##1. importing libraries

import numpy as np

import pandas as pd

import matplotlib. pyplot as plt

import matplotlib. pyplot as plt

import seaborn as sns

import scipy

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

from sklearn.ensemble import IsolationForest

from sklearn.neighbors import LocalOutlierFactor

from sklearn.svm import OneClassSVM

from pylab import rcParams

rcParams['figure.figsize'] = 14, 8

RANDOM\_SEED = 42

LABELS = ["Normal", "Fraud"]

##2.loading the data

data = pd.read\_csv(r"C:\Users\Admin\Downloads\creditcard.csv.zip")

data.head()

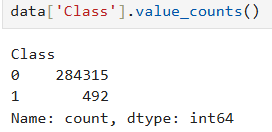
data.tail()

data.info()

data.isnull().sum()

##3.Analysis of class distribution

data['Class'].value\_counts()



count\_classes = pd.value\_counts(data['Class'], sort = True)

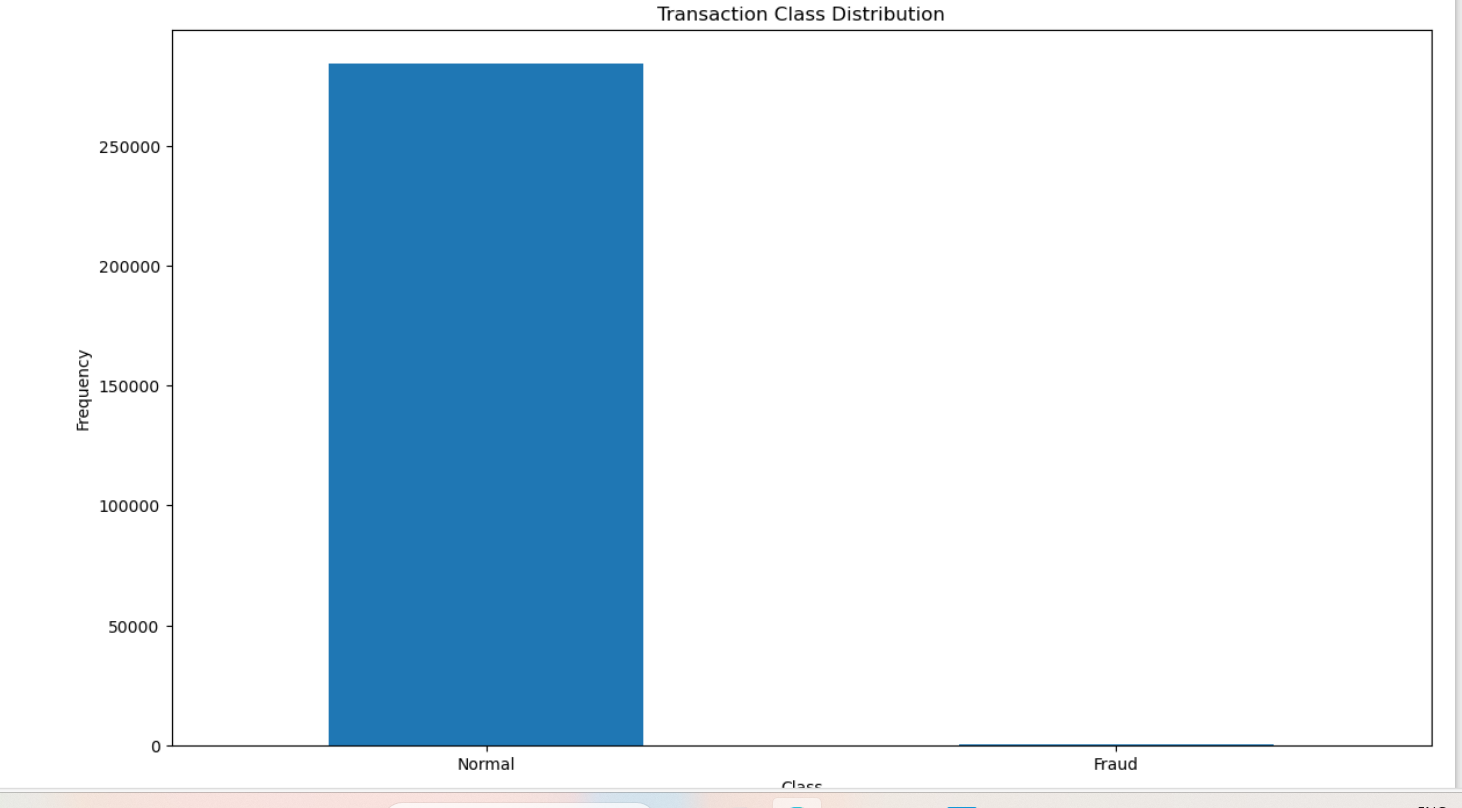
count\_classes.plot(kind = 'bar', rot=0)

