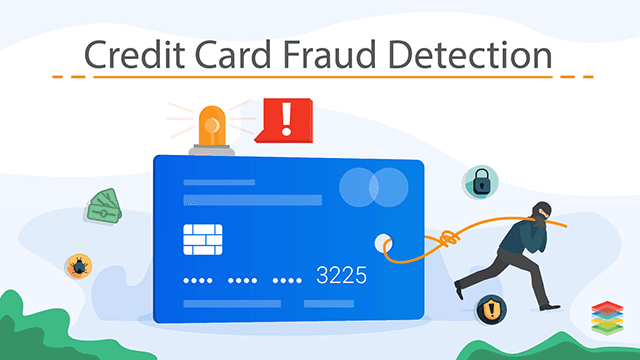
**TITLE: CRDIT CARD FRAUD DETECTION**

**PHASE 5: Project documentation and submission**

**Team member name: Abinaya K**



**Introduction:**

* Credit card fraud is a significant challenge in the financial industry, costing billions of dollars each year and eroding trust among consumers. Design thinking offers a human-centered approach to addressing this problem by focusing on understanding user needs, brainstorming innovative solutions, and rapidly testing and iterating on those solutions. This document outlines a design thinking approach to developing a credit card fraud detection system.

**Problem definition:**

* The problem is to develop a machine learning based system for real time credit card fraud detection. The goal is to create a solution that can accurately identify the fraudulent transactions while minimizing the false positives. This project involves data pre-processing, feature engineering, model selection, training and evaluation to create a robust fraud detection system.

**Design thinking process:**

* Credit card fraud detection is a critical component of the financial industry and e-commerce sector, playing a vital role in safeguarding consumers, businesses, and financial institutions from various forms of fraudulent activities. Here's an overview of its significance:
  + - * Protects consumers
      * Safeguards business
      * Supports financial institutions
      * Reduces operational costs
      * Compacts evolving fraud techniques
      * Enhances user experience
* Empathize: understand user needs

here are the target audiences for credit card fraud detection:

* + - * + Financial Institutions
        + Fraud Analysts
        + Law Enforcement
        + IT and Security Teams
        + Regulatory Authorities
        + Merchants
        + Technology Providers
        + Consumers' Advocacy Groups
        + Insurance Companies
        + Data Privacy Organizations
        + Third-Party Auditors and Compliance Experts
        + Credit card holders
* Global Fraud Losses: In 2021, global card fraud losses were estimated to be around $27 billion.
* Fraud Detection Accuracy: Modern fraud detection systems utilizing machine learning and AI achieve accuracy rates of around 95% or higher in identifying fraudulent transactions.
* Chargeback Rates: The chargeback rate for credit card transactions was approximately 0.07% in 2020.
* EMV Chip Adoption: As of 2021, the adoption of EMV chip technology in credit cards increased to over 86% in the United States, reducing card-present fraud significantly.
* Credit cardholders face several challenges related to the security and use of their credit cards. Here are three common challenges they may encounter:
* Unauthorized Transactions: Credit cardholders may experience unauthorized or fraudulent transactions on their card, resulting in financial losses and potential inconvenience. Detecting and reporting such transactions promptly can be a challenge.
* Identity Theft and Data Breaches: Credit cardholders are at risk of identity theft and data breaches, where their personal and financial information may be stolen and used for fraudulent activities. Safeguarding personal data and being vigilant against such threats can be challenging.
* Disputing Charges: Resolving disputes with merchants and credit card companies, especially in cases of disputed or unauthorized charges, can be a cumbersome process. Credit cardholders may need to navigate through complex procedures to have these issues resolved in their favor.

Define: problem statement and goals

* The problem is to develop a machine learning based system for real time credit card fraud detection. The goal is to create a solution that can accurately identify the fraudulent transactions while minimizing the false positives. This project involves data pre-processing, feature engineering, model selection, training and evaluation to create a robust fraud detection system.
* Here are the Goals of credit card fraud detection:
* Early Detection
* Minimize False Positives
* Reduce Financial Losses
* Protect Customer Trust
* Adapt to Emerging Threats
* Compliance with Regulations
* Improve Operational Efficiency
* Enhance User Experience
* Collaboration and Information Sharing
* Prevent Cross-Border Fraud

Ideate: generate solutions

In our project we have used following algorithms,

1. Anomaly detection
2. One class svm

A One-Class Support Vector Machine (One-Class SVM) is a machine learning algorithm designed for anomaly detection in imbalanced datasets, separating normal data from anomalies. It maximizes the margin between normal data points and the decision boundary, using a kernel function for non-linear patterns. It's employed in applications like fraud detection and network security.

