****

**Credit Card Fraud Detection**

**Submitted by:**

Muhammad Salman Asghar

**Submitted to:**

Usman Anwar

Dated: December, 2023

**Table of Content:**

Credit Card Fraud Detection……………………………...……….page 4

Credit Card Fraud Detection Model……………………………….page 5

Graphical representation of Amount………………………………page 7

Building the model………………………………………………...page 9

Credit Card Fraud Detection Prediction model……………….….page 11

Summary……………………………………………………...… .page12

**List Of Figures:**

Figure 1……………………………………………………..……page 7

Figure 2……………………………………………………..……page 8

**Credit Card Fraud Detection:**

A credit card fraud detection model in machine learning involves the development of algorithms and models to identify potentially fraudulent credit card transactions. Various machine learning techniques can be employed to distinguish between legitimate and fraudulent transactions based on patterns and features present in historical transaction data. Here's a basic overview of the steps involved in building a credit card fraud detection model:

* **Data Collection:**

Collect a dataset that includes historical credit card transactions labeled as either legitimate or fraudulent. The dataset should cover a diverse range of transaction scenarios and include features such as transaction amount, merchant information, time of day, and more.

* **Data Preprocessing:**

Clean and preprocess the dataset. This may involve handling missing values, normalizing numerical features, encoding categorical variables, and balancing the dataset to address class imbalance issues (since fraudulent transactions are often much less frequent than legitimate ones).

* **Feature Engineering**:

Identify and extract relevant features from the dataset. Feature engineering is crucial for training effective machine learning models. Common features for credit card fraud detection include transaction amount, frequency, location, and time-related features..

* **Model Training:**

Train the selected model using the preprocessed dataset. This involves feeding the algorithm the features of known transactions and allowing it to learn the patterns associated with legitimate and fraudulent transactions.

* **Model Evaluation**:

Evaluate the model's performance using a separate test dataset. Common evaluation metrics include precision, recall, F1 score, and area under the Receiver Operating Characteristic (ROC) curve.

**Credit Card Fraud Detection Model:**

I will create a model for credit card fraud detection using machine learning:

**Importing the Librabries:**

* Lets start with importing necessary libraries:

import pandas as pd

import numpy as np

import pickle

import matplotlib.pyplot as plt

from scipy import stats

import seaborn as sns

from pylab import rcParams

%matplotlib inline

sns.set(style='whitegrid', palette='muted', font\_scale=1.5)

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

RANDOM\_SEED = 42

LABELS = ["Normal", "Fraud"]

### **Importing the Data set:**

df = pd.read\_csv("C:/pdata/creditcard.csv")

* The data set I am going to use contains data about credit card transactions that occurred during a period of two days, with 492 frauds out of 284,807 transactions. All variables in the data set are numerical. The data has been transformed using PCA transformation(s) due to privacy reasons.The two features that haven’t been changed are Time and Amount. Time contains the seconds elapsed between each transaction and the first transaction in the data set.

**Analyzing the data:**

df.head()

**Checking the shape of data:**

df.shape

**Checking for null values:**

df.info()

* There are no null values in the data.

**Checking number of records of each kind of transaction class (Fraud and Non-Fraud):**

count\_classes = pd.value\_counts(df['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")

* The data set is highly imbalanced.

**Looking at each of the fraud(1) and non-fraud(0) transactions:**

frauds = df[df.Class == 1]

normal = df[df.Class == 0]

frauds.shape

normal.shape

Since only 3 of the features (time, amount and Class) are non-anomyzed, let’s explore them.

* Fraud transactions

frauds.Amount.describe()

* Non-fraud transactions

normal.Amount.describe()

### **Graphical representation of Amount:**

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

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

bins = 50

ax1.hist(frauds.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()

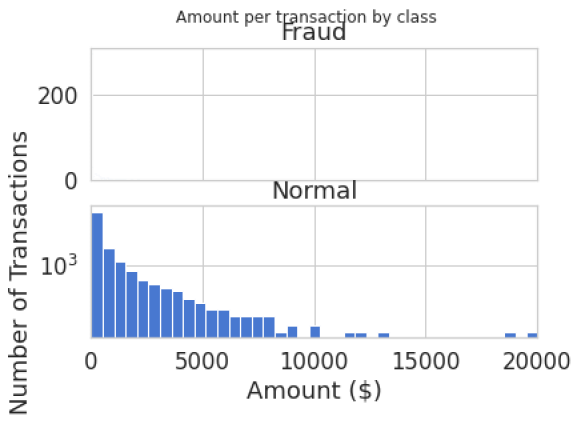


Figure 1

### **Plotting time of transaction to check for correlations:**

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

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

ax1.scatter(frauds.Time, frauds.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()

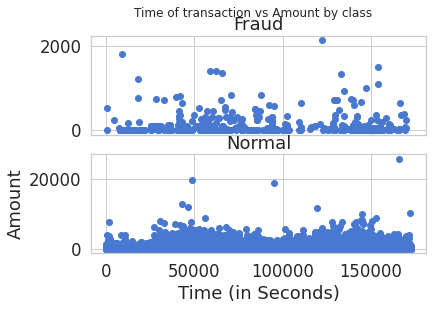


Figure 2

* The time does not seem to be a crucial feature in distinguishing normal vs fraud cases. Hence, I will drop it.

data = df.drop(['Time'], axis=1)

* The numerical amount in fraud and normal cases differ highly, hence we scale them.