plt.title("Transaction Class Distribution")

plt.xticks(range(2), LABELS)

plt.xlabel("Class")

plt.ylabel("Frequency")



## Normal and Fraud Transaction

fraud = data[data['Class']==1]

normal = data[data['Class']==0]

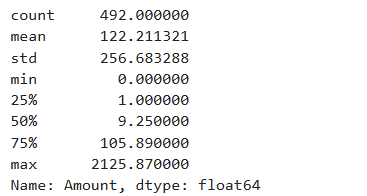
print(fraud.shape)



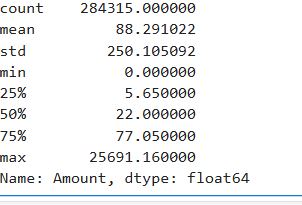
print(normal.shape)



fraud.Amount.describe()



normal.Amount.describe()



f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)

f.suptitle('Amount per transaction by class')

bins = 50

ax1.hist(fraud.Amount, bins = bins)

ax1.set\_title('fraud')

ax2.hist(normal.Amount, bins = bins)

ax2.set\_title('normal')

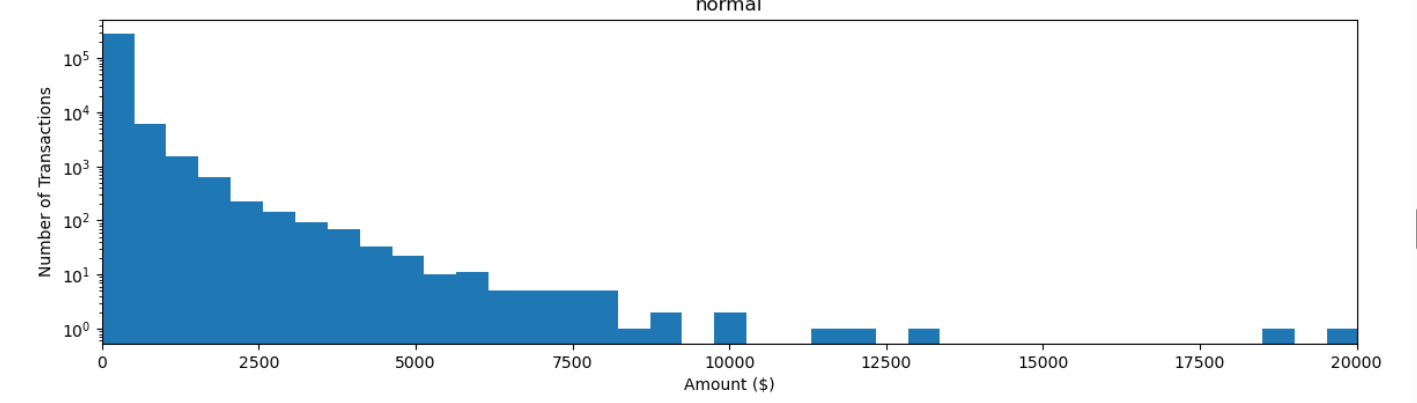
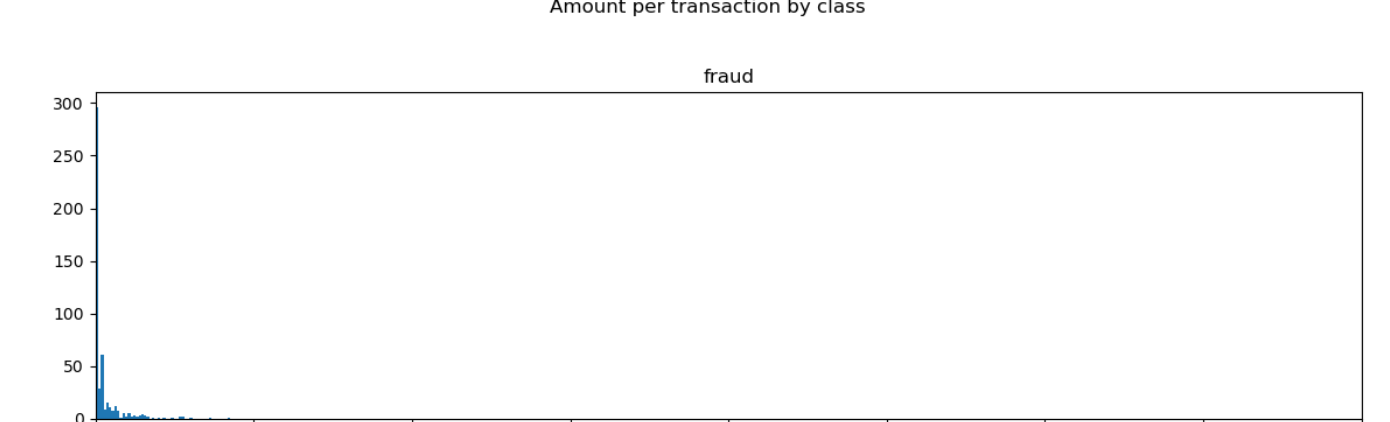
plt.xlabel('Amount ($)')