Data collection and processing:

**Data Collection:** The project will collect transaction data from various sources, including credit card companies, financial institutions, and e-commerce platforms. This data will serve as the foundation for building fraud detection models.

**Data Preprocessing**: Data preprocessing is essential for cleaning, transforming, and normalizing the data. This step ensures that the data is suitable for analysis and model training.

**Data visualization**:

Generate summary statistics like mean, median, standard deviation, etc., for numerical features.

Create histograms, box plots, or density plots to visualize the distribution of transaction amounts and other relevant features.

Plot the distribution of fraudulent vs. non-fraudulent transactions to understand class imbalance.

**Feature engineering:**

Feature engineering involves selecting relevant features or variables that contribute to fraud detection.

It may also involve creating new features that can enhance the accuracy of the models.

Techniques used:

• logistic regression

• random forest

• anomaly detection

**Model development and evaluation**:

• Machine learning models, including logistic regression, decision trees, random forests, and will be developed and trained using historical transaction data to predict fraudulent activities.

• The performance of the models will be regularly assessed using metrics such as precision, recall, and F1-score. Continuous model refinement and updates will be carried out to adapt to evolving fraud patterns.

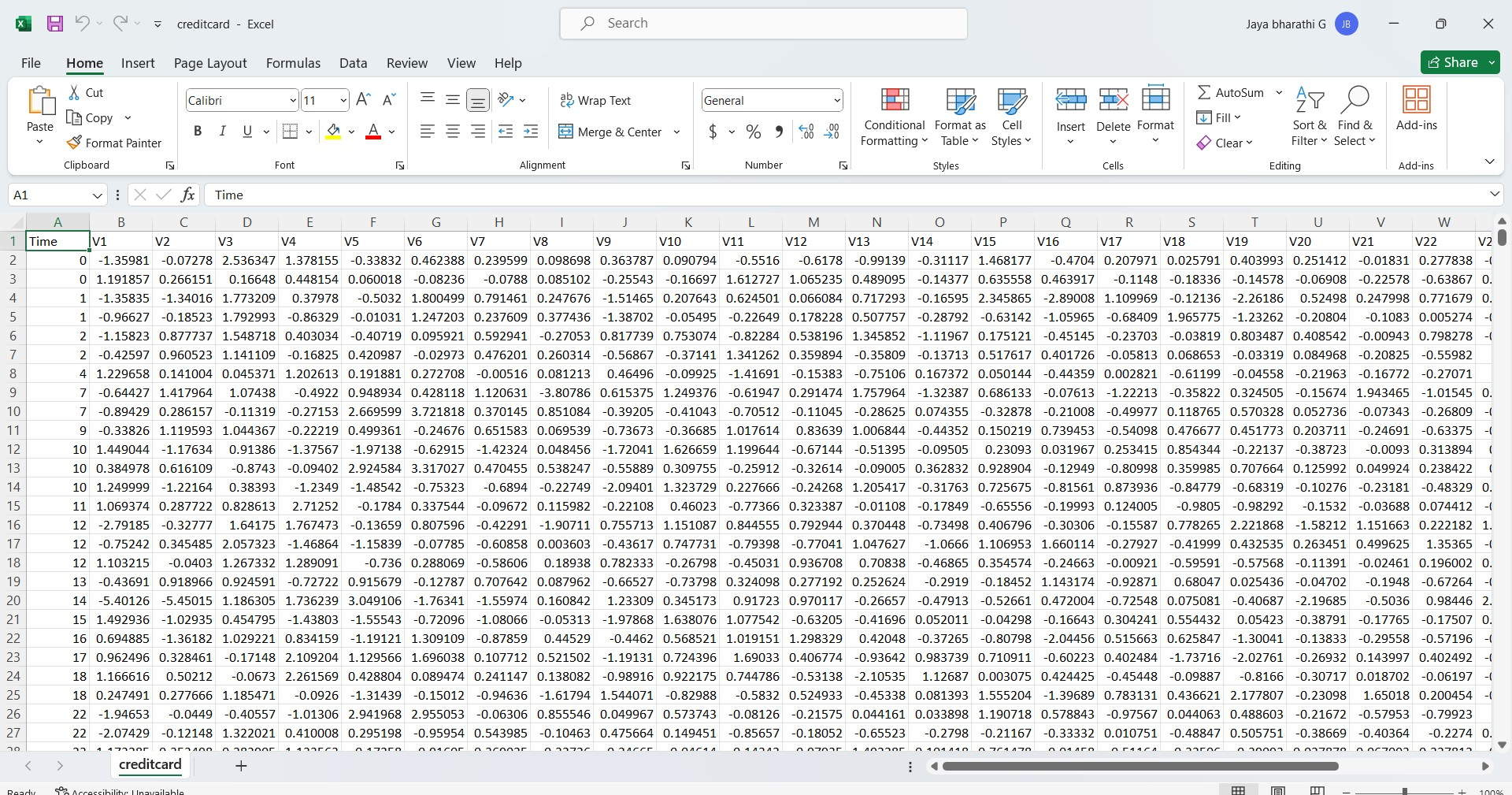
