FRAUD DETECTION IN FINANCIAL TRANSACTIONS USING MACHINE LEARNING

A PROJECT REPORT

Submitted by

VISHNU ANISH YADAV A [211421104034]

AKASH PATEL S [211421104014]

JEEVAN G [211421104071]

in partial fulfillment for the award of the degree

of

BACHELORS OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project report **"FRAUD DETECTION IN FINANCIAL TRANSACTIONS USING MACHINE LEARNING"** is the bonafide work Of **"VISHNU ANISH YADAV A, AKASH PATEL S, JEEVAN G"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. L. JABASHEELA, M.E., Ph.D.,	**Dr. E. RAJALAKSHMI, B.E., M.B.A., M.E., Ph.D.,**
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR**
DEPARTMENT OF CSE, PANIMALAR ENGINEERING COLLEGE, NARASATHPETTAI, CHENNAI – 600 123.	DEPARTMENT OF CSE, PANIMALAR ENGINEERING COLLEGE, NARASATHPETTAI, CHENNAI – 600 123.

Certified that the above candidate(s) was examined in the End Semester Project Viva-Voce

Examination held on ....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **VISHNU ANISH YADAV A (211421104034), AKASH PATEL S (211421104014), JEEVAN G (211421104071)** hereby declare that this project report titled **“FRAUD DETECTION IN FINANCIAL TRANSACTIONS USING MACHINE LEARNING”,** under the guidance of **DR E. RAJALAKSHMI B.E, M.B.A., M.E., Ph.D.,** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D.,** and **Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to Project Guide **DR E. RAJALAKSHMI B.E, M.B.A., M.E., Ph.D.** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

VISHNU ANISH YADAV A

AKASH PATEL S

JEEVAN G

ABSTRACT

Payments related fraud is a key aspect of cyber-crime agencies and recent research has shown that machine learning techniques can be applied successfully to detect fraudulent transactions in large amounts of payments data. Such techniques have the ability to detect fraudulent transactions that human auditors may not be able to catch and also do this on a real time basis.

In this project, we apply multiple supervised machine learning techniques to the problem of fraud detection using a publicly available simulated payment transactions data. We aim to demonstrate how supervised ML techniques can be used to classify data with high class imbalance with high accuracy.

We demonstrate that exploratory analysis can be used to separate fraudulent and non- fraudulent transactions. We also demonstrate that for a well separated datasets, tree- based algorithms like Random Forest work much better than Logistic Regression.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES:

CHAPTER 1
INTRODUCTION

1. INTRODUTION

Digital payments of various forms are rapidly increasing across the world. Payments companies are experiencing rapid growth in their transactions volume. For example, PayPal processed ~$578 billion in total payments in 2018. Along with this transformation, there is also a rapid increase in financial fraud that happens in these payment systems.

Preventing online financial fraud is a vital part of the work done by cybersecurity and cyber-crime teams. Most banks and financial institutions have dedicated teams of dozens of analysts building automated systems to analyze transactions taking place through their products and flag potentially fraudulent ones. Therefore, it is essential to explore the approach to solving the problem of detecting fraudulent entries/transactions in large amounts of data in order to be better prepared to solve cyber-crime cases.

1.1 PROBLEM STATEMENT

The objective of this problem is to develop a machine learning-based system that can accurately identify fraudulent transactions in a large set of financial transaction data. The system should be capable of distinguishing between legitimate and fraudulent transactions based on historical patterns, customer behaviors, and transaction features. The challenge lies in detecting new, previously unseen fraud patterns while maintaining a low rate of false positives

KEY COMPONENTS

TRANSACTION DATA:

The data includes various features such as transaction amount, transaction type, location, time, merchant details, user account information, device information, and transaction history.

FRAUDULANT vs NON- FRAUDULANT:

Transactions are typically labeled as either fraudulent (fraud) or legitimate (non-fraud). This labeled data is used to train machine learning models.

IMBALANCED DATASETS:

Fraudulent transactions are often much rarer than legitimate ones, leading to an imbalanced datasets. This imbalance can impact the performance of machine learning algorithms, making it harder to detect fraud.

FEATURES:

- **TRANSACTION AMOUNT :** Large or unusually small transactions might indicate fraud.
- **TRANSACTION TIME :** Transactions occurring at odd hours or at a higher frequency may be suspicious.
- **LOCATION AND DEVICE :** Transactions from new or unusual locations or devices may indicate fraud.
- **ACCOUNT HISTORY :** A sudden change in transaction patterns or behaviors may be a sign of fraud.
- **MERCHANT DETAILS :** Fraudsters may use certain types of merchants more frequently.

1.2 RESEARCH METHODOLOGY

The typical machine learning approach was followed in this project. The identified datasets has label-led class variable, which was used as the prediction variable in machine learning models.

- Through exploratory analysis, we analyzed the data set in detail and identified possible predictors of fraud.
- Through various visualization techniques, we observed the separation between fraud and non-fraud transactions.
- To solve the fraud detection problem, we experimented with two supervised machine learning techniques – Logistic Regression and Random Forest.

- Additionally, we also tried under-sampling to address the class imbalance in the datasets.
- The models were developed with cross-validation to avoid over fitting and obtain consist of performance.
- Performance measures, like Confusion Matrix and Area Under Curve (AUC), was used to compare the performance of the models.

This analysis was conducted using Python through Jupyter notebook. In-built libraries and methods were used to run the machine learning models. When needed, functions were defined to simplify specific analyses or visualizations.

The below diagram shows in detail the full process that was followed in the project.

1.3 LIMITATIONS OF STUDY

In this study, we evaluated the effectiveness of using specific supervised machine learning techniques to solve the problem of fraud detection in financial transactions. The limitations of the methods applied in this study are as follows:

- We used a per-labeled datasets to train the algorithms. However, usually, it is difficult to find labeled data and thus applying supervised machine learning techniques may not be feasible. In such cases, we should evaluate unsupervised techniques which were beyond the scope of this study.
- This study considers digital transactions data that includes amount transacted, the balance of recipient and originator, and time of transaction. These

variables that helped in detecting fraud may not apply to other types of financial transactions, such as credit card fraud.

- We evaluated two machine learning algorithm – Logistic Regression and Random Forest. Although the result of the study using these algorithms is good, it is necessary to evaluate other techniques to determine which algorithm works best for this application.
- Due to the large size of data, we were limited by computation capacity to explore different techniques such as grid search for parameter tuning, SMOTE sampling technique. These techniques may help in further improving the results of this study.

CHAPTER 2
LITERATURE SURVEY

2. LITERATURE REVIEW

Considerable literature is available on financial fraud detection due to its high importance in reducing cyber crimes and also from a business point of view. A few researchers have also conducted literature reviews of articles published in the 2000s and 2010s.

To detect financial fraud, researchers typically use outlier detection techniques (Jaya kumar et.al., 2013) with highly imbalanced datasets. Different types of financial frauds are also possible. One article suggests four categories of financial fraud – financial statement fraud, transaction fraud, insurance fraud and credit fraud (Janus ET Al., 2011). In this project, the focus is on transaction fraud specifically as it applies to mobile payments.

A variety of techniques have been tested to detect financial fraud.

- Phat ET Al., (2004) used Neural Networks, Naive Bayes and Decision Trees to detect automobile insurance fraud.
- Ravisankar et al., (2011) detect financial statement fraud in Chinese companies, another article used SVM, Genetic Programming, Logistic Regression and Neural Networks.
- Density-based clustering (Dharwa et al., 2011) and cost-sensitive Decision Trees (Sahin et al., 2013) have been used for credit card fraud.
- Sorournejad et al., (2016) discusses both supervised and unsupervised machine learning-based approaches involving ANN (Artificial Neural Networks), SVM, HMM (Hidden Markov Models), clustering.

- Wedge ET Al., (2018) address the problem of imbalanced data that result in a very high number of false positives, and some papers propose techniques to alleviate this problem.

However, there is very little literature available on detecting fraudulent transactions in mobile payments, probably due to relatively recent advancements in the technology.

Ashurbanipal ET Al., (2016) present a systematic review of the most used methods in financial fraud detection. The top 5 techniques are shown in the table below:

TECHNIQUES	**FREQUENCY OF USE**
Logistic Regression	13% (17 articles)
Neural Networks	11% (15 articles)
Decision Trees	11% (15 articles)
Support Vector Machines	9% (12 articles)
Naïve Bayes	6% (8 articles)

Table 1: Frequency of use of machine learning techniques in fraud detection problems

CHAPTER 3
METHODOLOGY

3.METHODOLOGY

This methodology served as the deliverable of the project. It describes the results of each phase that was tried out and do a comparison between them to identify which is the best technique to address the fraud detection problem.