plt.ylabel('Number of Transactions')

plt.xlim((0, 20000))

plt.yscale('log')

plt.show();



f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)

f.suptitle('Time of transaction vs Amount by class')

ax1.scatter(fraud.Time, fraud.Amount)

ax1.set\_title('fraud')

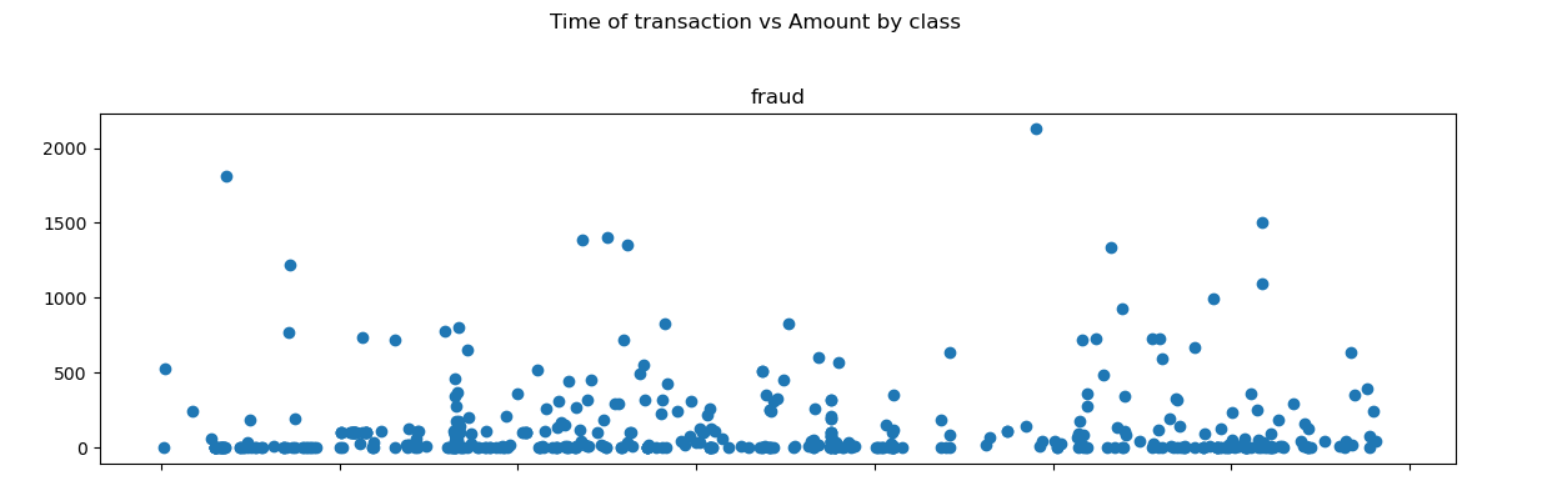
ax2.scatter(normal.Time, normal.Amount)

