#Required library and packages

import sys

import numpy

import pandas

import matplotlib

import seaborn

import scipy

import sklearn

#if u wanna check the version of library and packages that u use

print('Python: {}'.format(sys.version))

print('Numpy: {}'.format(numpy.\_\_version\_\_))

print('Pandas: {}'.format(pandas.\_\_version\_\_))

print('Matplotlib: {}'.format(matplotlib.\_\_version\_\_))

print('Seaborn: {}'.format(seaborn.\_\_version\_\_))

print('Scipy: {}'.format(scipy.\_\_version\_\_))

print('Sklearn: {}'.format(sklearn.\_\_version\_\_))

#

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#dataset to train the machine

data = pd.read\_csv("creditcard.csv")

#exploring the data

data.columns

print(data.shape)

print(data.describe())

data.hist(figsize=(20,20))

plt.show()

Fraud = data[data['Class'] == 1]

Valid = data[data['Class'] == 0]

outlier\_fraction = len(Fraud)/float(len(Valid))

print(outlier\_fraction)

print('Fraud Transaction {}'.format(len(Fraud)))

print('Valid Transaction {}'.format(float(len(Valid))))

#plotting a Heatmap

corrmat = data.corr()

fig = plt.figure(figsize=(15,12))

sns.heatmap(corrmat, vmax=.8, square = True)

plt.show()

#setting up target

columns = data.columns.tolist()

columns = [c for c in columns if c not in ['Class']]

target = 'Class'

X = data[columns]

Y = data[target]

print(X.shape)

print(Y.shape)

#applying the Algorithm

from sklearn.metrics import classification\_report, accuracy\_score

from sklearn.ensemble import IsolationForest

from sklearn.neighbors import LocalOutlierFactor

state = 1

classifiers = {

"Isolation Forest": IsolationForest(max\_samples = len(X),

contamination = outlier\_fraction,

random\_state = state),

"Local Outlier Factor" : LocalOutlierFactor(

n\_neighbors = 20,

contamination = outlier\_fraction

)

}

n\_outliers = len(Fraud)

for i, (clf\_name, clf) in enumerate(classifiers.items()):

if clf\_name == "Local Outlier Factor":

y\_pred = clf.fit\_predict(X)

score\_pred = clf.negative\_outlier\_factor\_

else:

clf.fit(X)

score\_pred = clf.decision\_function(X)

y\_pred = clf.predict(X)

y\_pred[y\_pred == 1] = 0

y\_pred[y\_pred == -1] = 1

n\_errors = (y\_pred != Y).sum()

#Result

print('{}:{}'.format(clf\_name,n\_errors))

print(accuracy\_score(Y, y\_pred))

print(classification\_report(Y, y\_pred))