Each phase of the project has an output that describes the findings in that phase. These deliverable were used in this final project are explained below –

METHODOLOGY PHASES	**PROJECT DELIVERABLES**
Understanding the data set	• Report on the summary of the data set and each variable it contains along with necessary visualizations
Exploratory Data Analysis	• Report on analysis conducted and critical findings with a full description of data slices considered • Hypothesis about the separation between fraud and non- fraud transactions • Visualizations and charts that show the differences between fraud and non-fraud transactions • Python code of the analysis performed
Modeling	• Report on the results of the different techniques tried out, iterations that were experimented with, data transformations and the detailed modeling approach • Python code used to build machine learning models
Final Project Report	• Final report summarizing the work done over the course of the project, highlighting the key findings, comparing different models and identifying best model for financial fraud detection

Table 2: Project Deliverables

3.1 TOOLS USED

This project was entirely done using Python, and the analysis was documented in a Jupiter notebook. Standard python libraries were used to conduct different analyses. These libraries are described below –

- ***sklearn*** – used for machine learning tasks
- ***seaboard*** – used to generate charts and visualizations
- ***pandas*** – used for reading and transforming the data

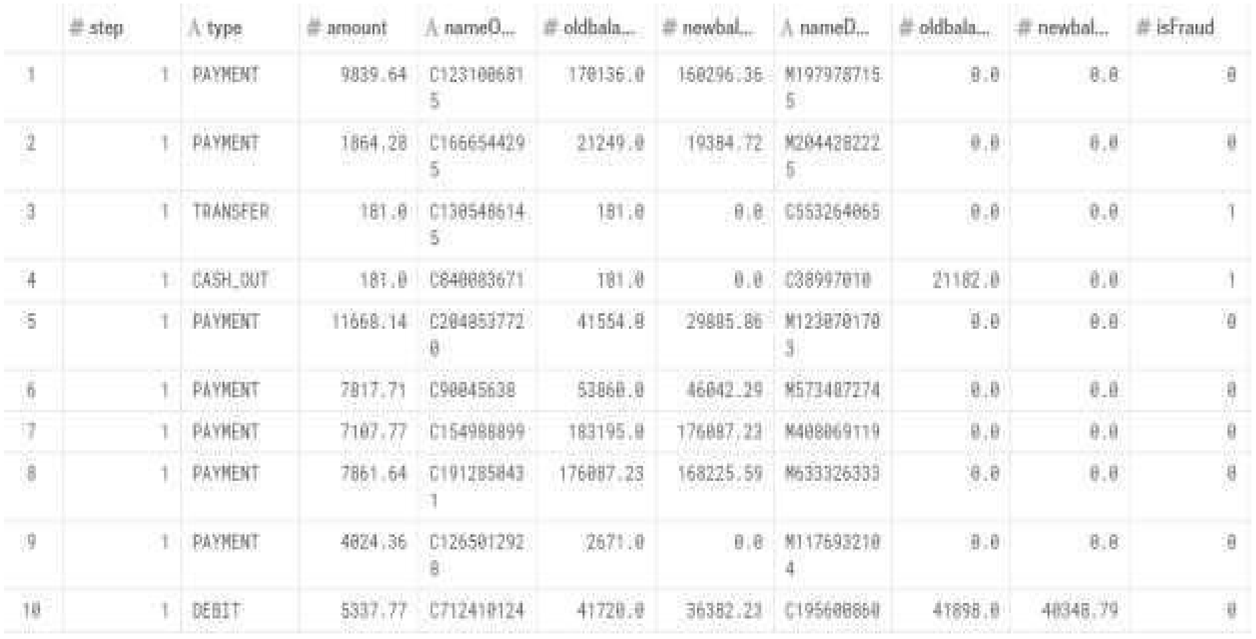
3.2 DATA SOURCES

Due to the private nature of financial data, there is a lack of publicly available datasets that can be used for analysis. In this project, a synthetic datasets, publicly available on Leakage, generated using a simulator called Pay-sim is used. The datasets was generated using aggregated metrics from the private datasets of a multinational mobile financial services company, and then malicious entries were injected. (TESTIMON @ NTNU, Kaggle).

The datasets contains 11 columns of information for ~6 million rows of data. The key columns available are –

- Type of transactions
- Amount transacted
- Customer ID and Recipient ID
- Old and New balance of Customer and Recipient

- Time step of the transaction
- Whether the transaction was fraudulent or not



	# step	A type	# amount	A nameO...	# oldbala...	# newbal...	A nameD...	# oldbala...	# newbal...	# isFraud
1	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
2	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
3	1	TRANSFER	181.0	C1305486145	181.0	0.0	C553264065	0.0	0.0	1
4	1	CASH_OUT	181.0	C840083671	181.0	0.0	C38997010	21182.0	0.0	1
5	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0
6	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.0	0.0	0
7	1	PAYMENT	7107.77	C154988899	183195.0	176087.23	M408069119	0.0	0.0	0
8	1	PAYMENT	7861.64	C1912850431	176087.23	168225.59	M633326333	0.0	0.0	0
9	1	PAYMENT	4024.36	C1265012928	2671.0	0.0	M1176932104	0.0	0.0	0
10	1	DEBIT	5337.77	C712410124	41720.0	36382.23	C195600860	41898.0	40348.79	0

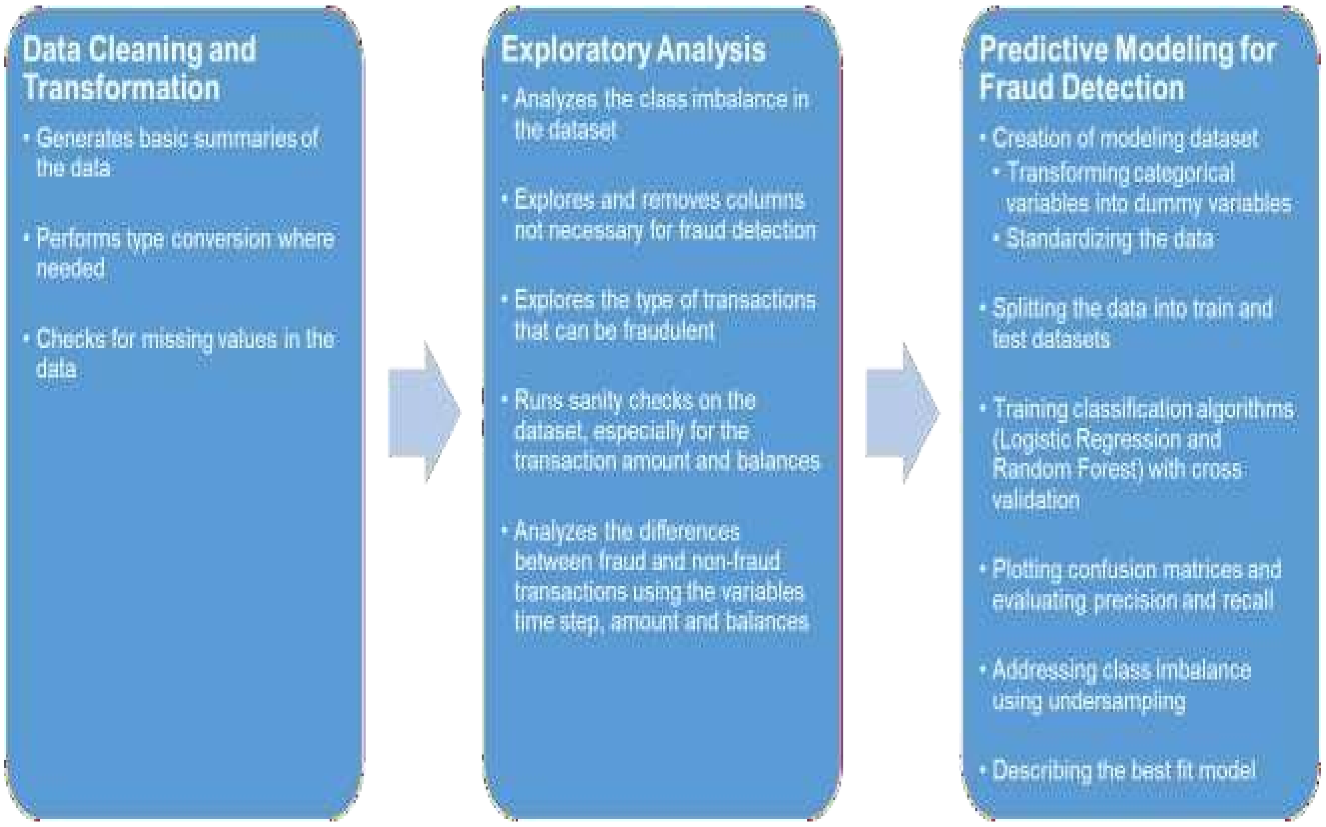
Figure 1: Snapshot of the raw datasets

CHAPTER 4
DATA ANALYSIS

4.DATA ANALYSIS

This section describes each step of the analysis conducted in detail. All analysis is documented in Jupiter notebook format, and the code is presented along with the outputs.

The analysis is split into three main sections. These are described in the diagram below.



