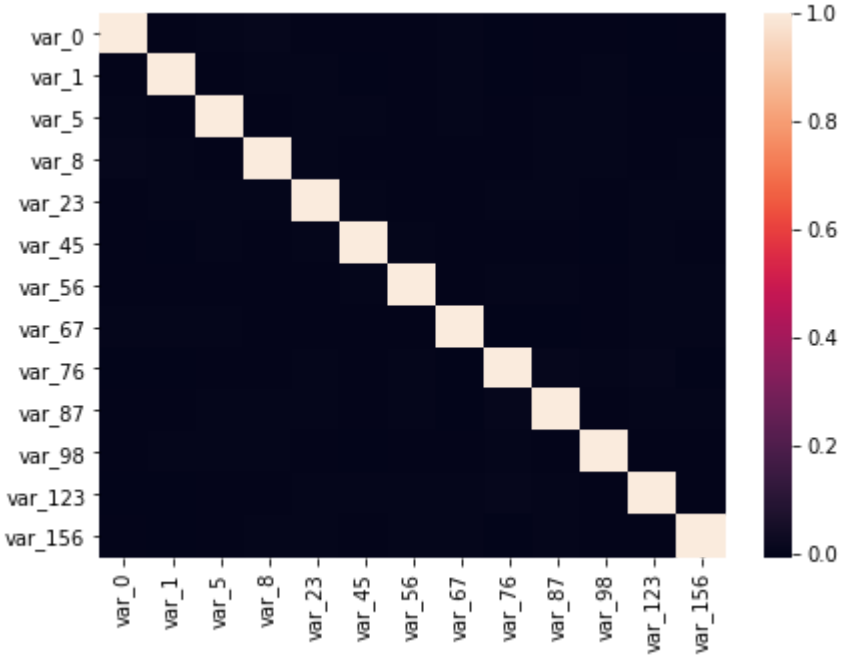
In [27]:
```
plt1.boxplot(sdf['var_56'])
```

Out[27]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x23da8af3ba8>,
  <matplotlib.lines.Line2D at 0x23da8af3ef0>],
 'caps': [<matplotlib.lines.Line2D at 0x23da8aea278>,
  <matplotlib.lines.Line2D at 0x23da8aea5c0>],
 'boxes': [<matplotlib.lines.Line2D at 0x23da8af3a58>],
 'medians': [<matplotlib.lines.Line2D at 0x23da8aea908>],
 'fliers': [<matplotlib.lines.Line2D at 0x23da8aeac50>],
 'means': []}
```



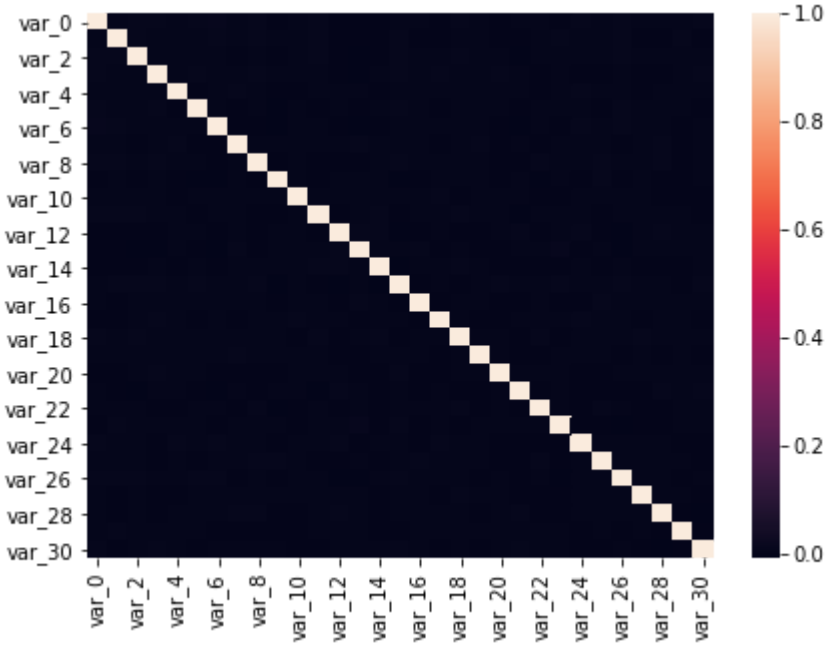
In [18]:
```
#FEATURE SELECTION
sdf_corr=sdf.loc[:,cnames]

f,ax=plt1.subplots(figsize=(7,5))
corr=sdf_corr.corr()
ax = sns.heatmap(corr)
```



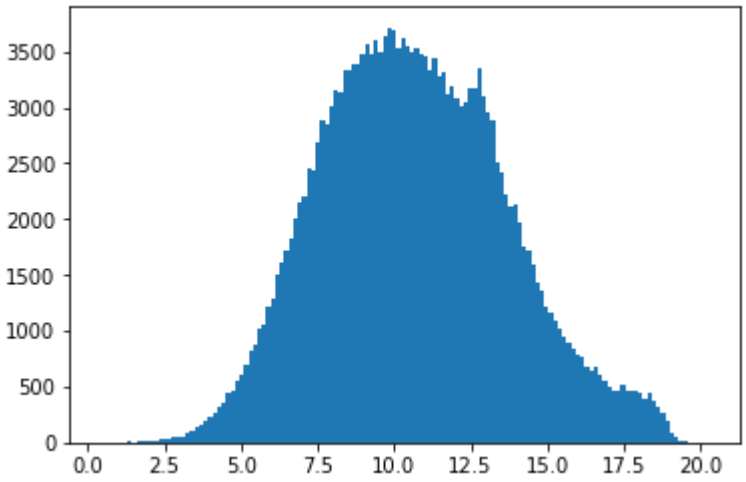
In [21]:
```
#FEATURE SELECTION
sdf_corr=sdf.loc[:,cnames1]