***Dataset:***

**Loading the data:**

For loading the data, we use the dataset using the given dataset link

Dataset link: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>

The dataset looks like below,



Observations The data set is highly skewed, consisting of 492 frauds in a total of 284,807 observations. This resulted in only 0.172 % fraud cases. This skewed set is justified by the low number of fraudulent transactions.

The dataset consists of numerical values from the 28 ‘Principal Component Analysis (PCA)’ transformed features, namely V1 to V28. Furthermore, there is no metadata about the original features provided, so pre-analysis or feature study could not be done.

The ‘Time’ and ‘Amount’ features are not transformed data.

There is no missing value in the dataset.

**Importing libraries:**

* numpy
* pandas
* matplotlib
* scikit

here’s the code for importing the libraries,

import numpy as np

import pandas as pd

import sklearn

import scipy

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.neighbours import LocalOutlierFactor

from sklearn.svm import OneClassSVM

from pylab import rcParams

rcParams[‘figure,figsize’]=14,8

RANDOM\_SEED=42

LABELS= [“Normal”, ”Fraud”]

#import plotly.plotly as py

import plotly.graph\_objs as go

import plotly

import plotly.figure\_factory as ff

from plotly.offline import init\_notebook, iplot

these are the required libraries

**Loading dataset:**

The dataset can be loaded and manipulated by using the pandas library

Here’s the code for loading the data set

data= pd.read\_csv(‘..input/creditcard.csv’)

here we saved the dataset in the .csv format else we can save the dataset in the xlv format