Data Cleaning and Transformation
• Generates basic summaries of the data
• Performs type conversion where needed
• Checks for missing values in the data
Exploratory Analysis
• Analyzes the class imbalance in the dataset
• Explores and removes columns not necessary for fraud detection
• Explores the type of transactions that can be fraudulent
• Runs sanity checks on the dataset, especially for the transaction amount and balances
• Analyzes the differences between fraud and non-fraud transactions using the variables time step, amount and balances
Predictive Modeling for Fraud Detection
• Creation of modeling dataset
• Transforming categorical variables into dummy variables
• Standardizing the data
• Splitting the data into train and test datasets
• Training classification algorithms (Logistic Regression and Random Forest) with cross validation
• Plotting confusion matrices and evaluating precision and recall
• Addressing class imbalance using undersampling
• Describing the best fit model

Figure 2: Structure of the analysis

4.1 DETAILED ANALYSIS

The following pages show the step by step process followed in executing the mentioned analysis structure. Relevant code snippets and graphics included are based on Python programming language.

4.2 DATA CLEANING

This section describes the data exploration conducted to understand the data and the differences between fraudulent and non-fraudulent transactions.

4.3 DATA DESCRIPTION

The data used for this analysis is a synthetically generated digital transactions datasets using a simulator called Pay-sim. Pay-sim simulates mobile money transactions based on a sample of real transactions extracted from one month of financial logs from a mobile money service implemented in an African country. It aggregates anonymize data from the private datasets to generate a synthetic datasets and then injects fraudulent transactions.

The datasets has over 6 million transactions and 11 variables. There is a variable named ‘fraud's that indicates actual fraud status of the transaction. This is the class variable for our analysis.

The columns in the datasets are described as follows:

NAME OF THE VARIABLES	**DESCRIPTION**
step	Maps a unit of time in the real world. 1 step is 1 hour of time.
type	Indicates the type of transaction. This can be CASH-IN, CASH-OUT, DEBIT, PAYMENT or TRANSFER
amount	amount of the transaction in local currency
nameOrig	identifier of the customer who started the transaction
oldbalanceOrg	initial balance of the originator before the transaction
newbalanceOrg	originator's balance after the transaction
nameDest	identifier of the recipient who received the transaction
oldbalanceDest	initial balance of the recipient before the transaction
newbalanceDest	recipient's balance after the transaction
isFraud	indicates whether the transaction is actually fraudulent or not. The value 1 indicates fraud and 0 indicates non-fraud

Table 3: Variable in the Datasets

4.4 TYPE CONVERSION

Since it is necessary that all columns in the data are of appropriate type for analysis, we check if there is any need for type conversion. Here are the initial types of the columns read by python.

The is-fraud variable is read as an integer. Since this is the class variable, we convert it to object type. The following python code is used to perform this conversion.

```
# Convert class variables type to object data['isFraud'] =
data['isFraud'].astype('object')
```

4.5 SUMMARY STATISTICS

Before proceeding with the analysis, we present the summary statistics of the variables. In case of numeric variables, we evaluate the mean, standard deviation and the range of values at different percentiles. In case of categorical variables, we evaluate only the number of unique categories, the most frequent category and its frequency.

count	6362620	6362620	6362620	6362620	6362620	6362620
mean	243.40	179861.90	833883.10	855113.67	1100701.67	1224996.4
std	142.33	603858.23	2888242.67	2924048.50	3399180.11	3674128.9
min	1.00	0.00	0.00	0.00	0.00	0.0
25%	156.00	13389.57	0.00	0.00	0.00	0.0
50%	239.00	74871.94	14208.00	0.00	132705.66	214661.4
75%	335.00	208721.48	107315.18	144258.41	943036.71	1111909.2
max	743.00	92445516.64	59585040.37	49585040.37	356015889.35	356179278.9

Figure 3: Summary Statistics of Categorical Variable

4.6 MISSING VALUES CHECK

In this phase, we also check if there are any missing values in the datasets. The following code and output indicate the total number of missing / NA values in all columns, which is zero.

```
# Missing Values Check

print('Maximum number of missing values in any column: ' +
str(data.isnull().sum().max()))
```

```
Maximum number of missing values in any column: 0
```

4.7 CLASS IMBALANCE

In this exploratory analysis, we assess the class imbalance in the datasets. The class imbalance is defined as a percentage of the total number of transactions presented in the is-fraud column.

The percentage frequency output for the is-fraud class variable is shown below:

	Fraud Flag	**Percentage_Transactions**
0	Non-Fraud	99.87
1	Fraud	0.13

As we can see from the figure.10 there is an enormous difference between the percentage_transactions.

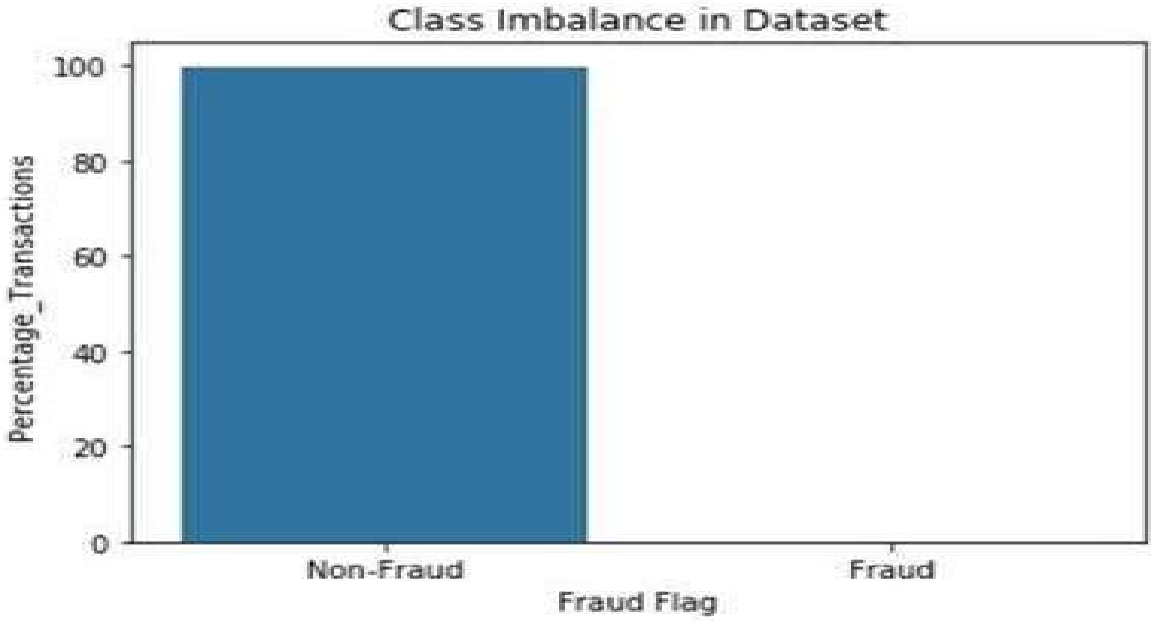


Figure 4: Class Imbalance Visualization

Only 0.13% (8,213) transactions in the datasets are fraudulent indicating high-class imbalance in the datasets. This is important because if we build a machine learning model on this highly skewed data, the non-fraudulent transactions will influence the training of the model almost entirely, thus affecting the results.

4.8 TYPES OF TRANSACTION

In this section, we are exploring the datasets by examining the 'type' variable. We present what the different 'types' of transactions are and which of these types can be fraudulent.

The following plot shows the frequencies of the different transaction types:

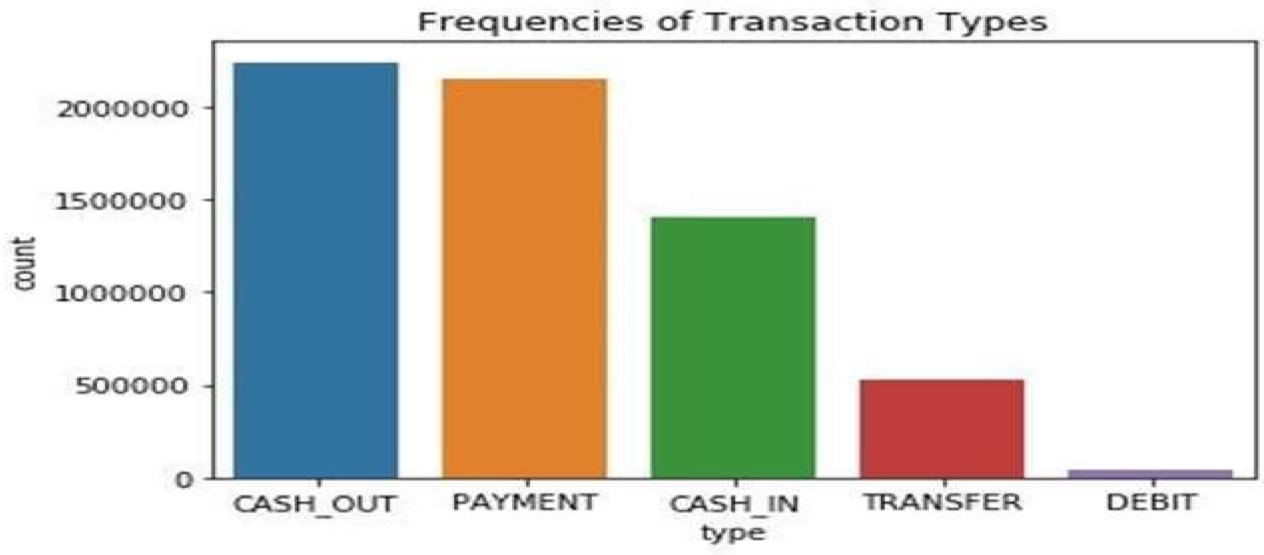


Figure 5: Frequencies of Transaction Types

The most frequent transaction types are **CASH-OUT** and **PAYMENT**.