f,ax=plt1.subplots(figsize=(7,5))
corr=sdf_corr.corr()
ax = sns.heatmap(corr)
```



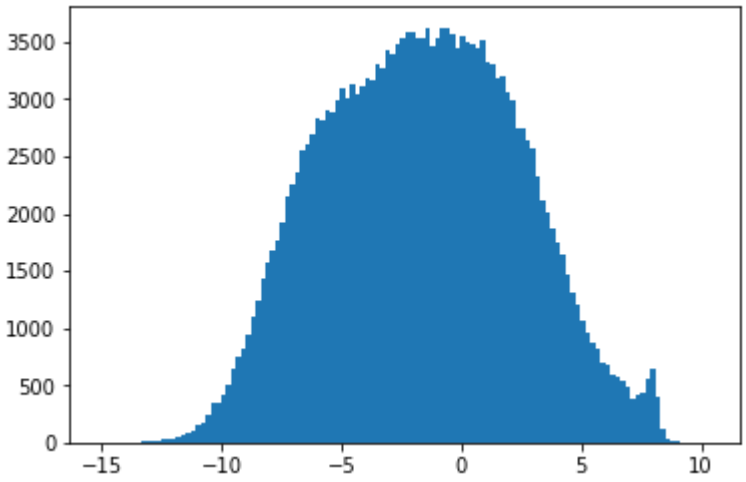
In [19]:
```
#Checking NORMALIZATION
plt1.hist(sdf['var_0'],bins='auto')
```

Out[19]:
```
(array([2.000e+00, 1.000e+00, 0.000e+00, 2.000e+00, 1.000e+00, 3.000e+00,
        6.000e+00, 5.000e+00, 1.000e+01, 7.000e+00, 1.000e+01, 1.600e+01,
        1.500e+01, 2.500e+01, 3.600e+01, 3.700e+01, 4.900e+01, 5.700e+01,
        5.200e+01, 8.000e+01, 1.080e+02, 1.400e+02, 1.640e+02, 1.990e+02,
        2.260e+02, 2.630e+02, 3.140e+02, 3.630e+02, 4.390e+02, 4.550e+02,
        5.460e+02, 6.020e+02, 6.910e+02, 8.250e+02, 8.720e+02, 1.026e+03,
        1.062e+03, 1.212e+03, 1.280e+03, 1.496e+03, 1.614e+03, 1.712e+03,
        1.834e+03, 2.010e+03, 2.145e+03, 2.195e+03, 2.445e+03, 2.437e+03,
        2.689e+03, 2.875e+03, 2.846e+03, 3.001e+03, 3.157e+03, 3.129e+03,
        3.330e+03, 3.327e+03, 3.389e+03, 3.379e+03, 3.473e+03, 3.570e+03,
        3.482e+03, 3.604e+03, 3.494e+03, 3.640e+03, 3.714e+03, 3.696e+03,
        3.524e+03, 3.625e+03, 3.545e+03, 3.488e+03, 3.532e+03, 3.475e+03,
        3.466e+03, 3.326e+03, 3.447e+03, 3.286e+03, 3.311e+03, 3.124e+03,
        3.195e+03, 3.073e+03, 3.016e+03, 3.054e+03, 3.169e+03, 3.162e+03,
        3.347e+03, 3.097e+03, 2.963e+03, 2.880e+03, 2.509e+03, 2.413e+03,
        2.222e+03, 2.121e+03, 2.125e+03, 1.967e+03, 1.754e+03, 1.718e+03,
        1.597e+03, 1.425e+03, 1.365e+03, 1.222e+03, 1.169e+03, 1.082e+03,
        1.015e+03, 9.470e+02, 8.940e+02, 8.340e+02, 7.900e+02, 7.610e+02,
        6.770e+02, 6.490e+02, 6.720e+02, 6.040e+02, 5.470e+02, 5.030e+02,
        4.640e+02, 4.560e+02, 5.240e+02, 4.550e+02, 4.670e+02, 4.540e+02,
        4.480e+02, 3.930e+02, 4.420e+02, 3.820e+02, 3.220e+02, 2.650e+02,
        1.970e+02, 8.100e+01, 4.200e+01, 2.100e+01, 8.000e+00, 3.000e+00,
        1.000e+00, 1.000e+00, 1.000e+00, 2.000e+00]),
 array([ 0.4084    ,  0.55477206,  0.70114412,  0.84751618,  0.99388824,
         1.14026029,  1.28663235,  1.43300441,  1.57937647,  1.72574853,
         1.87212059,  2.01849265,  2.16486471,  2.31123676,  2.45760882,
         2.60398088,  2.75035294,  2.896725  ,  3.04309706,  3.18946912,
         3.33584118,  3.48221324,  3.62858529,  3.77495735,  3.92132941,
         4.06770147,  4.21407353,  4.36044559,  4.50681765,  4.65318971,
         4.79956176,  4.94593382,  5.09230588,  5.23867794,  5.38505   ,
         5.53142206,  5.67779412,  5.82416618,  5.97053824,  6.11691029,
         6.26328235,  6.40965441,  6.55602647,  6.70239853,  6.84877059,
         6.99514265,  7.14151471,  7.28788676,  7.43425882,  7.58063088,
         7.72700294,  7.873375  ,  8.01974706,  8.16611912,  8.31249118,
         8.45886324,  8.60523529,  8.75160735,  8.89797941,  9.04435147,
         9.19072353,  9.33709559,  9.48346765,  9.62983971,  9.77621176,
         9.92258382, 10.06895588, 10.21532794, 10.3617    , 10.50807206,
        10.65444412, 10.80081618, 10.94718824, 11.09356029, 11.23993235,
        11.38630441, 11.53267647, 11.67904853, 11.82542059, 11.97179265,
        12.11816471, 12.26453676, 12.41090882, 12.55728088, 12.70365294,
        12.850025  , 12.99639706, 13.14276912, 13.28914118, 13.43551324,
        13.58188529, 13.72825735, 13.87462941, 14.02100147, 14.16737353,
        14.31374559, 14.46011765, 14.60648971, 14.75286176, 14.89923382,
        15.04560588, 15.19197794, 15.33835   , 15.48472206, 15.63109412,
        15.77746618, 15.92383824, 16.07021029, 16.21658235, 16.36295441,
        16.50932647, 16.65569853, 16.80207059, 16.94844265, 17.09481471,
        17.24118676, 17.38755882, 17.53393088, 17.68030294, 17.826675  ,
        17.97304706, 18.11941912, 18.26579118, 18.41216324, 18.55853529,
        18.70490735, 18.85127941, 18.99765147, 19.14402353, 19.29039559,
        19.43676765, 19.58313971, 19.72951176, 19.87588382, 20.02225588,
        20.16862794, 20.315     ]),
 <a list of 136 Patch objects>)
```


3500
3000
2500
2000
1500
1000
500
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

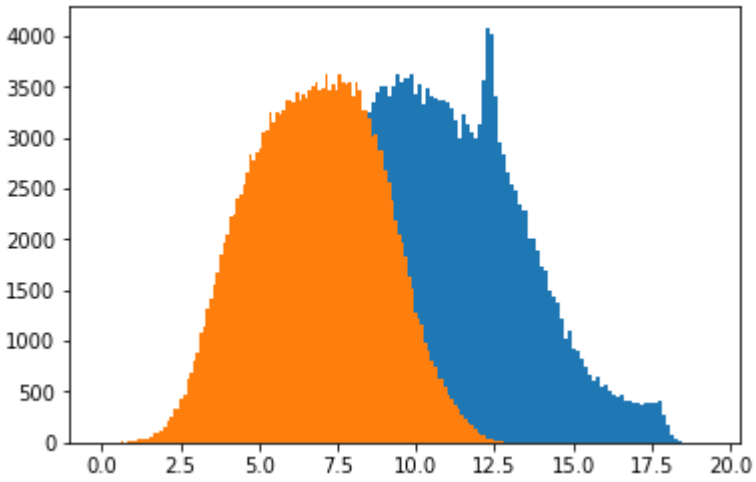
In [35]:
```
plt1.hist(sdf['var_1'],bins='auto')
```

Out[35]:
```
(array([1.000e+00, 1.000e+00, 0.000e+00, 0.000e+00, 3.000e+00, 4.000e+00,
        2.000e+00, 3.000e+00, 7.000e+00, 6.000e+00, 1.000e+01, 1.400e+01,
        3.200e+01, 3.700e+01, 3.500e+01, 5.000e+01, 6.400e+01, 9.000e+01,
        1.080e+02, 1.520e+02, 1.670e+02, 2.390e+02, 3.430e+02, 3.450e+02,
        4.110e+02, 5.070e+02, 6.510e+02, 7.570e+02, 8.130e+02, 9.460e+02,
        1.095e+03, 1.238e+03, 1.427e+03, 1.569e+03, 1.685e+03, 1.770e+03,
        1.920e+03, 2.147e+03, 2.249e+03, 2.363e+03, 2.547e+03, 2.613e+03,
        2.689e+03, 2.825e+03, 2.812e+03, 2.902e+03, 2.883e+03, 2.995e+03,
        3.087e+03, 3.012e+03, 3.131e+03, 3.047e+03, 3.106e+03, 3.187e+03,
        3.174e+03, 3.302e+03, 3.269e+03, 3.431e+03, 3.385e+03, 3.474e+03,
        3.529e+03, 3.586e+03, 3.586e+03, 3.540e+03, 3.532e+03, 3.626e+03,
        3.462e+03, 3.528e+03, 3.626e+03, 3.615e+03, 3.576e+03, 3.453e+03,
        3.551e+03, 3.500e+03, 3.474e+03, 3.447e+03, 3.524e+03, 3.326e+03,
        3.302e+03, 3.179e+03, 3.209e+03, 3.053e+03, 2.988e+03, 2.753e+03,
        2.754e+03, 2.634e+03, 2.578e+03, 2.329e+03, 2.118e+03, 2.007e+03,
        1.863e+03, 1.753e+03, 1.634e+03, 1.467e+03, 1.308e+03, 1.197e+03,
        1.056e+03, 9.590e+02, 8.780e+02, 8.130e+02, 6.940e+02, 6.850e+02,
        5.910e+02, 5.700e+02, 5.400e+02, 4.890e+02, 3.850e+02, 4.250e+02,
        4.290e+02, 5.490e+02, 6.380e+02, 3.920e+02, 1.210e+02, 2.300e+01,
        6.000e+00, 7.000e+00, 3.000e+00, 3.000e+00, 0.000e+00, 0.000e+00,
        2.000e+00, 3.000e+00]),
 array([-15.0434    , -14.8350377 , -14.62667541, -14.41831311,
        -14.20995082, -14.00158852, -13.79322623, -13.58486393,
        -13.37650164, -13.16813934, -12.95977705, -12.75141475,
        -12.54305246, -12.33469016, -12.12632787, -11.91796557,
        -11.70960328, -11.50124098, -11.29287869, -11.08451639,
        -10.8761541 , -10.6677918 , -10.45942951, -10.25106721,
        -10.04270492,  -9.83434262,  -9.62598033,  -9.41761803,
         -9.20925574,  -9.00089344,  -8.79253115,  -8.58416885,
         -8.37580656,  -8.16744426,  -7.95908197,  -7.75071967,
         -7.54235738,  -7.33399508,  -7.12563279,  -6.91727049,
         -6.7089082 ,  -6.5005459 ,  -6.29218361,  -6.08382131,
         -5.87545902,  -5.66709672,  -5.45873443,  -5.25037213,
         -5.04200984,  -4.83364754,  -4.62528525,  -4.41692295,
         -4.20856066,  -4.00019836,  -3.79183607,  -3.58347377,
         -3.37511148,  -3.16674918,  -2.95838689,  -2.75002459,
         -2.5416623 ,  -2.3333    ,  -2.1249377 ,  -1.91657541,
         -1.70821311,  -1.49985082,  -1.29148852,  -1.08312623,
         -0.87476393,  -0.66640164,  -0.45803934,  -0.24967705,
         -0.04131475,   0.16704754,   0.37540984,   0.58377213,
          0.79213443,   1.00049672,   1.20885902,   1.41722131,
          1.62558361,   1.8339459 ,   2.0423082 ,   2.25067049,
          2.45903279,   2.66739508,   2.87575738,   3.08411967,
          3.29248197,   3.50084426,   3.70920656,   3.91756885,
          4.12593115,   4.33429344,   4.54265574,   4.75101803,
          4.95938033,   5.16774262,   5.37610492,   5.58446721,
          5.79282951,   6.0011918 ,   6.2095541 ,   6.41791639,
          6.62627869,   6.83464098,   7.04300328,   7.25136557,
          7.45972787,   7.66809016,   7.87645246,   8.08481475,
          8.29317705,   8.50153934,   8.70990164,   8.91826393,
          9.12662623,   9.33498852,   9.54335082,   9.75171311,
          9.96007541,  10.1684377 ,  10.3768    ]),
 <a list of 122 Patch objects>)
```


3500
3000
2500
2000
1500
1000
500
0
−15
−10
−5
0
5
10

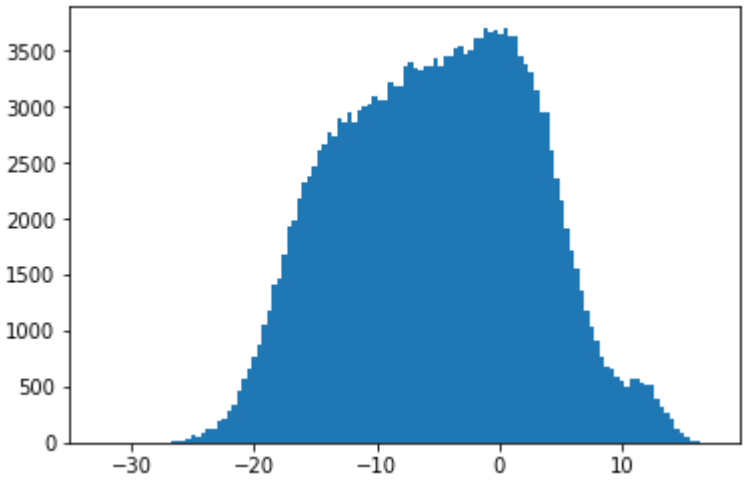
In [20]:
```
plt1.hist(sdf['var_2'],bins='auto')
plt1.hist(sdf['var_3'],bins='auto')
```

Out[20]:
```
(array([3.000e+00, 0.000e+00, 3.000e+00, 3.000e+00, 0.000e+00, 3.000e+00,
        7.000e+00, 5.000e+00, 1.000e+01, 9.000e+00, 1.100e+01, 2.700e+01,
        3.300e+01, 4.100e+01, 3.900e+01, 6.200e+01, 9.600e+01, 9.000e+01,
        1.120e+02, 1.550e+02, 2.180e+02, 2.580e+02, 3.290e+02, 3.390e+02,
        4.340e+02, 4.780e+02, 6.260e+02, 6.940e+02, 8.020e+02, 8.890e+02,
        1.072e+03, 1.137e+03, 1.324e+03, 1.410e+03, 1.550e+03, 1.666e+03,
        1.843e+03, 1.957e+03, 2.037e+03, 2.223e+03, 2.237e+03, 2.396e+03,
        2.448e+03, 2.544e+03, 2.649e+03, 2.838e+03, 2.783e+03, 2.851e+03,
        2.902e+03, 3.042e+03, 3.079e+03, 3.241e+03, 3.155e+03, 3.242e+03,
        3.237e+03, 3.266e+03, 3.375e+03, 3.370e+03, 3.345e+03, 3.437e+03,
        3.359e+03, 3.423e+03, 3.389e+03, 3.463e+03, 3.507e+03, 3.553e+03,
        3.470e+03, 3.494e+03, 3.616e+03, 3.473e+03, 3.529e+03, 3.467e+03,
        3.620e+03, 3.539e+03, 3.533e+03, 3.537e+03, 3.412e+03, 3.551e+03,
        3.474e+03, 3.267e+03, 3.265e+03, 3.185e+03, 3.015e+03, 3.036e+03,
        2.878e+03, 2.867e+03, 2.684e+03, 2.567e+03, 2.377e+03, 2.184e+03,
        2.055e+03, 1.965e+03, 1.820e+03, 1.635e+03, 1.517e+03, 1.277e+03,
        1.213e+03, 1.159e+03, 9.770e+02, 9.010e+02, 8.110e+02, 7.500e+02,
        6.350e+02, 6.180e+02, 5.450e+02, 4.630e+02, 4.250e+02, 3.730e+02,
        3.230e+02, 2.700e+02, 2.300e+02, 1.990e+02, 1.690e+02, 1.400e+02,
        9.700e+01, 6.700e+01, 7.700e+01, 3.900e+01, 3.500e+01, 1.000e+01,
        1.500e+01, 1.100e+01, 5.000e+00, 5.000e+00, 1.000e+00, 2.000e+00]),
 array([-0.0402    ,  0.0647881 ,  0.16977619,  0.27476429,  0.37975238,
         0.48474048,  0.58972857,  0.69471667,  0.79970476,  0.90469286,
         1.00968095,  1.11466905,  1.21965714,  1.32464524,  1.42963333,
         1.53462143,  1.63960952,  1.74459762,  1.84958571,  1.95457381,
         2.0595619 ,  2.16455   ,  2.2695381 ,  2.37452619,  2.47951429,
         2.58450238,  2.68949048,  2.79447857,  2.89946667,  3.00445476,
         3.10944286,  3.21443095,  3.31941905,  3.42440714,  3.52939524,
         3.63438333,  3.73937143,  3.84435952,  3.94934762,  4.05433571,
         4.15932381,  4.2643119 ,  4.3693    ,  4.4742881 ,  4.57927619,
         4.68426429,  4.78925238,  4.89424048,  4.99922857,  5.10421667,
         5.20920476,  5.31419286,  5.41918095,  5.52416905,  5.62915714,
         5.73414524,  5.83913333,  5.94412143,  6.04910952,  6.15409762,
         6.25908571,  6.36407381,  6.4690619 ,  6.57405   ,  6.6790381 ,
         6.78402619,  6.88901429,  6.99400238,  7.09899048,  7.20397857,
         7.30896667,  7.41395476,  7.51894286,  7.62393095,  7.72891905,
         7.83390714,  7.93889524,  8.04388333,  8.14887143,  8.25385952,
         8.35884762,  8.46383571,  8.56882381,  8.6738119 ,  8.7788    ,
         8.8837881 ,  8.98877619,  9.09376429,  9.19875238,  9.30374048,
         9.40872857,  9.51371667,  9.61870476,  9.72369286,  9.82868095,
         9.93366905, 10.03865714, 10.14364524, 10.24863333, 10.35362143,
        10.45860952, 10.56359762, 10.66858571, 10.77357381, 10.8785619 ,
        10.98355   , 11.0885381 , 11.19352619, 11.29851429, 11.40350238,
        11.50849048, 11.61347857, 11.71846667, 11.82345476, 11.92844286,
        12.03343095, 12.13841905, 12.24340714, 12.34839524, 12.45338333,
        12.55837143, 12.66335952, 12.76834762, 12.87333571, 12.97832381,
        13.0833119 , 13.1883    ]),
 <a list of 126 Patch objects>)
```


4000
3500
3000
2500
2000
1500
1000
500
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

In [37]:
```
plt1.hist(sdf['var_5'],bins='auto')
```

Out[37]:
```
(array([1.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00, 0.000e+00,
        1.000e+00, 1.000e+00, 2.000e+00, 0.000e+00, 2.000e+00, 2.000e+00,
        3.000e+00, 3.000e+00, 9.000e+00, 1.500e+01, 1.800e+01, 2.900e+01,
        6.200e+01, 5.600e+01, 8.600e+01, 1.170e+02, 1.250e+02, 1.940e+02,
        2.200e+02, 2.800e+02, 3.320e+02, 4.710e+02, 5.630e+02, 6.570e+02,
        7.730e+02, 8.820e+02, 1.058e+03, 1.181e+03, 1.404e+03, 1.464e+03,
        1.685e+03, 1.925e+03, 1.979e+03, 2.184e+03, 2.322e+03, 2.386e+03,
        2.469e+03, 2.614e+03, 2.674e+03, 2.768e+03, 2.738e+03, 2.901e+03,
        2.870e+03, 2.952e+03, 2.856e+03, 2.980e+03, 3.008e+03, 3.026e+03,
        3.094e+03, 3.067e+03, 3.065e+03, 3.223e+03, 3.180e+03, 3.189e+03,
        3.362e+03, 3.407e+03, 3.356e+03, 3.337e+03, 3.364e+03, 3.374e+03,
        3.430e+03, 3.361e+03, 3.451e+03, 3.462e+03, 3.529e+03, 3.545e+03,
        3.475e+03, 3.514e+03, 3.614e+03, 3.615e+03, 3.711e+03, 3.673e+03,
        3.683e+03, 3.652e+03, 3.700e+03, 3.641e+03, 3.626e+03, 3.448e+03,
        3.383e+03, 3.305e+03, 3.159e+03, 2.950e+03, 2.945e+03, 2.604e+03,
        2.368e+03, 2.157e+03, 1.922e+03, 1.715e+03, 1.557e+03, 1.355e+03,
        1.186e+03, 1.039e+03, 9.140e+02, 7.700e+02, 6.780e+02, 6.640e+02,
        5.920e+02, 5.590e+02, 5.060e+02, 5.770e+02, 5.720e+02, 5.280e+02,
        5.150e+02, 5.130e+02, 3.920e+02, 3.210e+02, 2.650e+02, 2.070e+02,
        1.170e+02, 8.000e+01, 4.200e+01, 2.200e+01, 1.400e+01, 5.000e+00,
        5.000e+00]),
 array([-32.5626    , -32.1509124 , -31.73922479, -31.32753719,
        -30.91584959, -30.50416198, -30.09247438, -29.68078678,
        -29.26909917, -28.85741157, -28.44572397, -28.03403636,
        -27.62234876, -27.21066116, -26.79897355, -26.38728595,
        -25.97559835, -25.56391074, -25.15222314, -24.74053554,
        -24.32884793, -23.91716033, -23.50547273, -23.09378512,
        -22.68209752, -22.27040992, -21.85872231, -21.44703471,
        -21.03534711, -20.6236595 , -20.2119719 , -19.8002843 ,
        -19.38859669, -18.97690909, -18.56522149, -18.15353388,
        -17.74184628, -17.33015868, -16.91847107, -16.50678347,
        -16.09509587, -15.68340826, -15.27172066, -14.86003306,
        -14.44834545, -14.03665785, -13.62497025, -13.21328264,
        -12.80159504, -12.38990744, -11.97821983, -11.56653223,
        -11.15484463, -10.74315702, -10.33146942,  -9.91978182,
         -9.50809421,  -9.09640661,  -8.68471901,  -8.2730314 ,
         -7.8613438 ,  -7.4496562 ,  -7.0379686 ,  -6.62628099,
         -6.21459339,  -5.80290579,  -5.39121818,  -4.97953058,
         -4.56784298,  -4.15615537,  -3.74446777,  -3.33278017,
         -2.92109256,  -2.50940496,  -2.09771736,  -1.68602975,
         -1.27434215,  -0.86265455,  -0.45096694,  -0.03927934,
          0.37240826,   0.78409587,   1.19578347,   1.60747107,
          2.01915868,   2.43084628,   2.84253388,   3.25422149,
          3.66590909,   4.07759669,   4.4892843 ,   4.9009719 ,
          5.3126595 ,   5.72434711,   6.13603471,   6.54772231,
          6.95940992,   7.37109752,   7.78278512,   8.19447273,
          8.60616033,   9.01784793,   9.42953554,   9.84122314,
         10.25291074,  10.66459835,  11.07628595,  11.48797355,
         11.89966116,  12.31134876,  12.72303636,  13.13472397,
         13.54641157,  13.95809917,  14.36978678,  14.78147438,
         15.19316198,  15.60484959,  16.01653719,  16.42822479,
         16.8399124 ,  17.2516    ]),
 <a list of 121 Patch objects>)
```



In [21]:
```
#SAMPLING