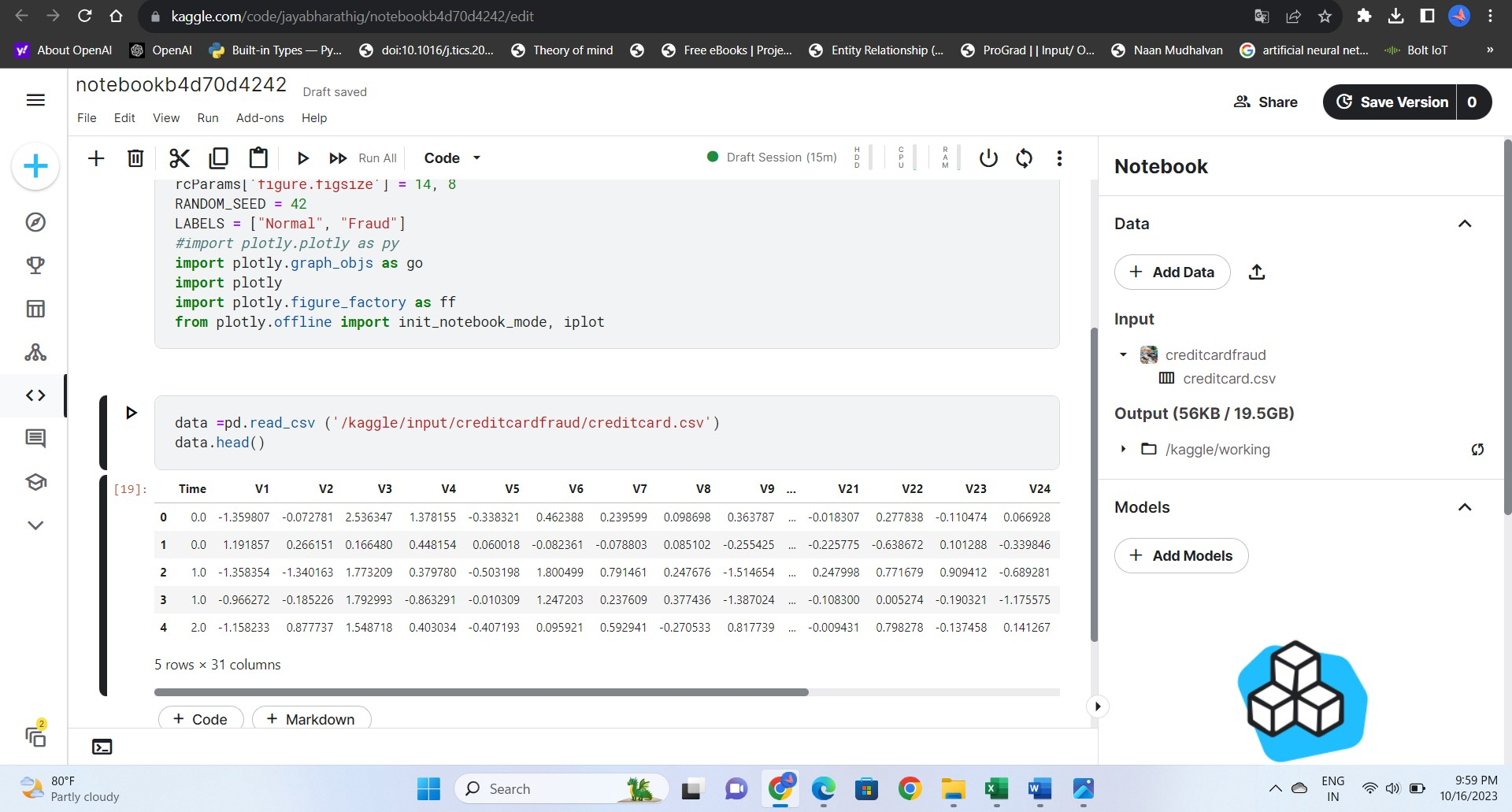
once the dataset is loaded, we can see the first 5 rows of the dataset by using the following line of code,

the head () function allows you to view the first few rows in the dataframe.

We can specify the number of rows as per our requirement.

data.head()

output:



Now for shaping the data we will use the following code,

To retrieve the dimensions of the dataframe we can access the shape attribute

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

data1.shape

output:

(28481, 31)

**Checking the missing values:**

For checking the missing values we use isnull() function,

data.isnull().sum()

output:

Time 0

V1 0

V2 0

V3 0

V4 0

V5 0

V6 0

V7 0

V8 0

V9 0

V10 0

V11 0

V12 0

V13 0

V14 0

V15 0

V16 0

V17 0

V18 0

V19 0

V20 0

V21 0

V22 0

V23 0

V24 0

V25 0

V26 0

V27 0

V28 0

Amount 0

Class 0

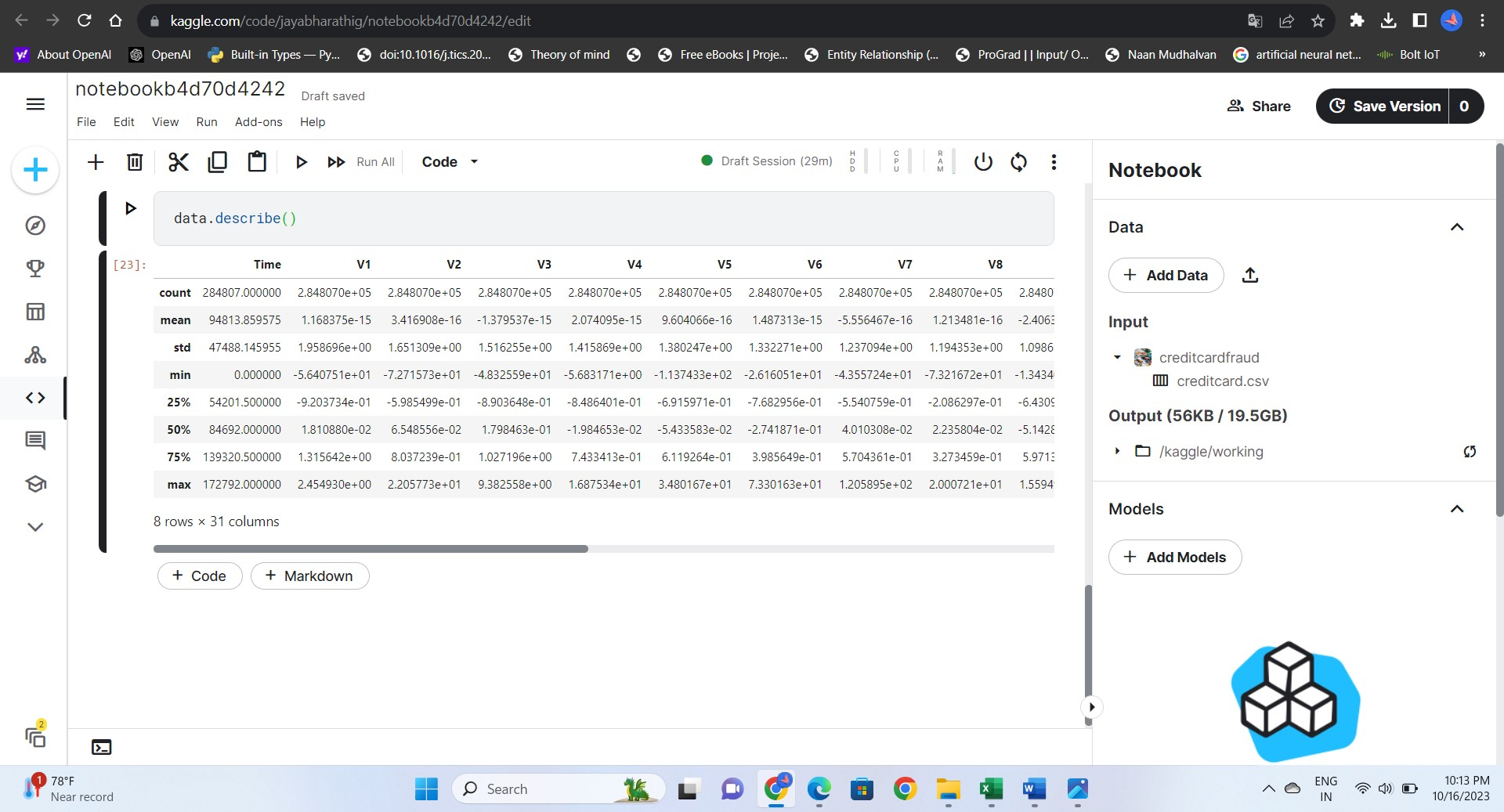
dtype: int64

from the above table ,we can see that there is no null values(missing values in the dataset)

**describe():**

to generate the summary statistics for the numerical columns in the data frame, we use describe() method.

data.describe()