### **Scaling the Amount using Standard Scaler**:

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

### from sklearn.preprocessing import MinMaxScaler

### from sklearn.linear\_model import LogisticRegression

### from sklearn.manifold import TSNE

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

### from sklearn.preprocessing import StandardScaler

### data['Amount'] = StandardScaler().fit\_transform(data['Amount'].values.reshape(-1, 1))

### **Building the model:**

* We will be using **autoencoders** for the fraud detection model. Using autoencoders, we train the database only to learn the representation of the non-fraudulent transactions. The reason behind applying this method is to let the model learn the best representation of non-fraudulent cases so that it automatically distinguishes the other case from it.

non\_fraud = data[data['Class'] == 0]

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

df = pd.concat([non\_fraud, fraud]).sample(frac=1).reset\_index(drop=True)

X = df.drop(['Class'], axis = 1).values

Y = df["Class"].values

**Spiting the data into 80% training and 20% testing:**

X\_train, X\_test = train\_test\_split(data, test\_size=0.2, random\_state=RANDOM\_SEED)

X\_train\_fraud = X\_train[X\_train.Class == 1]

X\_train = X\_train[X\_train.Class == 0]

X\_train = X\_train.drop(['Class'], axis=1)

y\_test = X\_test['Class']

X\_test = X\_test.drop(['Class'], axis=1)

X\_train = X\_train.values

X\_test = X\_test.values

X\_train.shape

### **Autoencoder model:**

input\_layer = Input(shape=(X.shape[1],))

* encoding part

encoded = Dense(100, activation='tanh', activity\_regularizer=regularizers.l1(10e-5))(input\_layer)

encoded = Dense(50, activation='relu')(encoded)

* decoding part

decoded = Dense(50, activation='tanh')(encoded)

decoded = Dense(100, activation='tanh')(decoded)

* output layer

output\_layer = Dense(X.shape[1], activation='relu')(decoded)

### **Training the credit card fraud detection model:**

autoencoder = Model(input\_layer, output\_layer)

autoencoder.compile(optimizer="adadelta", loss="mse")

### **Scaling the values:**

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

y = data["Class"].values

x\_scale = MinMaxScaler().fit\_transform(x.values)

x\_norm, x\_fraud = x\_scale[y == 0], x\_scale[y == 1]

autoencoder.fit(x\_norm[0:2000], x\_norm[0:2000],

batch\_size = 256, epochs = 10,

shuffle = True, validation\_split = 0.20);

### **Obtain the Hidden Representation:**

hidden\_representation = Sequential()

hidden\_representation.add(autoencoder.layers[0])

hidden\_representation.add(autoencoder.layers[1])

hidden\_representation.add(autoencoder.layers[2])

### **Model Prediction:**

norm\_hid\_rep = hidden\_representation.predict(x\_norm[:3000])

fraud\_hid\_rep = hidden\_representation.predict(x\_fraud)

### **Getting the representation data:**

rep\_x = np.append(norm\_hid\_rep, fraud\_hid\_rep, axis = 0)

y\_n = np.zeros(norm\_hid\_rep.shape[0])

y\_f = np.ones(fraud\_hid\_rep.shape[0])

rep\_y = np.append(y\_n, y\_f)

### **Train, test, split:**

train\_y, val\_y = train\_test\_split(rep\_x, rep\_y, test\_size=0.25)

## **Credit Card Fraud Detection Prediction model:**

clf = LogisticRegression(solver="lbfgs").fit(train\_x, train\_y)

pred\_y = clf.predict(val\_x)

print ("Classification Report: ")

print (classification\_report(val\_y, pred\_y))

print ("Accuracy Score: ", accuracy\_score(val\_y, pred\_y))

**Summary:**

A credit card fraud detection model is a sophisticated machine learning system developed to discern and thwart unauthorized or fraudulent transactions. Utilizing historical credit card transaction data, the model undergoes a meticulous process, starting with the collection of a dataset containing labeled instances of both legitimate and fraudulent transactions. This data is then subjected to preprocessing, which involves cleaning, normalizing, and addressing class imbalances. Relevant features such as transaction amount, frequency, and temporal attributes are extracted in the feature engineering phase. The model selection entails choosing an appropriate algorithm for binary classification, and subsequent training involves exposing the model to the historical dataset to recognize patterns indicative of fraud. Evaluation metrics like precision, recall, and ROC curve analysis gauge the model's performance on a separate test dataset. Following hyperparameter tuning, the trained model is deployed in a live environment for real-time monitoring and analysis of credit card transactions. Continuous monitoring, adaptability to emerging fraud patterns, and considerations for interpretability and explainability are integral for the model's effectiveness and compliance.