```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import preprocessing
```

```
train_transaction = pd.read_csv("/content/drive/My Drive/train_transaction.csv",index_col= 'Transact
train_identity = pd.read_csv("/content/drive/My Drive/train_identity.csv",index_col= 'TransactionID
test_transaction = pd.read_csv("/content/drive/My Drive/test_transaction.csv",index_col= 'Transacti
test_identity = pd.read_csv("/content/drive/My Drive/test_identity.csv",index_col= 'TransactionID')
```

```
associated_train_data = np.sum(train_transaction.index.isin(train_identity.index.unique()))
associated_test_data = np.sum(test_transaction.index.isin(test_identity.index.unique()))

train_association = associated_train_data/len(train_transaction.index)*100
test_association = associated_test_data/len(test_transaction.index)*100
```

```
print(train_association,'%','have associated identity information in training data')
print(test_association,'%','have associated identity information in testing data')
```

```
#merging data

train_data = train_transaction.merge(train_identity, how = "left",left_index = True, right_index = 
test_data = test_transaction.merge(test_identity, how = "left", left_index = True, right_index = Tru
```

```
train_data.head()
```

```
train_data = train_data.filter(['TransactionID','DeviceType','DeviceInfo','TransactionDT','Transact
                    'addr1','addr2','card4','card6','dist1','dist2','P_emaildomain','R_emaildomain',

test_data = test_data.filter(['TransactionID','DeviceType','DeviceInfo','TransactionDT','Transactio
                    ,'addr1','addr2','card4','card6','dist1','dist2','P_emaildomain','R_emaildomain
```

```
features_train = train_data.drop('isFraud', axis = 1)
label_train = train_data['isFraud']
features_test = test_data.copy()
```

```
for feature in features_train:
  if features_train[feature].dtype == 'object' or features_test[feature].dtype == 'object':
    le = preprocessing.LabelEncoder()
    le.fit(list(features_train[feature].values) + list(features_test[feature].values))
    features_train[feature] = le.transform(list(features_train[feature].values))
    features_test[feature] = le.transform(list(features_test[feature].values))
```

```
features_train.head()
```

```
#replacing NaN values with -1
#features_train.fillna(-1,inplace = True)
#features_test.fillna(-1,inplace = True)
```

```
features_train.fillna(features_train.mean(), inplace = True)
features_test.fillna(features_test.mean(), inplace = True)
```

```
x_train = features_train
y_train = label_train
x_test = features_test
```

```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(solver='lbfgs')
clf.fit(x_train, y_train)
preds = clf.predict(x_test)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:947: Convergence
  "of iterations.", ConvergenceWarning)
```

```
sub = pd.read_csv('/content/drive/My Drive/sample_submission.csv', index_col='TransactionID')
sub['isFraud'] = preds
sub.to_csv('myPrediction.csv')
```

```
from sklearn.tree import DecisionTreeClassifier

clf2 = DecisionTreeClassifier()
clf2.fit(x_train, y_train)
preds2 = clf2.predict(x_test)
sub = pd.read_csv('/content/drive/My Drive/sample_submission.csv', index_col='TransactionID')
sub['isFraud'] = preds2
sub.to_csv('myPrediction8.csv')
```

```
from sklearn.ensemble import RandomForestClassifier
clf3 = RandomForestClassifier()
clf3.fit(x_train, y_train)
preds3 = clf3.predict(x_test)
sub = pd.read_csv('/content/drive/My Drive/sample_submission.csv', index_col='TransactionID')
sub['isFraud'] = preds3
sub.to_csv('myPrediction9.csv')
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/ensemble/forest.py:245: FutureWarning: Th
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
df = pd.DataFrame(train_transaction)
print("Correlation Matrix")
print(df.corr())
print()
```

```
Correlation Matrix
                 isFraud  TransactionDT  ...      V338      V339
isFraud         1.000000       0.013103  ... -0.019356 -0.014663