#converting the target variable to string(Yes/No)
sdf['target']=sdf['target'].replace(1,'Yes')
sdf['target']=sdf['target'].replace(0,'No')
len(sdf.columns)
```

Out[21]:
```
201
```

In [22]:
```
#divide data  into train & test
x=sdf.values[:,1:201]
y=sdf.values[:,0]
x
```

Out[22]:
```
array([[8.9255, -6.7863, 11.9081, ..., 8.5635, 12.7803, -1.0914],
       [11.5006, -4.1473, 13.8588, ..., 8.7889, 18.355999999999998,
        1.9518],
       [8.6093, -2.7457, 12.0805, ..., 8.2675, 14.7222, 0.3965],
       ...,
       [11.2232, -5.0518, 10.5127, ..., 8.7155, 13.8329, 4.1995],
       [9.7148, -8.6098, 13.6104, ..., 10.0342, 15.5289, -13.9001],
       [10.8762, -5.7105, 12.1183, ..., 8.1857, 12.1284, 0.1385]],
      dtype=object)
```

In [23]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

In [24]:
```
#DECISION TREE
dt=tree.DecisionTreeClassifier(criterion='entropy').fit(x_train,y_train)
```

In [25]:
```
#predict new test case
y_pred=dt.predict(x_test)
```

In [28]:
```
#http://webgraphviz.com/
dotfile=open("pt.dot",'w')
```

In [29]:
```
df=tree.export_graphviz(dt,out_file=dotfile,feature_names=sdf.columns)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-29-f3c71448b0eb> in <module>
      1
----> 2 df=tree.export_graphviz(dt,out_file=dotfile,feature_names=sdf.columns
)

