import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.utils import shuffle

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

transactions = pd.read\_csv('datas.csv')

sample = transactions[transactions['Class']==0]

normcorr= sample.corr()

sns.heatmap(normcorr, cbar = True, square = True, annot=False, fmt= '.2f',annot\_kws={'size': 15},

cmap= 'coolwarm')

plt.show()

fraud = transactions[transactions['Class']==1]

fraudcorr = fraud.corr()

fraudcorr.head()

sns.heatmap(fraudcorr, cbar = True, square = True, annot=False, fmt= '.2f',annot\_kws={'size': 15},

cmap= 'coolwarm')

plt.show()

print('V9 - V10')

plt.scatter(fraud['V9'], fraud['V10'],s=1, color='r')

plt.scatter(sample['V9'], sample['V10'], s=1, color='g')

plt.show()

plt.clf()

print('V16-V17')

plt.scatter(sample['V16'], sample['V17'], s=1, color = 'g')

plt.scatter(fraud['V16'], fraud['V17'], s=1, color = 'r')

plt.show()

plt.clf()

print('V17 - V18')

plt.scatter(sample['V18'], sample['V17'], s=1, color = 'g')

plt.scatter(fraud['V18'], fraud['V17'], s=1, color = 'r')

plt.show()

plt.clf()

print('V1 - V3')

plt.scatter(sample['V1'], sample['V3'], s=1, color = 'g')

plt.scatter(fraud['V1'], fraud['V3'], s=1, color = 'r')

plt.show()

plt.clf()

print('V1 - V3')

plt.scatter(sample['V1'], sample['V3'], s=1, color = 'g')

plt.scatter(fraud['V1'], fraud['V3'], s=1, color = 'r')

plt.show()

plt.clf()

print('V1 - V2')

plt.scatter(sample['V1'], sample['V2'], s=1, color = 'g')

plt.scatter(fraud['V1'], fraud['V2'], s=1, color = 'r')

plt.show()

plt.clf()

transactions = transactions[['Class', 'V9', 'V10', 'V16', 'V17', 'V18','Amount']]

transactions.options.mode.chained\_assignment = None

sample = transactions[transactions['Class']==0]

fraud = transactions[transactions['Class'] == 1]

# need a very small but random sample of the legitimate data since it is massively over represented.

ignore\_me, sample = train\_test\_split(sample, test\_size = 0.01)

sample.options.mode.chained\_assignment = None

sample = pd.concat([sample, fraud])

# Break into train and test units.

train, test = train\_test\_split(sample, test\_size = 0.3)

trainy = train['Class']

testy = test['Class']

train.drop('Class', 1, inplace = True)

test.drop('Class', 1, inplace = True)

scaler = StandardScaler()

scaler.fit(train)

train = scaler.transform(train)

test = scaler.transform(test)

clf = SVC()

clf.fit(train, trainy)

outcome = list(clf.predict(test))

testy = list(testy)

count = 0

falsepos = 0

truepos = 0

falseneg = 0

trueneg = 0

for i in range (1,len(testy)):

if (outcome[i]==1):

if (testy[i] == 1):

truepos = truepos + 1

else:

falsepos = falsepos + 1

else:

if (testy[i] == 0):

trueneg = trueneg + 1

else:

falseneg = falseneg +1

count = count + 1

precision = truepos / (truepos + falsepos)

recall = truepos / (truepos + falseneg)

F1 = 2\*((precision \* recall ) / (precision + recall))

print("Precision = " + str(precision))

print("Recall = " + str(recall))

print("F1 = " + str(F1))