From the above possible types of transactions, only cash-out and transfer are considered as fraudulent transactions.

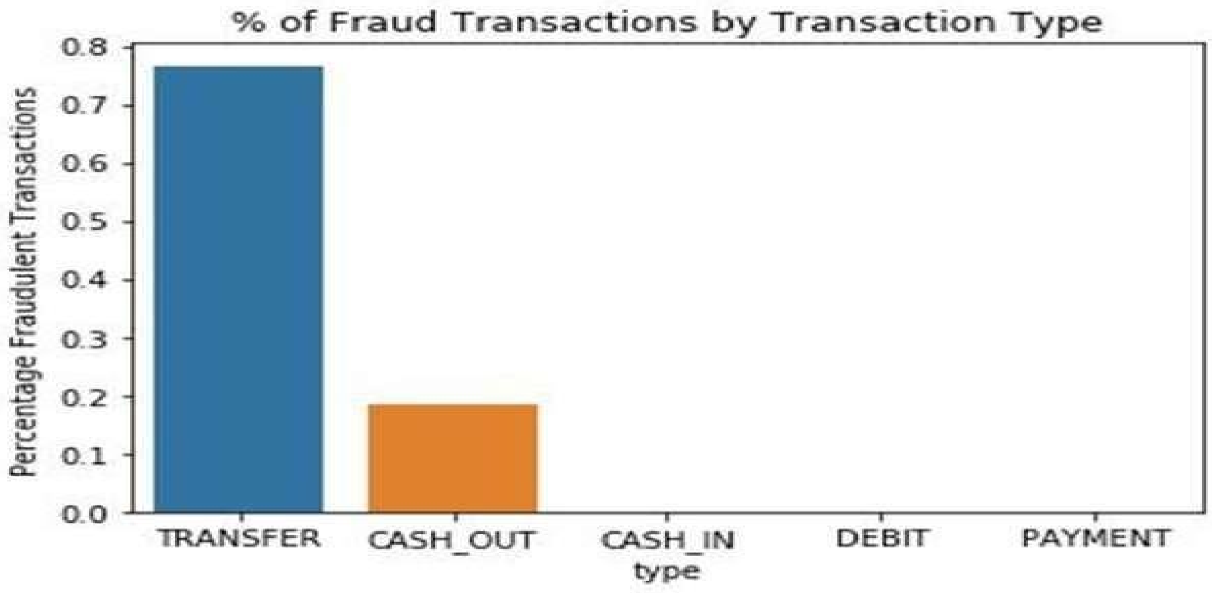


Figure 6: Frequencies of Transaction by Transaction Types

Only **CASH-OUT** and **TRANSFER** transactions can be fraudulent. So, it makes sense to retain only these two types of transactions in our datasets. .

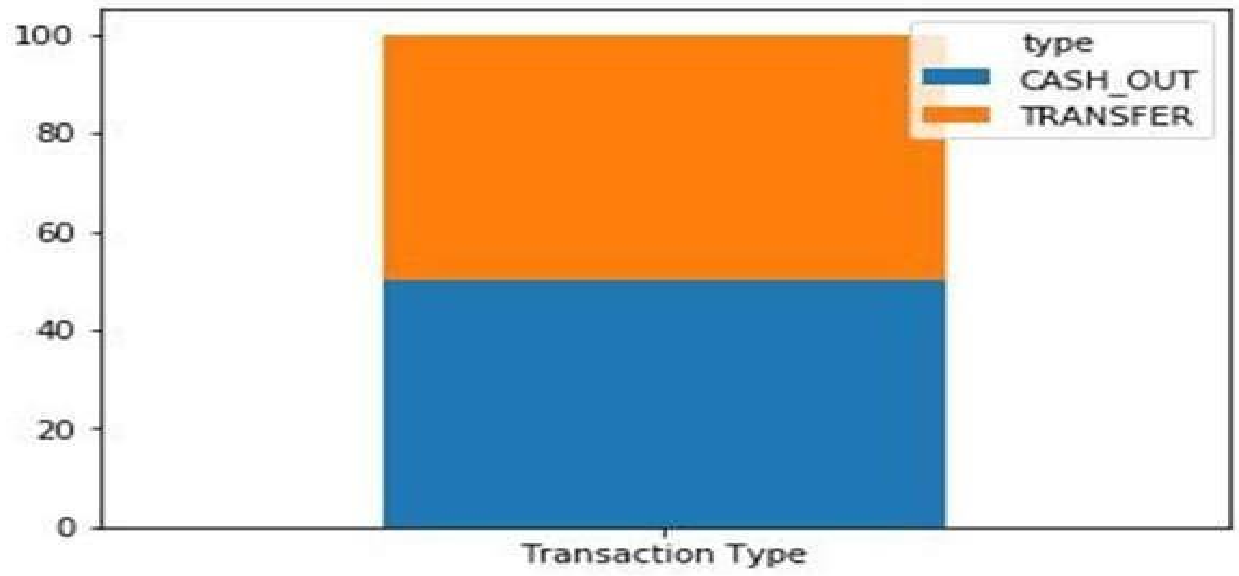


Figure 7: Split of Fraud Transaction by Transaction Type

Therefore, there is an almost equal likelihood that a fraudulent transaction can be **CASH_OUT** or **TRANSFER**.

Since only **CASH-OUT** and **TRANSFER** transactions can be fraudulent, we reduce the size of the datasets by retaining only these transaction types and removing PAYMENT, CASH-IN and DEBIT.

The following code performs and prints the number of new rows in the simplified data.

```
# Retaining only CASH-OUT and TRANSFER transactions
  data = data.loc[data['type'].isin(['CASH_OUT', 'TRANSFER']),:]
  print('The new data now has ', len(data), ' transactions.')
```

```
  The new data now has 2770393 transactions.
```

Therefore, we managed to reduce the data from over 6 million transactions to ~2.8 million transactions.

4.9 DATA SANITY CHECKS

4.1.0 NEGATIVE ARE ZERO TRANSACTION AMOUNT

First, we check if the amount column is always positive. The following two code snippets break this into the number of transactions where the amount is negative and those where the amount is 0.

```
# Check that there are no negative amounts
print('Number of transactions where the transaction amount is negative: ' +
str(sum(data['amount'] < 0)))
```

```
Number of transactions where the transaction amount is negative: 0
```

```
# Check instances where transacted amount is 0

print('Number of transactions where the transaction amount is negative: ' +
str(sum(data['amount'] == 0)))
```

```
Number of transactions where the transaction amount is negative: 16
```

There are only a few cases in which transacted amount is 0. We observe by exploring the data of these transactions that they are all fraudulent transactions. So, we can assume that if the transaction amount is 0, the transaction is fraudulent.

We remove these transactions from the data and include this condition while making the final predictions.

```
# Remove 0 amount values
data = data.loc[data['amount'] > 0,:]
```

4.1.1 ORIGINATORS BALANCE AND RECIPIENT'S BALANCE

In this section, we check if there are any ambiguities in the originator's balance or recipient's balance. The following output identifies instances where originator's initial balance or recipient's final balance is 0.

Percentage of transactions where originators initial balance is 0: 47.23%

Percentage of transactions where destination's final balance is 0: 0.6%

- Therefore, in almost half of the transactions, the originator's initial balance was recorded as 0. However, in less than 1% of cases, the recipient's final balance was recorded as 0.
- Ideally, the recipient's final balance should be equal to the recipient's initial balance plus the transaction amount. Similarly, the originator's final balance should be equal to originator's initial balance minus the transaction amount.
- Then, we check these conditions to see whether the old balance and new balance variables are captured accurately for both originator and recipient.

% transactions where originator balances are not accurately captured: 93.72

% transactions where destination balances are not accurately captured: 42.09

- Therefore, in most transactions, the originator's final balance is not accurately captured, and in almost half the cases, the recipient's final balance is not accurately captured.
- It could be interesting to see if any of the above discrepancies identified vary between fraudulent transactions and non-fraudulent transactions. This will be done in subsequent sections.