~\Anaconda3\lib\site-packages\sklearn\tree\export.py in export_graphviz(decis
ion_tree, out_file, max_depth, feature_names, class_names, label, filled, lea
ves_parallel, impurity, node_ids, proportion, rotate, rounded, special_charac
ters, precision)
    425                                 "does not match number of features,
 %d"
    426                                 % (len(feature_names),
--> 427                                    decision_tree.n_features_))
    428
    429         # The depth of each node for plotting with 'leaf' option

ValueError: Length of feature_names, 201 does not match number of features, 2
00
```

In [26]:
```
accuracy_score(y_test,y_pred)*100
```

Out[26]: 83.48166666666667

In [27]:
```
#build confusion matrix
CM=confusion_matrix(y_test,y_pred)
```

In [28]:
```
CM=pd.crosstab(y_test,y_pred)
CM
```

Out[28]:

col_0	No	Yes
row_0		
No	48851	5036
Yes	4875	1238

In [128]:
```
#let us save TP,TN,FP,FN
```

In [29]:
```
TN=CM.iloc[0,0]
FN=CM.iloc[1,0]
TP=CM.iloc[1,1]
FP=CM.iloc[0,1]
```

In [30]:
```
Accuracy=((TP+TN)*100)/(TP+TN+FP+FN)
Accuracy#=83.57
```

Out[30]: 83.48166666666667

In [33]:
```
FN
```

Out[33]: 4898

In [31]:
```
FNR=FN*100/(FN+TP)
FNR#=81.14
```

Out[31]: 79.74807786684116

In [32]:
```
FPR=FP*100/(FP+TN)
FPR#=9.18
```

Out[32]: 9.345482212778592

In [33]:
```
Recall=TP*100/(TP+FN)
Recall#=18.85
```

Out[33]: 20.251922133158843

In [34]:
```
Specificity=TN*100/(TN+FP)
Specificity#=90.81
```

Out[34]: 90.65451778722141

In [39]:
```
Precision=TP*100/(TP+FP)
Precision#=18.67
```

Out[39]: 19.73222824354479

In [40]:
```
#LOGISTIC REGRESSION
sdf_logit=pd.DataFrame(sdf)
```

In [41]:
```
type(sdf_logit)
```

Out[41]: pandas.core.frame.DataFrame

In [42]:
```
Sample_Index=np.random.rand(len(sdf_logit))<0.8
```

In [43]:
```
train=sdf_logit[Sample_Index]
```

In [44]:
```
test=sdf_logit[~Sample_Index]
len(test)
```

Out[44]: 40023

In [45]:
```
train_cols=train.columns[1:201]
```

In [45]:
```
train_cols
```

Out[45]:
```
Index(['var_0', 'var_1', 'var_2', 'var_3', 'var_4', 'var_5', 'var_6', 'var_
7',
       'var_8', 'var_9',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=200)
```

In [46]:
```
logit=sm.Logit(train['target'],train[train_cols]).fit()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-46-80d8ab4507a0> in <module>
----> 1 logit=sm.Logit(train['target'],train[train_cols]).fit()