To determine the number of fraud and valid transactions in the entire dataset

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

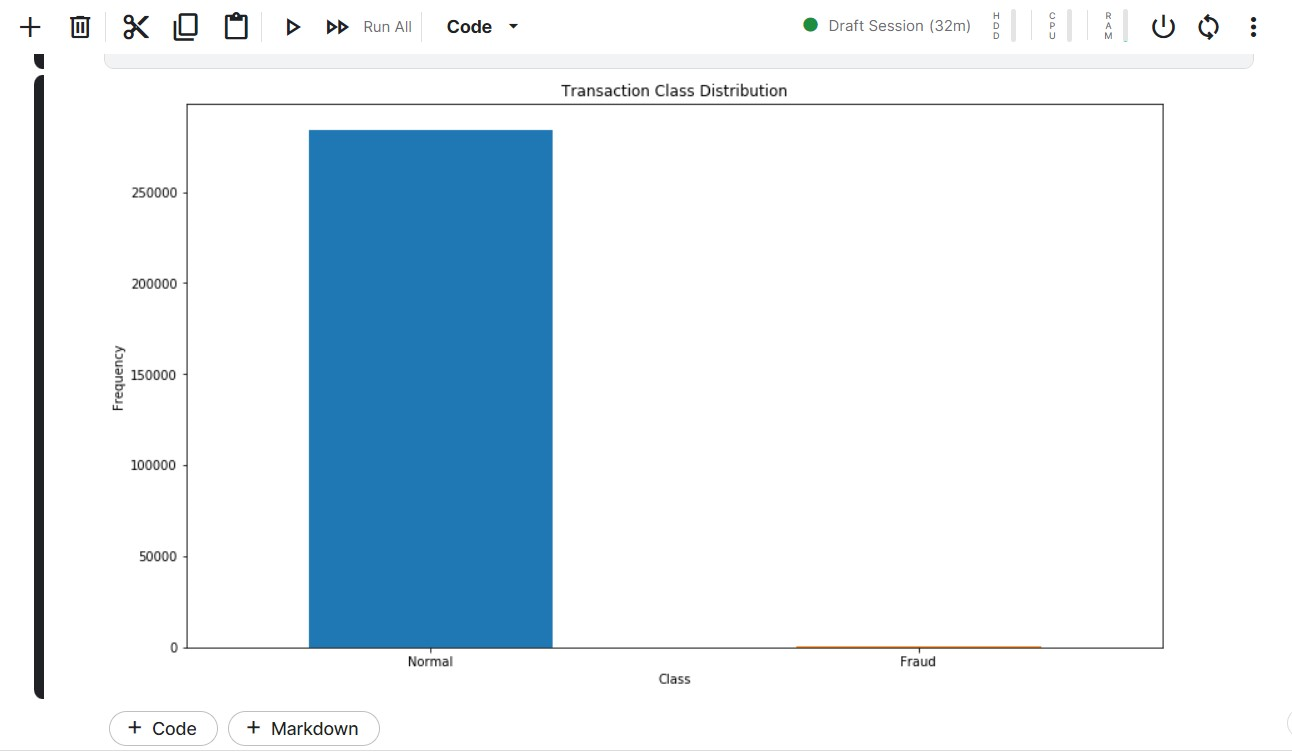
count\_classes.plot(kind = 'bar', rot=0)

plt.title("Transaction Class Distribution")

plt.xticks(range(2), LABELS)

plt.xlabel("Class")

plt.ylabel("Frequency")

output: 

**Assigning the transaction class "0 = NORMAL & 1 = FRAUD**

Normal = data[data['Class']==0]

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

Normal.shape

Fraud.shape

Output:

(284315, 31)

(492, 31)

Now we will try to analyse how different are the amount of money used in different transaction classes?

Normal.Amount.describe()

Output:

count 284315.000000

mean 88.291022

std 250.105092

min 0.000000

25% 5.650000

50% 22.000000

75% 77.050000

max 25691.160000

Name: Amount, dtype: float64

Fraud.Amount.describe()

Output:

count 492.000000

mean 122.211321

std 256.683288

min 0.000000

25% 1.000000

50% 9.250000

75% 105.890000

max 2125.870000

Name: Amount, dtype: float64

The loading and preprocessing has been done in this phase let us use some visualization techniques to draw the conclusion

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 ($)')

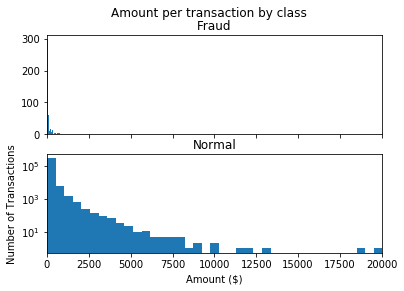
plt.ylabel('Number of Transactions')

plt.xlim((0, 20000))

plt.yscale('log')

plt.show()

output:



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)

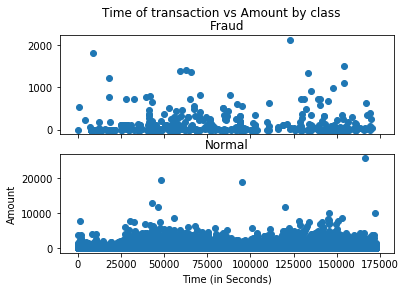
ax2.set\_title('Normal')

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

plt.ylabel('Amount')

plt.show();

output:



init\_notebook\_mode(connected=True)

plotly.offline.init\_notebook\_mode(connected=True)

trace = go.Scatter(

x = Fraud.Time,

y = Fraud.Amount,

mode = 'markers'