4.1.2 FRAUD TRANSACTIONS ANALYSIS

In this section, an additional exploratory analysis is performed to identify if any of the variables can predict a fraud.

TIME STEPS:

We start by analyzing the time step variable. The number of transactions in each time step by fraud status was measured in order to identify if there are any particular time steps where fraudulent transactions are more common than others. From the data description, we know that each time step is an hour.

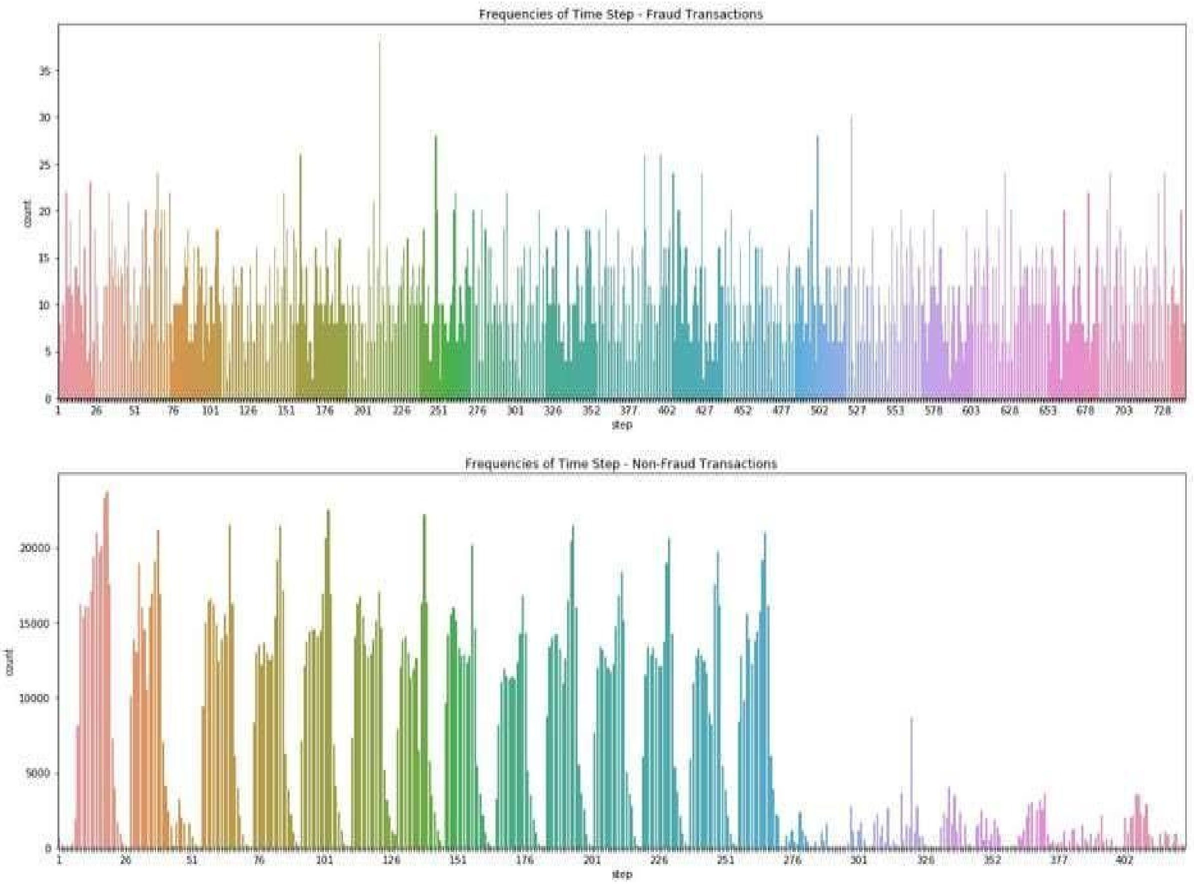


Figure 8: Fraud and Non-Fraud Transactions Count by Time Step

From Figure.19 show that the fraud transactions are almost uniformly spread out across time steps, whereas non-fraudulent transactions are more concentrated in specific time steps. This could be a differentiable between the two categories and can help in the training of the classification models.

TRANSACTION AMOUNT:

We now check if there are any differences between fraud and non-fraud transactions in terms of the transaction amount.

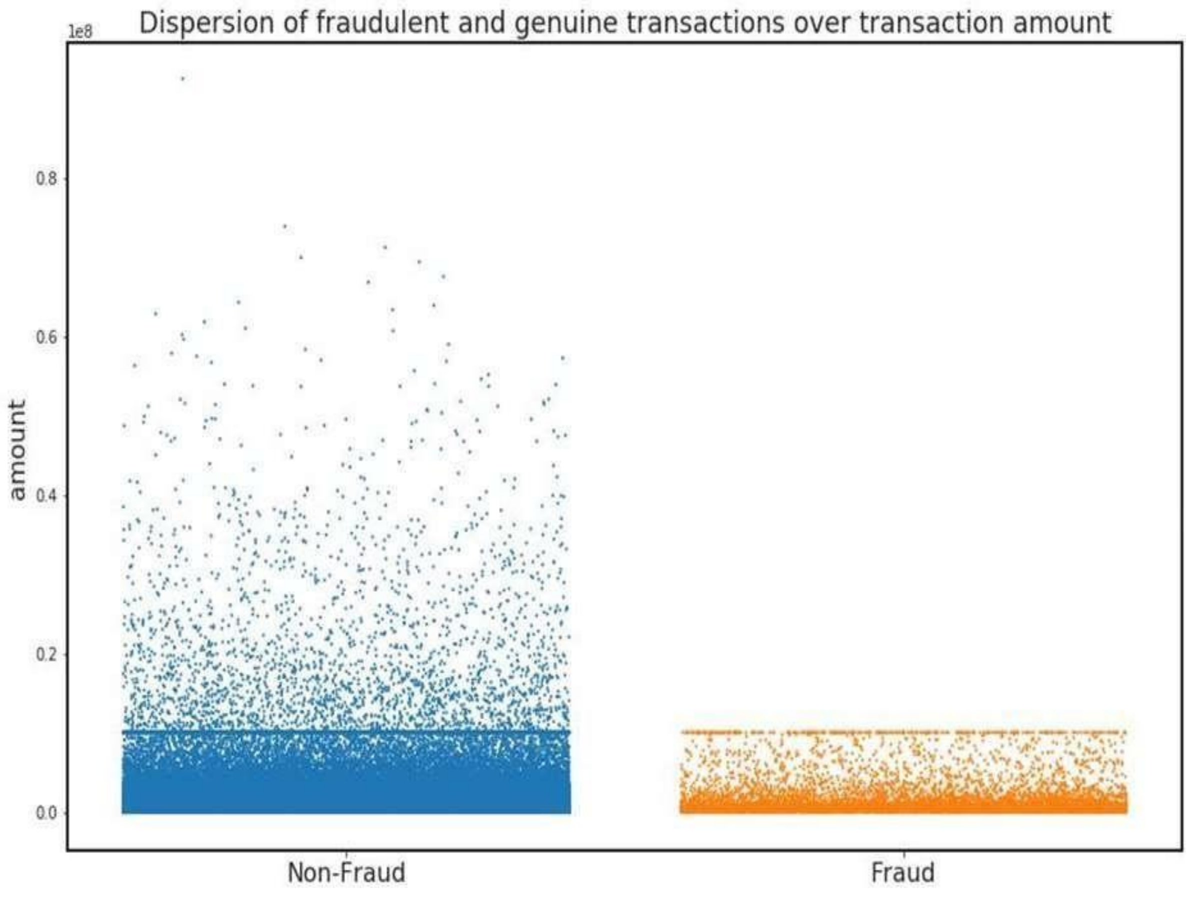


Figure 9: Transaction Amount of Fraud and Non-Fraud Transactions

The distribution of the transaction amount suggests that the amount can be slightly higher for Non-Fraud transactions, but nothing can be said conclusively about differences Fraud and Non-Fraud in terms of the transaction amount.

BALANCES:

In the previous section on Sanity Checks, we noticed that there are inaccuracies in how the ‘balance’ variable is captured for both originator and recipient. We also observed that in almost half the cases, the originator’s initial balance is recorded as 0.

In the below code, we compare the percentage of cases where originator’s initial balance is 0.

% of fraudulent transactions where initial balance of originator is 0: 0.31%

% of genuine transactions where initial balance of originator is 0: 47.37%

- In fraudulent transactions, originator’s initial balance is 0 only 0.3% of the time as compared to 47% in case of non-fraudulent transactions. This could be another potential differentiable between the two categories.
- We check the inaccuracy in the balance variable and compare between fraud and non- fraud. The inaccuracy is defined as the difference between what the balance should be accounting for the transaction amount and what it is recorded as balance.