TransactionDT   0.013103       1.000000  ...  0.210240  0.167524
TransactionAmt  0.011320       0.011920  ...  0.082064  0.105996
card1          -0.013640       0.010625  ...  0.118885  0.091271
card2           0.003388      -0.019202  ...  0.055566  0.043249
card3           0.154151      -0.011222  ...  0.001970  0.001800
card5          -0.033580      -0.024132  ... -0.233342 -0.182758
addr1           0.005596      -0.000051  ...  0.034816  0.026908
addr2          -0.030387       0.051972  ...  0.003999  0.003237
dist1           0.021522      -0.027295  ...       NaN       NaN
dist2          -0.019054      -0.026860  ... -0.024142 -0.018521
C1              0.030570      -0.049318  ... -0.001844  0.000636
C2              0.037229      -0.051126  ... -0.003692 -0.000788
C3             -0.006833      -0.007546  ...  0.019139  0.026224
C4              0.030382      -0.053104  ... -0.002631  0.000040
C5             -0.030754       0.023800  ...       NaN       NaN
C6              0.020909      -0.046612  ... -0.006151 -0.002677
C7              0.028160      -0.055402  ...       NaN       NaN
C8              0.032139      -0.056288  ...  0.008200  0.007812
C9             -0.031703       0.032732  ...       NaN       NaN
C10             0.028396      -0.057734  ... -0.002643 -0.001228
C11             0.027484      -0.050181  ... -0.000769  0.001452
C12             0.031905      -0.054738  ...       NaN       NaN
C13            -0.011146      -0.015022  ...  0.076952  0.058586
C14             0.007921      -0.039721  ...  0.001260  0.002883
D1             -0.067193       0.074031  ...  0.016462  0.009810
D2             -0.083583       0.027109  ... -0.226233 -0.208795
D3             -0.046271      -0.007200  ... -0.254541 -0.234136
D4             -0.067216       0.059797  ... -0.046530 -0.040483
D5             -0.064638       0.001767  ... -0.146667 -0.141390
...                  ...            ...  ...       ...       ...
V310            0.011071       0.061206  ...  0.453877  0.338102
V311            0.001300      -0.000028  ...  0.003111  0.003675
V312            0.037578       0.039969  ...  0.318674  0.245361
V313            0.041494       0.029642  ...  0.195048  0.145857
V314            0.038535       0.042646  ...  0.627145  0.476594
V315            0.048298       0.028649  ...  0.178293  0.134305
V316           -0.002960       0.060300  ...  0.680669  0.472927
V317            0.005010       0.059212  ...  0.750318  0.515846
V318            0.000997       0.065552  ...  0.763977  0.537431
V319            0.000061       0.018373  ...  0.488169  0.581797
V320            0.004961       0.055153  ...  0.817420  0.749238
V321            0.001677       0.041065  ...  0.695989  0.713821
V322           -0.021541       0.207888  ...  0.678443  0.471132
V323           -0.023329       0.230193  ...  0.759181  0.525305
V324           -0.024006       0.234674  ...  0.767756  0.544056
V325            0.007792       0.091039  ...  0.240886  0.209176
V326           -0.006838       0.257295  ...  0.617962  0.455429
V327           -0.001050       0.248494  ...  0.648252  0.485927
V328           -0.011053       0.247530  ...  0.721737  0.601530
V329           -0.023099       0.252599  ...  0.799920  0.586808
V330           -0.021164       0.258603  ...  0.780555  0.611463
V331           -0.021982       0.205671  ...  0.752075  0.572187
V332           -0.023468       0.226312  ...  0.800310  0.579686
V333           -0.024134       0.231072  ...  0.823754  0.620566
```

```
V334          -0.000451      0.005762  ...  0.054911  0.055864
V335          -0.005456      0.184407  ...  0.552533  0.411911
V336          -0.002402      0.105783  ...  0.353950  0.274392
V337          -0.005702      0.075892  ...  0.742652  0.907378
V338          -0.019356      0.210240  ...  1.000000  0.940009
V339          -0.014663      0.167524  ...  0.940009  1.000000

[379 rows x 379 columns]
```