**Credit Card Fraud Detection**

Q1. What is credit card fraud detection?

Credit card fraud detection is the process of identifying whether a credit card transaction is legitimate or fraudulent. This is typically done by analyzing the transaction data for patterns or characteristics that are typical of fraudulent activity. Credit card fraud detection systems use machine learning algorithms to analyze transactions in real-time and flag any transactions that seem suspicious. This helps to protect consumers from fraudulent activity and helps to prevent losses for credit card companies.

Q2. Which organizations use credit card fraud detection?

Some examples of organizations that use credit card fraud detection systems include:

* Visa
* Mastercard
* American Express
* Discover
* JPMorgan Chase
* Bank of America
* Wells Fargo
* Citigroup
* Amazon
* eBay
* PayPal
* Walmart
* Target
* Apple
* Google
* Microsoft
* Intel

This is just a small sample of the many organizations that use credit card fraud detection systems. There are many other credit card companies, banks, and businesses that use these systems to protect themselves and their customers from fraudulent activity.

Top of Form

Q3. Challenges in involved in credit card fraud detection?

1. Enormous Data is processed every day and the model build must be fast enough to respond to the scam in time.
2. Imbalanced Data i.e most of the transactions *(99.8%)* are not fraudulent which makes it really hard for detecting the fraudulent ones
3. Data availability as the data is mostly private.
4. Misclassified Data can be another major issue, as not every fraudulent transaction is caught and reported.
5. Adaptive techniques used against the model by the scammers.

Q4. How to tackle these challenges?

1. The model used must be simple and fast enough to detect the anomaly and classify it as a fraudulent transaction as quickly as possible.
2. Imbalance can be dealt with by properly using some methods which we will talk about in the next paragraph
3. For protecting the privacy of the user the dimensionality of the data can be reduced.
4. A more trustworthy source must be taken which double-check the data, at least for training the model.
5. We can make the model simple and interpretable so that when the scammer adapts to it with just some tweaks we can have a new model up and running to deploy.

Q5. Coding part

1. Importing all the libraries and packages

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib import gridspec

1. Loading the data

data = pd.read\_csv('/kaggle/input/credit-card/creditcard.csv')

1. Understanding the data

data.head()

data.shape

data.tail(10)

1. Describing the data

data.describe()

1. Imbalance in the data

#determine number of fraud cases in the dataset

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

valid = data[data['Class']== 0]

outlierFraction = len(fraud)/float(len(valid))

print(outlierFraction)

print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))

print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))

print("Amount details of the fraudent transaction")

fraud.Amount.describe()

print("Amount details of the valid transaction")

valid.Amount.describe()

1. Plotting the correlation matrix

#plotting the correlation matrix

corrmat = data.corr()

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

sns.heatmap(corrmat, vmax = 0.8, square = True)

plt.show()

1. Separating the X & Y values

#dividing the x and y from the dataset

x = data.drop(['Class'], axis = 1)

y = data['Class']

print(x.shape)

print(y.shape)

xdata = x.values

ydata = y.values

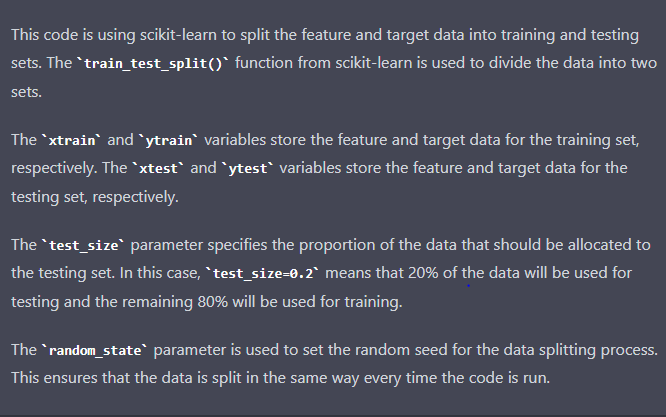
1. Training and Testing Data Bifurcation