)

data = [trace]

plotly.offline.iplot({

"data": data

})

**Determine the number of fraud and valid transactions in the dataset**

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)))

output:

0.0017234102419808666

Fraud Cases : 49

Valid Cases : 28432

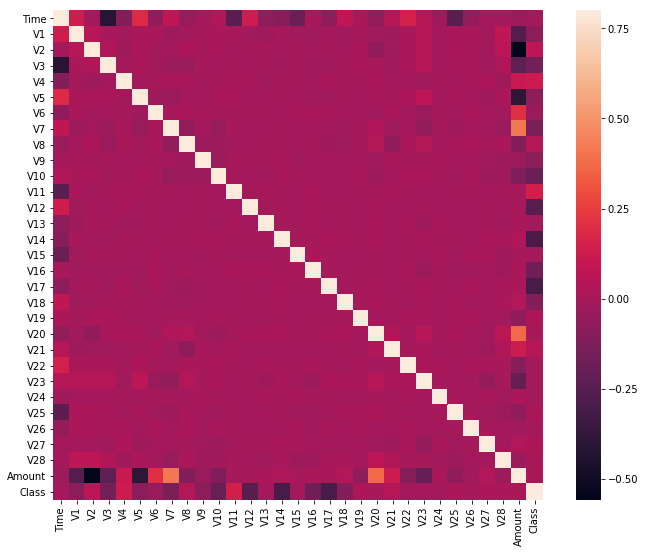
**Correlation matrix:**

correlation\_matrix = data1.corr()

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

sns.heatmap(correlation\_matrix,vmax=0.8,square = True)

plt.show()

output:  


**Evaluation metrics:**

**Classification confusion metrics:**

|  |  |  |
| --- | --- | --- |
|  | Actual positive  Y=1 | Actual negative  Y=0 |
| Predicted positive | True positive (TP) | False positive (FP) |
| Predicted negative | False negative (FN) | True negative (TN) |

From this table, several statistics are extracted. In particular:

• Accuracy = T P+T N T P+T N+F P+FN

• Recall = T P T P+FN

• Precision = T P T P+F P

• F1Score = 2 Precision·Recall/ Precision+Recall

**Data Preprocessing:**

• Perform data preprocessing steps, such as normalization and handling missing values.

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**Model training**:

*Anomaly Detection*

Anomaly detection is a technique used to identify unusual patterns that do not conform to expected behavior, called outliers. It has many applications in business, from intrusion detection (identifying strange patterns in network traffic that could signal a hack) to system health monitoring (spotting a malignant tumor in an MRI scan), and from fraud detection in credit card transactions to fault detection in operating environments.

What Are Anomalies? Anomalies can be broadly categorized as:

Point anomalies: A single instance of data is anomalous if it's too far off from the rest. Business use case: Detecting credit card fraud based on "amount spent."

Contextual anomalies: The abnormality is context specific. This type of anomaly is common in time-series data. Business use case: Spending $100 on food every day during the holiday season is normal, but may be odd otherwise.

Collective anomalies: A set of data instances collectively helps in detecting anomalies. Business use case: Someone is trying to copy data form a remote machine to a local host unexpectedly, an anomaly that would be flagged as a potential cyberattack.

Anomaly detection is similar to — but not entirely the same as — noise removal and novelty detection.

Novelty detection is concerned with identifying an unobserved pattern in new observations not included in training data like a sudden interest in a new channel on YouTube during Christmas, for instance.

Noise removal (NR) is the process of removing noise from an otherwise meaningful signal.

Anomaly Detection Techniques

Simple Statistical Methods

The simplest approach to identifying irregularities in data is to flag the data points that deviate from common statistical properties of a distribution, including mean, median, mode, and quantiles. Let's say the definition of an anomalous data point is one that deviates by a certain standard deviation from the mean. Traversing mean over time-series data isn't exactly trivial, as it's not static. You would need a rolling window to compute the average across the data points. Technically, this is called a rolling average or a moving average, and it's intended to smooth short-term fluctuations and highlight long-term ones. Mathematically, an n-period simple moving average can also be defined as a "low pass filter."

Challenges with Simple Statistical Methods The low pass filter allows you to identify anomalies in simple use cases, but there are certain situations where this technique won't work. Here are a few:

The data contains noise which might be similar to abnormal behavior, because the boundary between normal and abnormal behavior is often not precise.