~\Anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py in __ini
t__(self, endog, exog, **kwargs)
    416
    417     def __init__(self, endog, exog, **kwargs):
--> 418         super(BinaryModel, self).__init__(endog, exog, **kwargs)
    419         if (not issubclass(self.__class__, MultinomialModel) and
    420                 not np.all((self.endog >= 0) & (self.endog <= 1))):

~\Anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py in __ini
t__(self, endog, exog, **kwargs)
    169     """
    170     def __init__(self, endog, exog, **kwargs):
--> 171         super(DiscreteModel, self).__init__(endog, exog, **kwargs)
    172         self.raise_on_perfect_prediction = True
    173

~\Anaconda3\lib\site-packages\statsmodels\base\model.py in __init__(self, end
og, exog, **kwargs)
    210
    211     def __init__(self, endog, exog=None, **kwargs):
--> 212         super(LikelihoodModel, self).__init__(endog, exog, **kwargs)
    213         self.initialize()
    214

~\Anaconda3\lib\site-packages\statsmodels\base\model.py in __init__(self, end
og, exog, **kwargs)
     62         hasconst = kwargs.pop('hasconst', None)
     63         self.data = self._handle_data(endog, exog, missing, hasconst,
---> 64                                       **kwargs)
     65         self.k_constant = self.data.k_constant
     66         self.exog = self.data.exog

~\Anaconda3\lib\site-packages\statsmodels\base\model.py in _handle_data(self,
endog, exog, missing, hasconst, **kwargs)
     85
     86     def _handle_data(self, endog, exog, missing, hasconst, **kwargs):
---> 87         data = handle_data(endog, exog, missing, hasconst, **kwargs)
     88         # kwargs arrays could have changed, easier to just attach her
e
     89         for key in kwargs:

~\Anaconda3\lib\site-packages\statsmodels\base\data.py in handle_data(endog,
 exog, missing, hasconst, **kwargs)
    631     klass = handle_data_class_factory(endog, exog)
    632     return klass(endog, exog=exog, missing=missing, hasconst=hascons
t,
--> 633                  **kwargs)

~\Anaconda3\lib\site-packages\statsmodels\base\data.py in __init__(self, endo
g, exog, missing, hasconst, **kwargs)
     74             self.orig_endog = endog
     75             self.orig_exog = exog
```

```
---> 76             self.endog, self.exog = self._convert_endog_exog(endog, e
xog)
     77
     78         # this has side-effects, attaches k_constant and const_idx

~\Anaconda3\lib\site-packages\statsmodels\base\data.py in _convert_endog_exog
(self, endog, exog)
    472         exog = exog if exog is None else np.asarray(exog)
    473         if endog.dtype == object or exog is not None and exog.dtype =
= object:
--> 474             raise ValueError("Pandas data cast to numpy dtype of obje
ct. "
    475                              "Check input data with np.asarray(dat
a).")
    476         return super(PandasData, self)._convert_endog_exog(endog, exo
g)