#using scikit learn to divide data into trainig and testing

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

#split the data into training and testing

xtrain, xtest, ytrain, ytest = train\_test\_split(xdata, ydata, test\_size = 0.2, random\_state = 42)



1. Building model using sckitlearn

# Building the Random Forest Classifier (RANDOM FOREST)

from sklearn.ensemble import RandomForestClassifier

# random forest model creation

rfc = RandomForestClassifier()

rfc.fit(xtrain, ytrain)

# predictions

yPred = rfc.predict(xtest)

1. Evaluating the model

# Evaluating the classifier

# printing every score of the classifier

# scoring in anything

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

from sklearn.metrics import precision\_score, recall\_score

from sklearn.metrics import f1\_score, matthews\_corrcoef

from sklearn.metrics import confusion\_matrix

n\_outliers = len(fraud)

n\_errors = (yPred != ytest).sum()

print("The model used is Random Forest classifier")

acc = accuracy\_score(ytest, yPred)

print("The accuracy is {}".format(acc))

prec = precision\_score(ytest, yPred)

print("The precision is {}".format(prec))

rec = recall\_score(ytest, yPred)

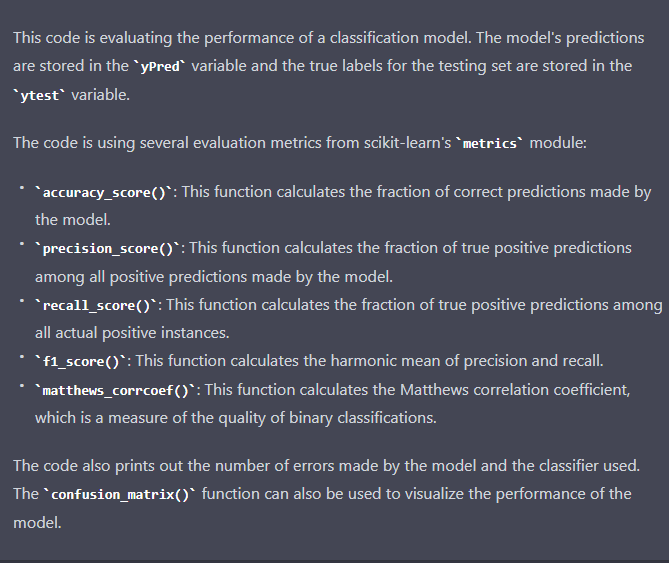
print("The recall is {}".format(rec))

f1 = f1\_score(ytest, yPred)

print("The F1-Score is {}".format(f1))

MCC = matthews\_corrcoef(ytest, yPred)

print("The Matthews correlation coefficient is{}".format(MCC))



1. Visualizing the confusion matrix

# printing the confusion matrix

LABELS = ['Normal', 'Fraud']

conf\_matrix = confusion\_matrix(ytest, yPred)

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

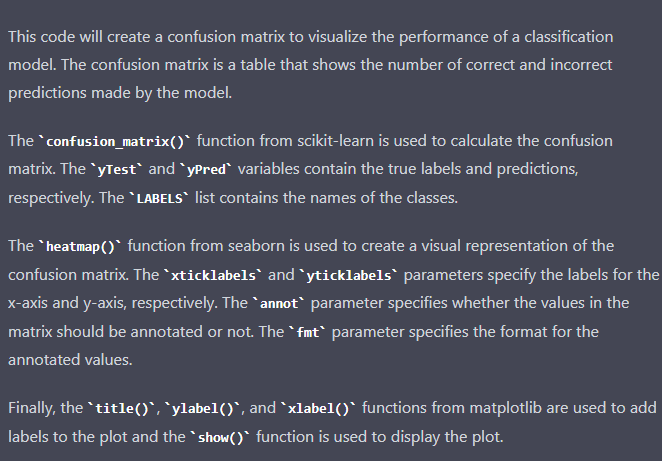
sns.heatmap(conf\_matrix, xticklabels = LABELS,yticklabels = LABELS, annot = True, fmt ="d");

plt.title("Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()



Q6. Answers by other data scientists

