#import the necessary packages

import sys

import numpy

import pandas

import matplotlib

import seaborn

import scipy

import sklearn

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 the necessary packages

import csv

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

#load the dataset from csv file

data=pd.read\_csv("C:\\Users\\Asus\\Documents\\Python\\creditcard.csv")

#explore the dataset

print(data.columns)

print(data.shape)

print(data.describe())

data = data.sample(frac = 0.1, random\_state = 1)

print(data.shape)

#plot histogram of each parameter

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

plt.show()

#determine the number of fraud cases in data set

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

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

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

print(outlier\_fraction)

print('Fraud cases: {}'.format(len(Fraud)))

print('Valid cases: {}'.format(len(Valid)))

#Correlation Matrix

corrmat = data.corr()

fig = plt.figure(figsize = (12,9))

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

plt.show()

#get all the columns from the data frame

columns = data.columns.tolist()

#filter the columns to remove data we do not want

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

#store the target we'll be predicting on

target = "Class"

#store the variable we'll be predicting on

X = data[columns]

Y = data[target]

#print the shape of X and Y

print(X.shape)

print(Y.shape)

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

from sklearn.ensemble import IsolationForest

from sklearn.neighbors import LocalOutlierFactor

#define a random state

state = 1

#define the outlier detection methods

classifiers = {

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

contamination = outlier\_fraction,

random\_state = state),

"Local Outlier Factor":LocalOutlierFactor(

n\_neighbors = 20,

contamination = outlier\_fraction)

}

#fit the model

n\_outliers = len(Fraud)

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

#fit the data and tag outliers

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

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

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

else:

clf.fit(X)

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

y\_pred = clf.predict(X)

#Reshape the prediction values to 0 for valid , 1 for fraud

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

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

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

#Run classification metrics

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

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

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