ValueError: Pandas data cast to numpy dtype of object. Check input data with
 np.asarray(data).
```

In [165]:
```
logit.summary()
```

Out[165]:

Logit Regression Results

Dep. Variable:	target	**No. Observations:**	159907
Model:	Logit	**Df Residuals:**	159707
Method:	MLE	**Df Model:**	199
Date:	Mon, 22 Jul 2019	**Pseudo R-squ.:**	0.2923
Time:	19:49:38	**Log-Likelihood:**	-36847.
converged:	True	**LL-Null:**	-52065.
		LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
var_0	0.0569	0.003	18.122	0.000	0.051	0.063
var_1	0.0417	0.002	17.485	0.000	0.037	0.046
var_2	0.0686	0.004	19.015	0.000	0.062	0.076
var_3	0.0146	0.005	3.082	0.002	0.005	0.024
var_4	0.0286	0.006	4.807	0.000	0.017	0.040
var_5	0.0129	0.001	10.461	0.000	0.010	0.015
var_6	0.2635	0.011	23.830	0.000	0.242	0.285
var_7	0.0012	0.003	0.417	0.676	-0.004	0.007
var_8	0.0174	0.003	5.965	0.000	0.012	0.023
var_9	-0.1087	0.008	-13.986	0.000	-0.124	-0.093
var_10	0.0006	0.002	0.316	0.752	-0.003	0.004
var_11	0.0127	0.002	7.825	0.000	0.009	0.016
var_12	-1.1141	0.050	-22.310	0.000	-1.212	-1.016
var_13	-0.0385	0.002	-18.587	0.000	-0.043	-0.034
var_14	-0.0080	0.004	-1.863	0.063	-0.016	0.000
var_15	0.1397	0.023	5.951	0.000	0.094	0.186
var_16	0.0095	0.004	2.517	0.012	0.002	0.017
var_17	-0.0004	0.001	-0.277	0.782	-0.003	0.002
var_18	0.0179	0.001	14.590	0.000	0.016	0.020
var_19	0.0037	0.001	3.057	0.002	0.001	0.006
var_20	-0.0118	0.002	-7.152	0.000	-0.015	-0.009
var_21	-0.0229	0.001	-19.354	0.000	-0.025	-0.021
var_22	0.0716	0.003	21.310	0.000	0.065	0.078
var_23	-0.1765	0.018	-9.567	0.000	-0.213	-0.140
var_24	0.0285	0.003	11.170	0.000	0.024	0.034
var_25	0.1794	0.034	5.320	0.000	0.113	0.246
var_26	0.0331	0.002	20.528	0.000	0.030	0.036

var_27	-0.0041	0.006	-0.636	0.525	-0.017	0.008
var_28	-0.0981	0.012	-7.933	0.000	-0.122	-0.074
var_29	0.0110	0.004	2.970	0.003	0.004	0.018
var_30	-0.0010	0.001	-0.828	0.408	-0.003	0.001
var_31	-0.0428	0.005	-9.502	0.000	-0.052	-0.034
var_32	0.0404	0.004	10.814	0.000	0.033	0.048
var_33	-0.0351	0.002	-15.592	0.000	-0.040	-0.031
var_34	-0.3194	0.018	-17.881	0.000	-0.354	-0.284
var_35	0.0236	0.002	12.659	0.000	0.020	0.027
var_36	-0.0420	0.003	-13.554	0.000	-0.048	-0.036
var_37	0.0123	0.004	2.853	0.004	0.004	0.021
var_38	0.0008	0.002	0.332	0.740	-0.004	0.005
var_39	-0.0006	0.002	-0.249	0.804	-0.005	0.004
var_40	0.0208	0.001	17.856	0.000	0.018	0.023
var_41	-4.916e-05	0.002	-0.030	0.976	-0.003	0.003
var_42	-0.0436	0.014	-3.138	0.002	-0.071	-0.016
var_43	-0.2852	0.031	-9.138	0.000	-0.346	-0.224
var_44	-0.0297	0.002	-18.512	0.000	-0.033	-0.027
var_45	-0.0031	0.000	-6.798	0.000	-0.004	-0.002
var_46	0.0056	0.003	1.662	0.097	-0.001	0.012
var_47	0.0037	0.001	3.987	0.000	0.002	0.005
var_48	0.0097	0.001	11.383	0.000	0.008	0.011
var_49	0.0123	0.001	9.987	0.000	0.010	0.015
var_50	-0.0636	0.014	-4.556	0.000	-0.091	-0.036
var_51	0.0095	0.001	8.106	0.000	0.007	0.012
var_52	0.0196	0.002	10.092	0.000	0.016	0.023
var_53	0.2859	0.013	22.704	0.000	0.261	0.311
var_54	-0.0064	0.001	-5.534	0.000	-0.009	-0.004
var_55	0.0111	0.002	6.521	0.000	0.008	0.014
var_56	-0.0325	0.003	-11.955	0.000	-0.038	-0.027
var_57	-0.0695	0.012	-5.703	0.000	-0.093	-0.046
var_58	-0.0184	0.002	-8.173	0.000	-0.023	-0.014
var_59	-0.0417	0.011	-3.668	0.000	-0.064	-0.019
var_60	0.0053	0.002	2.330	0.020	0.001	0.010
var_61	0.0025	0.001	3.011	0.003	0.001	0.004
var_62	0.0271	0.005	5.696	0.000	0.018	0.036
var_63	-0.0169	0.003	-5.443	0.000	-0.023	-0.011

var_64	-0.0280	0.007	-4.298	0.000	-0.041	-0.015
var_65	0.0077	0.003	2.994	0.003	0.003	0.013
var_66	0.0672	0.009	7.796	0.000	0.050	0.084
var_67	0.0195	0.001	14.865	0.000	0.017	0.022
var_68	6.1138	0.304	20.124	0.000	5.518	6.709
var_69	0.0058	0.002	2.370	0.018	0.001	0.011
var_70	0.0073	0.001	8.957	0.000	0.006	0.009
var_71	0.4152	0.036	11.442	0.000	0.344	0.486
var_72	-0.0093	0.002	-3.768	0.000	-0.014	-0.004
var_73	-0.0032	0.001	-2.459	0.014	-0.006	-0.001
var_74	0.0046	0.001	6.613	0.000	0.003	0.006
var_75	-0.0207	0.002	-13.013	0.000	-0.024	-0.018
var_76	-0.0262	0.001	-21.725	0.000	-0.029	-0.024
var_77	-0.0140	0.003	-5.468	0.000	-0.019	-0.009
var_78	0.0831	0.005	17.061	0.000	0.074	0.093
var_79	0.0099	0.007	1.338	0.181	-0.005	0.024
var_80	-0.0244	0.001	-19.089	0.000	-0.027	-0.022
var_81	-0.1118	0.004	-27.188	0.000	-0.120	-0.104
var_82	0.0084	0.001	7.379	0.000	0.006	0.011
var_83	-0.0075	0.001	-6.468	0.000	-0.010	-0.005
var_84	0.0060	0.002	3.832	0.000	0.003	0.009
var_85	-0.0162	0.002	-6.526	0.000	-0.021	-0.011
var_86	-0.0173	0.001	-14.048	0.000	-0.020	-0.015
var_87	-0.0214	0.002	-12.477	0.000	-0.025	-0.018
var_88	-0.0216	0.004	-5.581	0.000	-0.029	-0.014
var_89	0.0396	0.003	14.657	0.000	0.034	0.045
var_90	0.0071	0.001	9.646	0.000	0.006	0.009
var_91	0.8871	0.063	14.056	0.000	0.763	1.011
var_92	-0.0359	0.002	-15.510	0.000	-0.040	-0.031
var_93	-0.1982	0.018	-11.269	0.000	-0.233	-0.164
var_94	0.0599	0.003	17.214	0.000	0.053	0.067
var_95	0.2026	0.015	13.127	0.000	0.172	0.233
var_96	0.0017	0.001	1.459	0.144	-0.001	0.004
var_97	0.0037	0.001	4.830	0.000	0.002	0.005
var_98	-0.0212	0.014	-1.573	0.116	-0.048	0.005
var_99	0.1104	0.005	21.404	0.000	0.100	0.121
var_100	0.0010	0.001	0.914	0.360	-0.001	0.003

var_101	-0.0068	0.002	-3.490	0.000	-0.011	-0.003
var_102	-0.0076	0.001	-6.820	0.000	-0.010	-0.005
var_103	-0.0361	0.052	-0.691	0.490	-0.139	0.066
var_104	-0.0472	0.005	-9.578	0.000	-0.057	-0.038
var_105	0.0908	0.011	8.052	0.000	0.069	0.113
var_106	0.0554	0.005	10.863	0.000	0.045	0.065
var_107	-0.0186	0.001	-14.548	0.000	-0.021	-0.016
var_108	-0.8158	0.056	-14.637	0.000	-0.925	-0.707
var_109	-0.0349	0.002	-15.751	0.000	-0.039	-0.031
var_110	0.0535	0.003	21.398	0.000	0.049	0.058
var_111	0.0804	0.009	9.029	0.000	0.063	0.098
var_112	0.0534	0.006	8.724	0.000	0.041	0.065
var_113	-0.0109	0.002	-5.003	0.000	-0.015	-0.007
var_114	-0.0876	0.010	-8.954	0.000	-0.107	-0.068
var_115	-0.0635	0.004	-17.290	0.000	-0.071	-0.056
var_116	-0.0502	0.006	-8.564	0.000	-0.062	-0.039
var_117	0.0010	0.001	1.344	0.179	-0.000	0.002
var_118	0.0159	0.001	14.452	0.000	0.014	0.018
var_119	0.0221	0.002	9.581	0.000	0.018	0.027
var_120	-0.0027	0.001	-3.357	0.001	-0.004	-0.001
var_121	-0.0763	0.006	-13.414	0.000	-0.087	-0.065
var_122	-0.0275	0.002	-14.744	0.000	-0.031	-0.024
var_123	-0.0207	0.002	-13.279	0.000	-0.024	-0.018
var_124	0.0067	0.004	1.896	0.058	-0.000	0.014
var_125	0.3149	0.030	10.411	0.000	0.256	0.374
var_126	0.0191	0.012	1.535	0.125	-0.005	0.044
var_127	-0.0409	0.003	-13.307	0.000	-0.047	-0.035
var_128	0.0262	0.003	8.782	0.000	0.020	0.032
var_129	-0.0051	0.002	-2.163	0.031	-0.010	-0.000
var_130	0.1339	0.012	11.570	0.000	0.111	0.157
var_131	-0.2107	0.021	-9.971	0.000	-0.252	-0.169
var_132	-0.0597	0.007	-8.987	0.000	-0.073	-0.047
var_133	0.4742	0.025	18.602	0.000	0.424	0.524
var_134	0.0098	0.002	6.267	0.000	0.007	0.013
var_135	0.0100	0.001	7.879	0.000	0.008	0.012
var_136	-0.0013	0.001	-1.402	0.161	-0.003	0.001
var_137	0.0106	0.001	9.732	0.000	0.008	0.013

var_138	0.0125	0.002	5.857	0.000	0.008	0.017
var_139	-0.0304	0.001	-24.385	0.000	-0.033	-0.028
var_140	0.0121	0.002	6.122	0.000	0.008	0.016
var_141	-0.0161	0.001	-11.167	0.000	-0.019	-0.013
var_142	-0.0119	0.002	-6.969	0.000	-0.015	-0.009
var_143	-0.0150	0.003	-4.553	0.000	-0.021	-0.009
var_144	0.0824	0.010	7.859	0.000	0.062	0.103
var_145	0.0263	0.002	10.608	0.000	0.021	0.031
var_146	-0.0815	0.004	-21.531	0.000	-0.089	-0.074
var_147	0.0164	0.001	12.603	0.000	0.014	0.019
var_148	-0.8567	0.048	-17.793	0.000	-0.951	-0.762
var_149	-0.0140	0.001	-15.005	0.000	-0.016	-0.012
var_150	-0.0391	0.004	-9.961	0.000	-0.047	-0.031
var_151	0.0242	0.002	9.981	0.000	0.019	0.029
var_152	-0.0101	0.003	-3.157	0.002	-0.016	-0.004
var_153	-0.0074	0.005	-1.535	0.125	-0.017	0.002
var_154	-0.0294	0.002	-15.136	0.000	-0.033	-0.026
var_155	0.0222	0.002	13.292	0.000	0.019	0.025
var_156	-0.0675	0.010	-6.661	0.000	-0.087	-0.048
var_157	0.0179	0.002	10.356	0.000	0.014	0.021
var_158	-0.0015	0.001	-1.236	0.216	-0.004	0.001
var_159	0.0126	0.002	5.375	0.000	0.008	0.017
var_160	-0.0012	0.001	-1.333	0.183	-0.003	0.001
var_161	0.0566	0.044	1.274	0.203	-0.031	0.144
var_162	0.0739	0.007	10.872	0.000	0.061	0.087
var_163	0.0197	0.002	10.828	0.000	0.016	0.023
var_164	0.0235	0.002	13.199	0.000	0.020	0.027
var_165	-0.0362	0.002	-18.870	0.000	-0.040	-0.032
var_166	-0.4848	0.026	-18.658	0.000	-0.536	-0.434
var_167	0.0122	0.001	9.878	0.000	0.010	0.015
var_168	0.0145	0.003	4.675	0.000	0.008	0.021
var_169	-0.4099	0.026	-15.634	0.000	-0.461	-0.359
var_170	0.0380	0.002	17.473	0.000	0.034	0.042
var_171	0.0083	0.002	4.585	0.000	0.005	0.012
var_172	-0.0145	0.001	-13.028	0.000	-0.017	-0.012
var_173	0.0238	0.002	14.590	0.000	0.021	0.027
var_174	-0.0270	0.001	-20.061	0.000	-0.030	-0.024

var_175	0.0283	0.003	8.520	0.000	0.022	0.035
var_176	0.0039	0.001	3.017	0.003	0.001	0.006
var_177	-0.0492	0.004	-13.290	0.000	-0.056	-0.042
var_178	-0.0075	0.001	-6.621	0.000	-0.010	-0.005
var_179	0.0598	0.003	17.663	0.000	0.053	0.066
var_180	0.0208	0.002	11.323	0.000	0.017	0.024
var_181	0.0415	0.007	5.887	0.000	0.028	0.055
var_182	-0.0038	0.001	-3.510	0.000	-0.006	-0.002
var_183	-0.0046	0.002	-2.130	0.033	-0.009	-0.000
var_184	0.0173	0.001	16.736	0.000	0.015	0.019
var_185	0.0008	0.002	0.367	0.713	-0.003	0.005
var_186	-0.0305	0.003	-10.039	0.000	-0.036	-0.025
var_187	0.0049	0.001	5.827	0.000	0.003	0.007
var_188	-0.0295	0.002	-12.050	0.000	-0.034	-0.025
var_189	0.0227	0.010	2.286	0.022	0.003	0.042
var_190	0.0392	0.002	18.493	0.000	0.035	0.043
var_191	0.0509	0.003	16.072	0.000	0.045	0.057
var_192	-0.0962	0.007	-14.573	0.000	-0.109	-0.083
var_193	-0.0149	0.002	-6.117	0.000	-0.020	-0.010
var_194	-0.0181	0.003	-5.871	0.000	-0.024	-0.012
var_195	0.0683	0.007	10.125	0.000	0.055	0.081
var_196	0.0145	0.002	8.145	0.000	0.011	0.018
var_197	-0.1264	0.010	-12.058	0.000	-0.147	-0.106
var_198	-0.0572	0.003	-18.047	0.000	-0.063	-0.051
var_199	0.0075	0.001	8.019	0.000	0.006	0.009

In [166]:
```
test['Actual_prob']=logit.predict(test[train_cols])
```