  We calculate the balance inaccuracies for both the originator and destination as follows:

```
# Defining inaccuracies in originator and recipient balances

data['origBalance_inacc'] = (data['oldbalanceOrg'] - data['amount']) -
data['newbalanceOrig']

data['destBalance_inacc'] = (data['oldbalanceDest'] + data['amount']) -
data['newbalanceDest']
```

In the following figures, we depicted the distribution of the balance inaccuracy feature of originator and destination balances for fraud and non-fraud transactions as below:

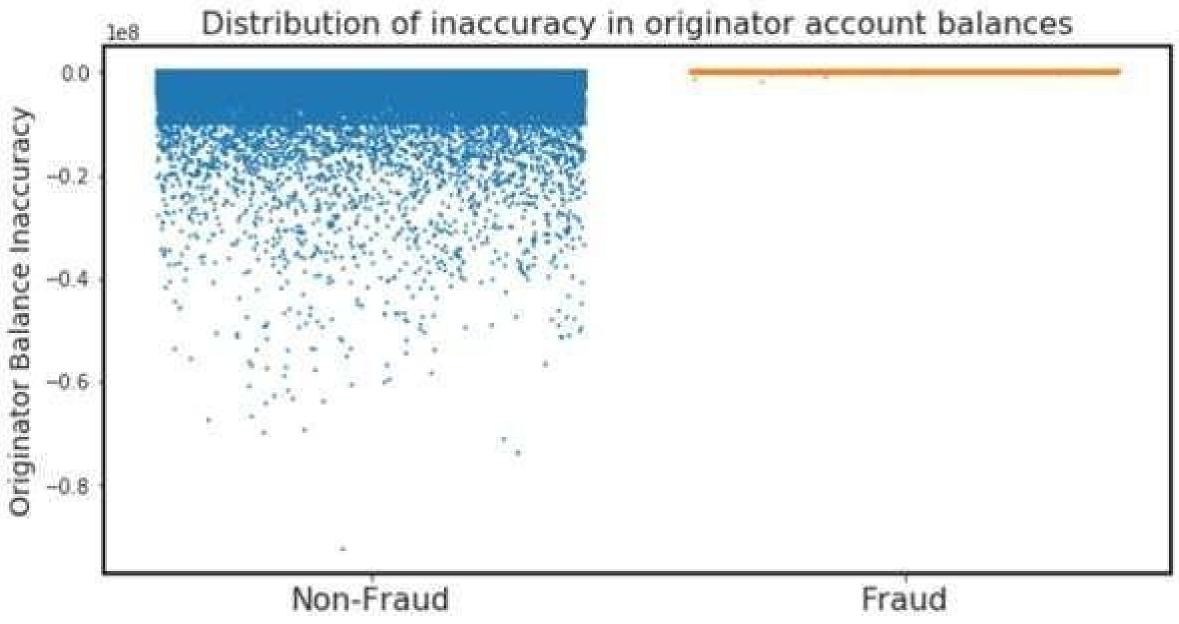


Figure 10: Originator Balance Inaccuracy of Fraud and Non-Fraud Transactions

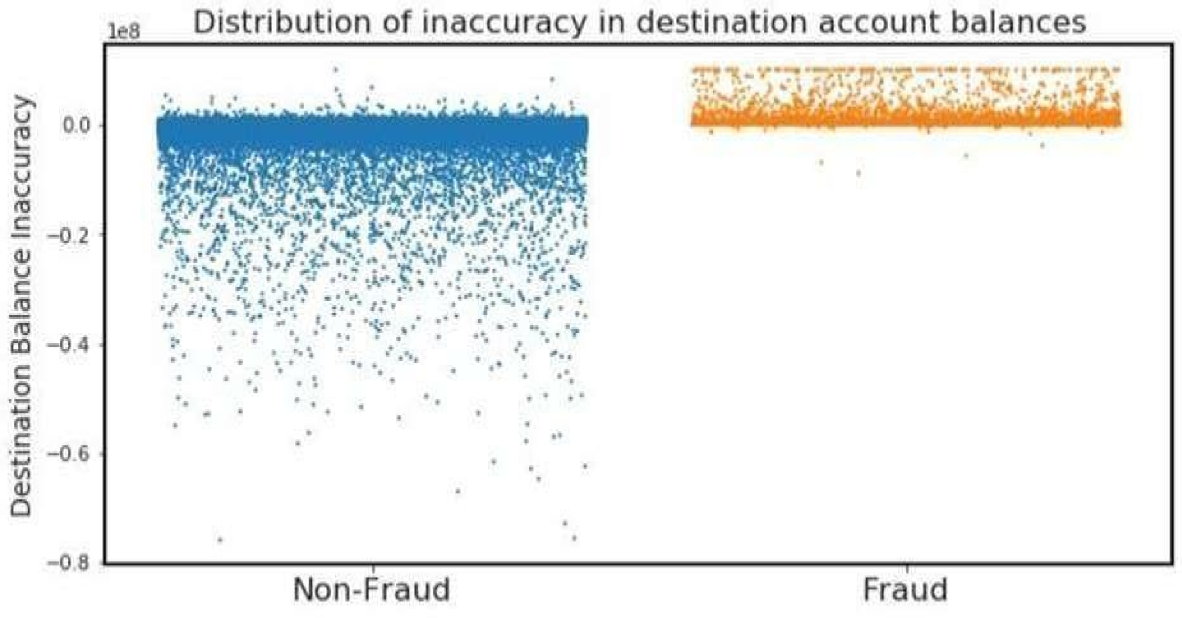


Figure 11: Destination Balance Inaccuracies of Fraud and Non-Fraud Transactions

There are differences between fraud and non-fraud in the inaccuracy measures we analyzed above. In particular, it appears that the inaccuracy in destination balance is almost always negative for non-fraud transactions, whereas it is almost always positive for fraud transactions. This could also be potential predictors of fraud.
Overall, we identified a few dimensions along which fraudulent transactions can be distinguished from non-fraudulent transactions. These are as follows:

- **TIME STEP** - fraudulent transactions have are equally likely to occur in all time steps, but genuine transactions peak in specific time steps

- **BALANCES** - initial balance of originator is much more likely to be 0 in case of genuine transactions than fraud transactions

- **INACCURACY IN BALANCES** - inaccuracy in destination balance is likely to be negative in case of genuine transactions but positive in case of fraud transactions

The below scatter plot shows a clear differentiation between fraudulent and non-fraudulent transactions along time step and destination balance inaccuracy dimensions.

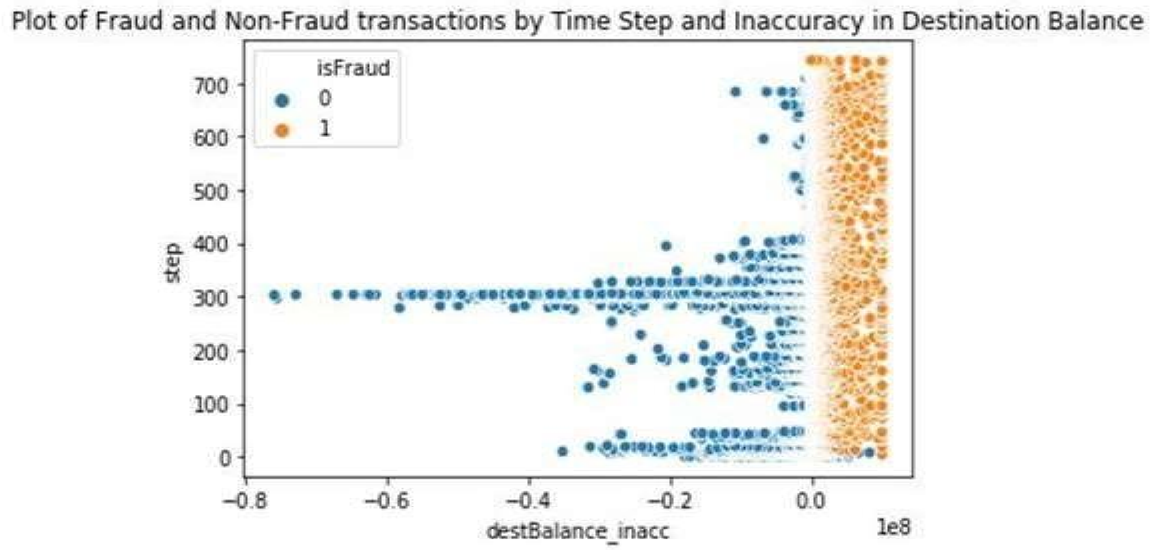


Figure 12: Separation between Fraud and Non-Fraud Transaction

4.1.3 PREDECTIVE MODELING FOR FRAUD DETECTION

In the previous sections, we identified dimensions that make fraudulent transactions detectable. Based on these results, we build supervised classification models.

4.1.4 MODELING DATASET CREATION

In this section, we choose the variables needed for the ML model, encode categorical variables as numeric and standardize the data.