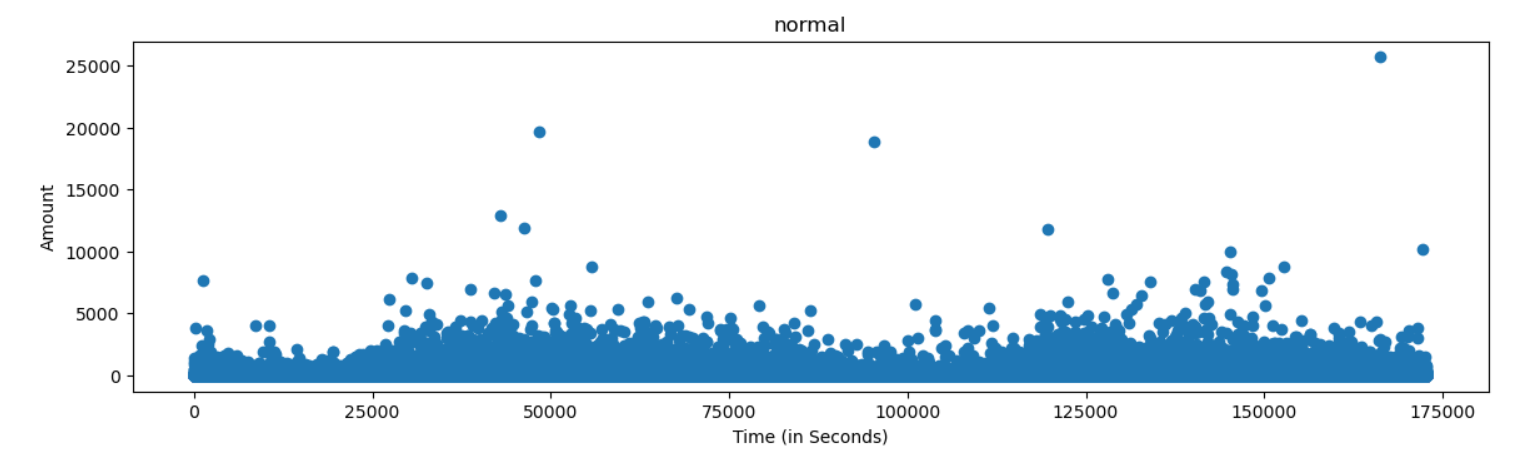
ax2.set\_title('normal')

plt.xlabel('Time (in Seconds)')

plt.ylabel('Amount')

plt.show()





data1= data.sample(frac = 0.1,random\_state=1)

data1.shape

data.shape



Fraud = data1[data1['Class']==1]

Valid = data1[data1['Class']==0]

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

print(outlier\_fraction)

print("Fraud Cases : {}".format(len(Fraud)))

print("Valid Cases : {}".format(len(Valid)))

import seaborn as sns

#get correlations of each features in dataset

corrmat = data1.corr()

top\_corr\_features = corrmat.index

plt.figure(figsize=(20,20))

#plot heat map

g=sns.heatmap(data[top\_corr\_features].corr(),annot=True,cmap="RdYlGn")

#Create independent and Dependent Features

columns = data1.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 variable we are predicting

target = "Class"

# Define a random state

state = np.random.RandomState(42)

X = data1[columns]

Y = data1[target]

X\_outliers = state.uniform(low=0, high=1, size=(X.shape[0], X.shape[1]))

# Print the shapes of X & Y

print(X.shape)

print(Y.shape)

classifiers = {

"Isolation Forest":IsolationForest(n\_estimators=100, max\_samples=len(X),

contamination=outlier\_fraction,random\_state=state, verbose=0),

"Local Outlier Factor":LocalOutlierFactor(n\_neighbors=20, algorithm='auto',

leaf\_size=30, metric='minkowski',

p=2, metric\_params=None, contamination=outlier\_fraction),

"Support Vector Machine":OneClassSVM(kernel='rbf', degree=3, gamma=0.1,nu=0.05,

max\_iter=-1)

}

type(classifiers)

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\_prediction = clf.negative\_outlier\_factor\_

elif clf\_name == "Support Vector Machine":

clf.fit(X)

y\_pred = clf.predict(X)

else:

clf.fit(X)

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

y\_pred = clf.predict(X)

#Reshape the prediction values to 0 for Valid transactions , 1 for Fraud transactions

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 :")

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

print("Classification Report :")

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