```
C:\Users\user\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
```

In [169]:
```
test['ActualVal']=1
test.loc[test.Actual_prob<0.5,'ActualVal']=0
```

```
C:\Users\user\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
C:\Users\user\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/st
able/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

In [170]:
```
test.head()
```

Out[170]:

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	...	var_
2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	-4.9193	...	1.6
3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	-5.8609	...	0.7
13	1	16.3699	1.5934	16.7395	7.3330	12.1450	5.9004	4.8222	20.9729	1.1064	...	4.3
16	0	5.0615	0.2689	15.1325	3.6587	13.5276	-6.5477	5.2757	9.8710	2.5569	...	0.5
19	0	4.4090	-0.7863	15.1828	8.0631	11.2831	-0.7356	6.3801	16.0218	2.4621	...	2.5

5 rows × 203 columns

In [171]:
```
CM=pd.crosstab(test['target'],test['ActualVal'])
```

In [172]:
```
CM
```

Out[172]:

ActualVal	0	1
target		
0	35529	494
1	2942	1128

In [173]:
```
TN=CM.iloc[0,0]
FN=CM.iloc[1,0]
TP=CM.iloc[1,1]
FP=CM.iloc[0,1]
```

In [174]:
```
Accuracy=((TP+TN)*100)/(TP+TN+FP+FN)
#Accuracy=91.42
```

Out[174]: 91.42992542339061

In [175]:
```
FNR=FN*100/(FN+TP)
FNR#=72.28
```

Out[175]: 72.28501228501229

In [176]:
```
FPR=FP*100/(FP+TN)
#FPR=1.37
```

Out[176]: 1.3713460844460483

In [178]:
```
Recall=TP*100/(TP+FN)
#Recall=27.71
```

Out[178]: 27.714987714987714

In [179]:
```
Specificity=TN*100/(TN+FP)
#Specificity=98.63
```

Out[179]: 98.62865391555395

In [180]:
```
Precision=TP*100/(TP+FP)
Precision#=9.76
```

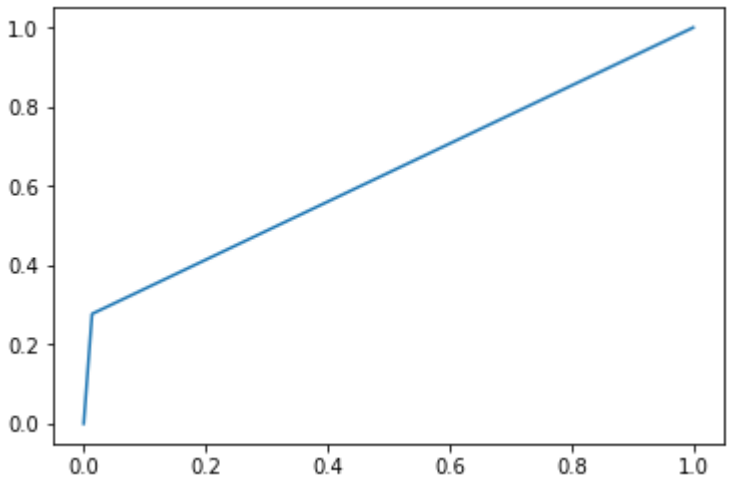
Out[180]: 69.54377311960542

In [ ]:
```

```

In [194]:
```
from sklearn import metrics
fpr,tpr,_=metrics.roc_curve(test['target'],test['ActualVal'])
```

In [196]:
```
plt1.plot(fpr,tpr)
plt1.show()
```



In [76]:
```
auc=np.trapz(tpr,fpr)
#auc=0.631
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-76-c348a34830c8> in <module>
----> 1 auc=np.trapz(tpr,fpr)
      2 #auc=0.631

NameError: name 'tpr' is not defined
```

In [75]:
```
x_train
```

Out[75]:
```
array([[10.9498, 4.2601, 4.7294, ..., 8.6465, 18.0604, 1.18],
       [12.765, -6.5636, 12.8731, ..., 8.697000000000001, 12.5575,
        -0.2376],
       [12.9363, -9.9686, 6.7736, ..., 9.9644, 13.6574, 5.7519],
       ...,
       [6.3436, -4.4653, 6.3046, ..., 7.7537, 13.0901, -1.6363],
       [6.5696, 5.2355, 9.7173, ..., 9.7726, 16.9061, 3.6166],
       [9.5376, -1.729, 5.9327, ..., 8.1401, 11.6016, 12.228]],
      dtype=object)
```

In [47]:
```
#NAive Bayes
from sklearn.naive_bayes import GaussianNB
```

In [48]:
```
#Naive Bayes Implementation
NB_model=GaussianNB().fit(x_train,y_train)
```

In [49]:
```
#predict test class
NB_Predictions=NB_model.predict(x_test)
```

In [50]:
```
#Build Confusion Matrix
CM=pd.crosstab(y_test,NB_Predictions)
```

In [51]:
```
CM
```

Out[51]:

col_0	No	Yes
row_0		
No	53099	865
Yes	3812	2224

In [52]:
```
TN=CM.iloc[0,0]
FN=CM.iloc[1,0]
TP=CM.iloc[1,1]
FP=CM.iloc[0,1]
```

In [53]:
```
Accuracy=((TP+TN)*100)/(TP+TN+FP+FN)
Accuracy#=92.205
```

Out[53]: 92.205

In [54]:
```
FNR=FN*100/(FN+TP)
FNR#=63.15
```

Out[54]: 63.154406891981445

In [56]:
```
FPR=FP*100/(FP+TN)
FPR#=1.60
```

Out[56]: 1.6029204654955156

In [57]:
```
Recall=TP*100/(TP+FN)
Recall#=36.84
```

Out[57]: 36.845593108018555

In [58]:
```
Specificity=TN*100/(TN+FP)
Specificity#=98.39
```

Out[58]: 98.39707953450448

In [59]:
```
Precision=TP*100/(TP+FP)
Precision#=71.99
```

Out[59]: 71.99741016510197

In [61]:
```
#GETTING THE TEST FILE FROM HDD
sdf1=pd.read_csv("test.csv",sep=',')
```

In [63]:
```
sdf1.columns
```

Out[63]:
```
Index(['ID_code', 'var_0', 'var_1', 'var_2', 'var_3', 'var_4', 'var_5',
       'var_6', 'var_7', 'var_8',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=201)
```

In [64]:
```
missing_val=pd.DataFrame(sdf1.isnull().sum())
```

In [65]:
```
missing_val
```

Out[65]:

	0
ID_code	0
var_0	0
var_1	0
var_2	0
var_3	0
var_4	0
var_5	0
var_6	0
var_7	0
var_8	0
var_9	0
var_10	0
var_11	0
var_12	0
var_13	0
var_14	0
var_15	0
var_16	0
var_17	0
var_18	0
var_19	0
var_20	0
var_21	0
var_22	0
var_23	0
var_24	0
var_25	0
var_26	0
var_27	0
var_28	0
...	...
var_170	0
var_171	0
var_172	0
var_173	0

	0
var_174	0
var_175	0
var_176	0
var_177	0
var_178	0
var_179	0
var_180	0
var_181	0
var_182	0
var_183	0
var_184	0
var_185	0
var_186	0
var_187	0
var_188	0
var_189	0
var_190	0
var_191	0
var_192	0
var_193	0
var_194	0
var_195	0
var_196	0
var_197	0
var_198	0
var_199	0