Let us recall columns in the dataset's

Index(['step', 'type', 'amount', 'nameOrig', 'oldbalanceOrg', 'newbalanceOri g', 'nameless, 'counterbalance, 'counterbalance, 'fraud's, 'overbalance_inaccuracy, 'destBalance_inacc'],dtype='object')

The name (or ID) of the originator and destination are not needed for classification. So, we remove them.

```
# Removing name columns

data = data.drop(['nameOrig', 'nameDest'], axis=1)
```

4.1.5 CREATING DUMMY VARIABLES

We have one categorical variable in the datasets – the transaction type. This feature needs to be encoded as binary variables, and dummy variables need to be created. The following code snippet is used to perform this.

```
# Creating dummy variables through one hot encoding for 'type'
column data = pd.get_dummies(data, columns=['type'], prefix=['type']
```

This creates two binary dummy variables – **type_CASH_OUT** and **type_TRANSFER**.

4.1.6 STANDARDIZING THE DATA

In this transformation, we convert all columns in the data to have the same range. This is done through the standard scales feature available in python. The following code snippet is used to perform this transformation.

```
# Normalization of the dataset
std_scaler = StandardScaler()
data_scaled =
    pd.DataFrame(std_scaler.fit_transform(data.loc[:,~data.columns.isin(['isFraud'])
    ])) data_scaled.columns = data.columns[:-1] data_scaled['isFraud'] =
    data['isFraud']
```

4.1.7 CREATE TRAIN AND TEST DATASETS

We split the scaled datasets into training and testing datasets. We decide to use 70% of the original data for training and the remaining 30% for testing.

The following code snippet is used to create training and testing datasets.

```
X = data_scaled.loc[:, data_scaled.columns != 'isFraud']
y = data_scaled.loc[:, data_scaled.columns == 'isFraud']

X_train_original,    X_test_original,    y_train_original,    y_test_original    =
train_test_split(X,y,test_size = 0.3, random_state = 0)

label_encoder      =      LabelEncoder()      y_train_original      =
label_encoder.fit_transform(y_train_original.values.ravel())  y_test_original  =
label_encoder.fit_transform(y_test_original.values.ravel())
```

Then we check whether the class imbalance in train and test datasets are similar. The following code output indicates the % of transactions that are fraud in the two datasets –

```
Class imbalance in train dataset: 0.297%
```

Class imbalance in test dataset 0.291%

Therefore, the class imbalance is similar, and we can proceed with the training of the algorithms.

4.1.8 CLASSIFICATION MODELS FOR FRAUD DETECTION

- We define two models to perform the classification: **Logistic Regression** and **Random Forest**.
- To measure the performance of the models, Recall is a useful metric. High-class imbalance datasets typically result in poor Recall, although accuracy may be high. Precision will also be a consideration because reduced precision implies that the company that is trying to detect fraud will incur more cost in screening the transactions. In fraud detection problems, though, accurately identifying fraudulent transactions is more critical than incorrectly classifying legitimate transactions as fraudulent.
- Alternatively, we could also go with Area Under Curve (AUC) of the ROC curve. However, this will not adequately capture if the model is correctly identifying most of the fraudulent transactions. Therefore, we use this as a validation of the model performance.

The following code snippet is used to define the accuracy of the two models.

```
accuracy_dict = {}
model_lr = LogisticRegression()
model_rf = RandomForestClassifier()
```

```
scr = 'recall'
```

We also need to do cross-validation to ensure the models do not over-fit the training data. For this, we use Stratified 5-fold since we need to ensure that the class imbalance is retained in the validation sets.

```
skf = StratifiedKFold(5)
```

4.1.9 LOGISTIC REGRESSION MODEL

In this section, we train the logistic regression model and calculate the mean recall score.
This parameter will serve as a benchmark for further experiments.

```
sc_lr = cross_val_score(model_lr, X_train_original, y_train_original, cv=skf, scoring=scr)
```

The following output indicates how the Logistic Regression model performs on the training datasets.

```
Logistic Regression's average recall score across validation sets is: 50.67%
```

Therefore, the default Logistic Regression model is able to capture only half of the actual Fraud cases.

We plot the confusion matrix's for the train and test datasets of the logistic regression model, and we check the precision and recall in each case.

```
sc_lr = cross_val_score(model_lr, X_train_original, y_train_original, cv=skf,
scoring=scr)
```

The following output indicates how the Logistic Regression model performs on the training datasets.

```
Logistic Regression's average recall score across validation sets is: 50.67
```

Therefore, the default Logistic Regression model is able to capture only half of the actual Fraud cases.

We plot the confusion matrix's for the train and test datasets of the logistic regression model, and we check the precision and recall in each case.

Precision: 91.03%

Recall: 50.88%

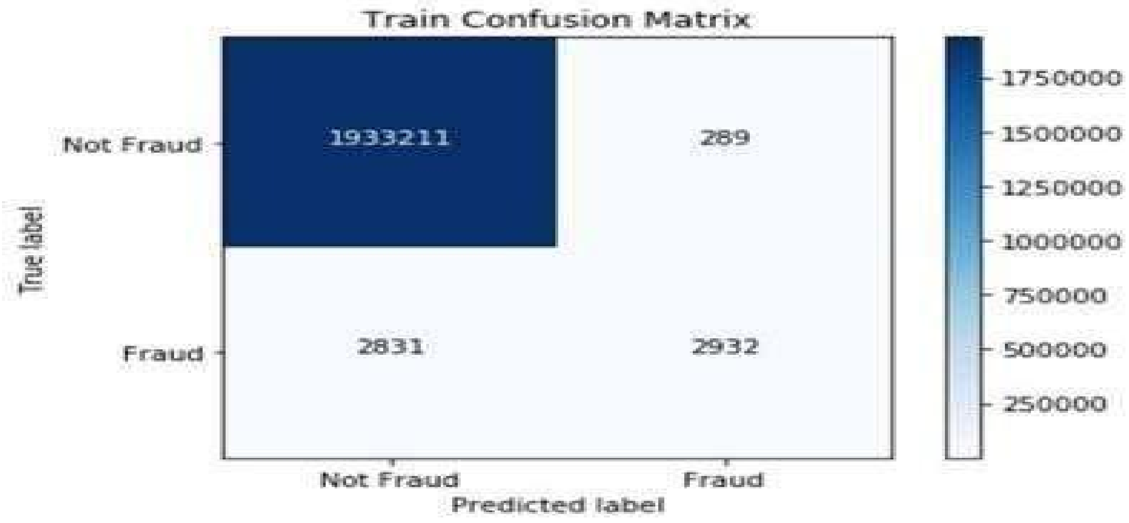


Figure 13: Logistic Regression - Train Confusion Martix

Precision: 90.12%
Recall: 51.7%

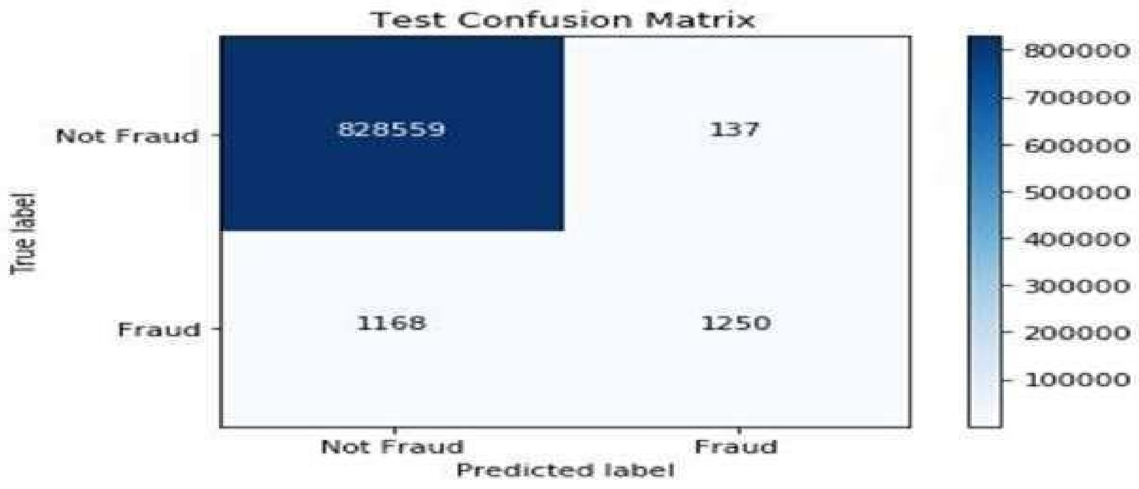


Figure 14 : Logistic Regression - Test Confusion Martix

From the above results, there are two results:

- The training and testing datasets are consistent, and there is no over fitting.
- High precision and low Recall indicate that running the algorithm on the data with high-class imbalance will not provide excellent results.

4.2.0 RANDOM FOREST MODEL

In this section, we repeat the same steps using a different classification algorithm such as Random Forest, and we calculate the mean recall score. We can compare with the Logistic Regression model to evaluate which is to perform better.

```
sc_rf = cross_val_score(model_rf, X_train_original, y_train_original, cv=skf,
scoring=scr)
```

The following output indicates how the Random Forest model performs on the training datasets.

```
Random Forest's average recall score across validation sets is: 99.48%
```

The Random Forest model seems to produce excellent results on the training datasets. Again, we plot the confusion matrices for the training and testing datasets and we check the precision and recall in each case.