The definition of abnormal or normal may frequently change, as malicious adversaries constantly adapt themselves. Therefore, the threshold based on moving average may not always apply.

The pattern is based on seasonality. This involves more sophisticated methods, such as decomposing the data into multiple trends in order to identify the change in seasonality.

**Machine Learning-Based Approaches**

Below is a brief overview of popular machine learning-based techniques for anomaly detection.

a)Density-Based Anomaly Detection Density-based anomaly detection is based on the k-nearest neighbors algorithm.

Assumption: Normal data points occur around a dense neighborhood and abnormalities are far away.

The nearest set of data points are evaluated using a score, which could be Eucledian distance or a similar measure dependent on the type of the data (categorical or numerical). They could be broadly classified into two algorithms:

K-nearest neighbor: k-NN is a simple, non-parametric lazy learning technique used to classify data based on similarities in distance metrics such as Eucledian, Manhattan, Minkowski, or Hamming distance.

Relative density of data: This is better known as local outlier factor (LOF). This concept is based on a distance metric called reachability distance.

b)Clustering-Based Anomaly Detection Clustering is one of the most popular concepts in the domain of unsupervised learning.

Assumption: Data points that are similar tend to belong to similar groups or clusters, as determined by their distance from local centroids.

K-means is a widely used clustering algorithm. It creates 'k' similar clusters of data points. Data instances that fall outside of these groups could potentially be marked as anomalies.

c)Support Vector Machine-Based Anomaly Detection

A support vector machine is another effective technique for detecting anomalies. A SVM is typically associated with supervised learning, but there are extensions (OneClassCVM, for instance) that can be used to identify anomalies as an unsupervised problems (in which training data are not labeled). The algorithm learns a soft boundary in order to cluster the normal data instances using the training set, and then, using the testing instance, it tunes itself to identify the abnormalities that fall outside the learned region. Depending on the use case, the output of an anomaly detector could be numeric scalar values for filtering on domain-specific thresholds or textual labels (such as binary/multi labels).

Isolation Forest Anomaly Detection Algorithm Density-Based Anomaly Detection (Local Outlier Factor)Algorithm Support Vector Machine Anomaly Detection Algorithm Credit Card Fraud Detection Problem Statement: The Credit Card Fraud Detection Problem includes modeling past credit card transactions with the knowledge of the ones that turned out to be fraud. This model is then used to identify whether a new transaction is fraudulent or not. Our aim here is to detect 100 % of the fraudulent transactions while minimizing the incorrect fraud classifications.

**Model evaluation:**

We can evaluate the model by using isolation forest , one calss vm algorithms.

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)

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))

output:

/opt/conda/lib/python3.10/site-packages/sklearn/base.py:439: UserWarning:

X does not have valid feature names, but IsolationForest was fitted with feature names

Isolation Forest: 73

Accuracy Score :

0.9974368877497279

Classification Report :

precision recall f1-score support

0 1.00 1.00 1.00 28432

1 0.26 0.27 0.26 49

accuracy 1.00 28481

macro avg 0.63 0.63 0.63 28481

weighted avg 1.00 1.00 1.00 28481

Local Outlier Factor: 97

Accuracy Score :

0.9965942207085425

Classification Report :

precision recall f1-score support

0 1.00 1.00 1.00 28432

1 0.02 0.02 0.02 49

accuracy 1.00 28481

macro avg 0.51 0.51 0.51 28481

weighted avg 1.00 1.00 1.00 28481

Support Vector Machine: 8516

Accuracy Score :

0.7009936448860644

Classification Report :

precision recall f1-score support

0 1.00 0.70 0.82 28432

1 0.00 0.37 0.00 49

accuracy 0.70 28481

macro avg 0.50 0.53 0.41 28481

weighted avg 1.00 0.70 0.82 28481

}

Conclusion:

In conclusion, credit card fraud detection is a critical component of the financial industry's efforts to protect cardholders and institutions from fraudulent activities. By leveraging advanced technologies, collaboration, and a user-centric approach, it aims to detect anomalies, reduce financial losses, and maintain the trust of customers. The ongoing evolution of fraud detection methods is vital in addressing the ever-changing landscape of credit card fraud, ensuring the security and integrity of financial transactions.