201 rows × 1 columns

In [66]:
```
del sdf1['ID_code']
```

In [67]:
```
sdf1.columns
```

Out[67]:
```
Index(['var_0', 'var_1', 'var_2', 'var_3', 'var_4', 'var_5', 'var_6', 'var_
7',
       'var_8', 'var_9',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=200)
```

In [68]:
```
#predict test class
NB_Predictions=NB_model.predict(sdf1)
```

In [70]:
```
NB_Predictions
```

Out[70]:
```
array(['No', 'No', 'No', ..., 'No', 'No', 'No'], dtype='<U3')
```

In [71]:
```
NB_Predictions=pd.DataFrame(NB_Predictions)
```

In [74]:
```
sdf_final=pd.concat([sdf1.reset_index(drop=True),NB_Predictions],axis=1)
```

In [76]:
```
sdf_final
```

Out[76]:

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9
0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	8.8100
1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	5.9739
2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	8.3442
3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	7.4578
4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	7.1437
5	5.9862	-2.2913	8.6058	7.0685	14.2465	-8.6761	4.2467	14.7632	1.8790	7.2842
6	8.4624	-6.1065	7.3603	8.2627	12.0104	-7.2073	4.1670	13.0809	-4.3004	6.3181
7	17.3035	-2.4212	13.3989	8.3998	11.0777	9.6449	5.9596	17.8477	-4.8068	7.4643
8	6.9856	0.8402	13.7161	4.7749	8.6784	-13.7607	4.3386	14.5843	2.5883	7.2215
9	10.3811	-6.9348	14.6690	9.0941	11.9058	-10.8018	3.4508	20.2816	-1.4112	6.7401
10	8.3431	-4.1427	9.1985	9.8229	11.2494	2.9678	5.5184	15.6290	2.8032	8.9513
11	10.6137	-2.1898	8.9090	3.8014	13.8602	-5.9802	5.5515	15.4716	-0.1714	7.6178
12	12.7465	-4.9467	15.5490	6.4580	13.7572	-25.5371	4.4893	15.1682	3.1754	7.5722
13	11.7836	1.9979	10.3347	7.8857	13.1020	5.0167	4.9548	23.6527	3.5911	5.8546
14	7.0360	1.6797	9.3865	3.2605	10.7569	-8.0802	4.7885	15.0583	0.6459	4.8661
15	14.8595	-4.5378	13.6483	5.6480	9.9144	1.5190	5.0358	13.4524	-2.5419	9.4450
16	14.1732	-5.1490	9.7591	3.7316	10.3700	-21.9202	7.7130	18.8749	0.4680	7.8453
17	9.0936	-8.7414	17.1160	6.0126	9.2144	-3.6761	4.6477	20.1053	1.7687	7.9974
18	15.7875	0.1671	10.7782	3.8521	9.1190	11.0196	6.1113	18.4368	-1.0728	7.0586
19	13.3874	1.0716	8.8767	7.8374	11.6404	6.2512	4.8837	18.2178	4.3871	9.5648
20	8.0259	-4.6740	8.6431	2.2198	11.4555	-14.0227	6.9192	17.8559	0.4283	6.5548
21	14.3356	0.2317	9.5604	5.7603	10.3184	-6.4721	4.6898	13.7783	1.8342	9.2284
22	10.4255	-6.1758	12.4846	7.9845	9.7032	-14.5969	4.4173	19.3606	-0.5899	6.9213
23	12.3322	-6.3835	7.2471	5.0403	10.0875	-1.5252	5.8230	17.9494	-3.8454	6.2356
24	14.1844	-9.1044	9.7453	9.2638	9.3302	2.6818	5.4711	18.5414	2.2065	8.3338
25	10.0029	0.2530	7.5335	6.9343	11.6866	-6.5147	6.7327	19.8941	-6.6497	8.0114
26	6.9056	-4.8626	11.8932	5.3693	11.2551	-18.9716	5.5991	18.9809	5.5612	7.8337
27	8.7562	-3.0647	11.7990	9.2162	10.9847	-22.4902	4.2991	13.9800	3.3233	7.7593
28	9.7243	-1.5151	11.5582	5.7360	12.1907	6.9664	4.4125	17.4770	-3.9683	7.5912
29	13.2430	1.2738	10.4245	3.1863	11.4951	-1.4755	5.1005	17.1687	1.7115	9.1463
...	...	...	...	...	...	...	...	...	...	...
199970	12.7260	-1.6706	12.3598	9.1114	10.1868	-9.5857	5.3494	23.6362	2.0626	6.2033
199971	9.4700	-6.7655	12.6591	9.1842	11.8260	0.0264	5.0633	20.7034	2.4171	8.2646
199972	13.3243	1.0870	8.4555	3.6929	11.2423	1.3986	4.4765	19.1021	-2.6573	8.6612
199973	14.2830	-1.8421	11.3664	8.5772	8.8645	-13.8986	4.1603	19.4591	5.6445	9.2011

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9
199974	4.5171	-5.2068	7.6007	8.1426	10.4433	-17.2322	4.4205	20.3407	-1.0196	5.6569
199975	13.4796	2.7000	10.9653	9.1581	13.2959	-3.0995	5.1483	20.9766	1.2932	7.6743
199976	12.6337	-6.9793	9.8703	9.9180	10.8092	2.5809	6.7764	18.3443	4.1498	7.8825
199977	10.8078	-4.6108	9.0021	9.8910	12.4514	-3.7566	4.2958	19.9677	0.8806	8.2828
199978	9.9317	-2.2815	11.1707	5.6826	12.7396	-4.0659	6.2569	12.7697	-2.1645	8.9019
199979	10.5933	-1.2672	13.6817	6.3789	12.8649	-5.4964	6.4800	13.5986	4.0315	8.8308
199980	13.4136	5.3912	9.6202	8.5025	12.0951	11.3431	5.8323	12.1429	-3.1511	6.6322
199981	7.9218	-5.7464	11.4171	6.7972	11.6260	-8.7730	5.4601	12.1401	5.1918	8.2214
199982	7.2189	1.6606	10.4651	4.4382	10.5562	-5.2083	4.7197	10.7883	-8.1002	7.6637
199983	11.8527	5.4321	12.7268	10.2392	12.4740	-14.6939	6.6544	14.1274	-0.4182	8.7811
199984	12.7445	-6.1135	9.9046	7.5790	14.8852	4.5083	6.3353	21.5936	-4.0102	8.5375
199985	14.8983	2.1302	7.4747	7.1744	11.8252	13.1758	5.1614	13.7914	-4.8184	6.5496
199986	19.2884	-2.8384	11.9149	6.6611	12.3112	12.9244	5.6492	16.0449	5.3597	8.2981
199987	11.2942	3.6321	15.3300	6.6904	10.9223	-5.6537	6.0221	11.7757	-0.5163	8.9841
199988	6.4535	-2.1707	10.7623	8.1571	7.9365	4.6091	4.9564	11.4483	2.8938	6.5602
199989	9.0436	-3.0491	10.8737	7.8789	11.0275	-10.1812	6.1978	16.4603	4.4421	9.1971
199990	5.5416	1.7340	9.6938	5.0126	11.3049	-15.9906	5.0937	17.7960	-3.1050	6.9197
199991	8.7935	-4.0646	9.9480	8.6947	11.0497	-0.5129	5.6410	21.5338	5.6578	5.3441
199992	16.4229	-5.0254	13.1385	5.4599	13.1347	-2.6212	4.7829	14.7163	0.0779	8.9048
199993	14.6764	-8.1066	7.1167	2.4138	10.3845	-11.9327	4.7563	16.0455	0.4510	8.7944
199994	8.2964	-2.3119	11.2139	9.1357	8.5339	4.0350	5.7000	11.0102	4.9089	8.3779
199995	13.1678	1.0136	10.4333	6.7997	8.5974	-4.1641	4.8579	14.7625	-2.7239	6.9937
199996	9.7171	-9.1462	7.3443	9.1421	12.8936	3.0191	5.6888	18.8862	5.0915	6.3545
199997	11.6360	2.2769	11.2074	7.7649	12.6796	11.3224	5.3883	18.3794	1.6603	5.7341
199998	13.5745	-0.5134	13.6584	7.4855	11.2241	-11.3037	4.1959	16.8280	5.3208	8.9032
199999	10.4664	1.8070	10.2277	6.0654	10.0258	1.0789	4.8879	14.4892	-0.5902	7.8362

200000 rows × 201 columns

In [77]:
```
#Writing the final dataset into HDD
sdf_final.to_csv("santander_final.csv",index=False)
```

In [ ]:
```
```