Precision: 100.0%

Recall: 99.84%

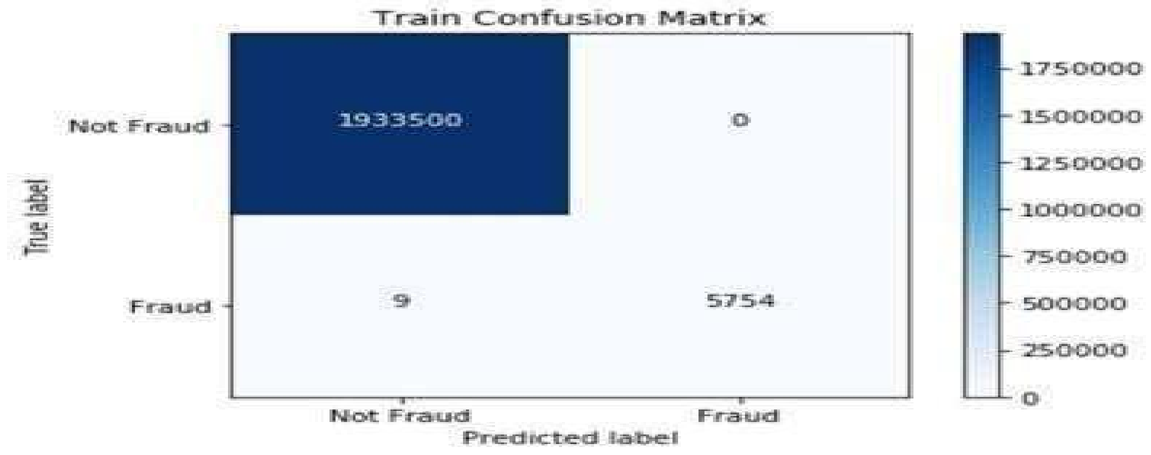


Figure 15: Random Forest - Train Confusion Matrix

Precision: 100.0%

Recall: 99.79%

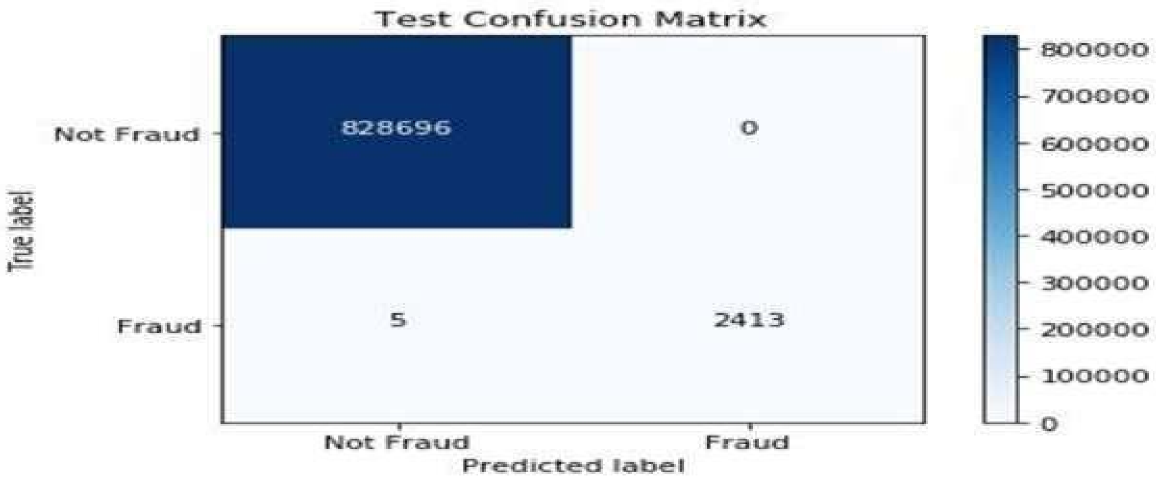


Figure 16: Random Forest - Test Confusion Matrix

The Random Forest algorithm gives almost perfect results. Comparing the recall scores with Logistic Regression, Random Forest performs much better in detecting fraud.

Also, the performance of the Random Forest model is consistent between the training and testing datasets. So, there is no over fitting.

The following table compares the results of the two models:

MODEL	**TRAIN PRECISION**	**TRAIN RECALL**	**TEST PRECISION**	**TEST RECALL**
Logistic Regression	91.03%	50.88%	90.12%	51.7%
Random Forest	100%	99.84%	100%	99.79%

Table 4: Comparison of Result of Logistic Regression and Random Forest

Regardless of the positive results from the Random Forest model, we should try to improve the results of Logistic Regression through parameter tuning and by addressing the class imbalance. In the following section, we present these techniques.

4.2.1 ADDRESSING CLASS IMBALANCE

There are many techniques to address high-class imbalanced datasets. A few examples are as follows –

- **UNDER SAMPLING:** In this method, random samples from the majority class are deleted so that the class imbalance is more manageable.

- **OVER SAMPLING:** In this method, observations of the minority class are resampled with repetition to increase their presence in the data

- **SMOTE:** This is a type of oversampling, but instead of repeating the observations, it synthesizes new plausible observations of the minority class.

- We use undersampling as it is less computation-intensive. We also do this only for the logistic regression model as the random forest model is already giving excellent results. The aim is to check if it is possible to get better performance than what we observed with the Random Forest model.
- We train the Logistic Regression model on a subset of the original training datasets. We retain all the fraud cases and randomly select an equal number of non-fraud cases to create an under-sampled training datasets.

The following code snippet is used to do this

```
# Undersampling the training dataset

fraud_indices_train = np.where(y_train_original == 1)[0] non_fraud_indices_train
= np.where(y_train_original == 0)[0]

undersample_non_fraud_indices_train =
np.random.choice(non_fraud_indices_train, len(fraud_indices_train), replace =
False)                    undersample_non_fraud_indices_train              =
np.array(undersample_non_fraud_indices_train)

undersample_indices_train        =        np.concatenate([fraud_indices_train,
undersample_no n_fraud_indices_train])

X_train_undersample =
X_train_original.loc[X_train_original.reset_index(drop=True).index.isin(undersa
mple_i ndices_train),:]
y_train_under sample = y_train_original[under sample_indices_train.to-list()]
```

Following code, the output indicates the number of transactions in the under sampled data – There are 11526 rows in the under sampled training data.

LOGISTIC REGRESSION PARAMETER TUNING

We now identify the best Logistic Regression model for the under-sampled datasets by tuning the 'Cost function' and 'Regularization factor' parameters. The following output describes the recall scores for different combinations of the penalty and cost function.

```
Recall of Logistic Regression for l1 penalty and C = 0.001 is: 0.0%
Recall of Logistic Regression for l1 penalty and C = 0.01 is: 22.22%
Recall of Logistic Regression for l1 penalty and C = 0.1 is: 41.02%
Recall of Logistic Regression for l1 penalty and C = 1 is: 43.83%
Recall of Logistic Regression for l1 penalty and C = 10 is: 44.15%
Recall of Logistic Regression for l1 penalty and C = 100 is: 44.16%
Recall of Logistic Regression for l1 penalty and C = 1000 is: 44.16%
Recall of Logistic Regression for l2 penalty and C = 0.001 is: 43.21%
Recall of Logistic Regression for l2 penalty and C = 0.01 is: 44.13%
Recall of Logistic Regression for l2 penalty and C = 0.1 is: 44.16%
Recall of Logistic Regression for l2 penalty and C = 1 is: 44.16%
Recall of Logistic Regression for l2 penalty and C = 10 is: 44.16%
Recall of Logistic Regression for l2 penalty and C = 100 is: 44.16%
Recall of Logistic Regression for l2 penalty and C = 1000 is: 44.16%
```

Therefore, the best Logistic Regression model with under sampling (l1 penalty and C of 100) has a recall of <50%.

The default random forest model performs better than logistic regression-
model.

4.2.2 BEST FIT MODEL DETAILS

The Random Forest model gave the best results above. The parameters of this model are presented in the following code.

```
<bound method BaseEstimator.get_params of
RandomForestClassifier(bootstrap=Tr
ue,class_weight=None,criterion='gini',max_depth=None,max_features='au
to',
max_leaf_nodes=None,min_impurity_decrease=0.0,min_impurity_split=No
ne,
min_samples_leaf=1,min_samples_split=2,min_weight_fraction_leaf=0.0,
n_estimators=10, n_jobs=None, oob_score=False, random_state=None,
verbose=0,  warm_start=False)>
```

The model uses 10 trees in the forest (n_estimators) and has an infinite max depth. Positive cross- validation results remove the possibility of overfilling.

In the following figure, we present the relative feature importance of the random forest model. The following plot shows which variables are contributing more to make the fraud prediction.

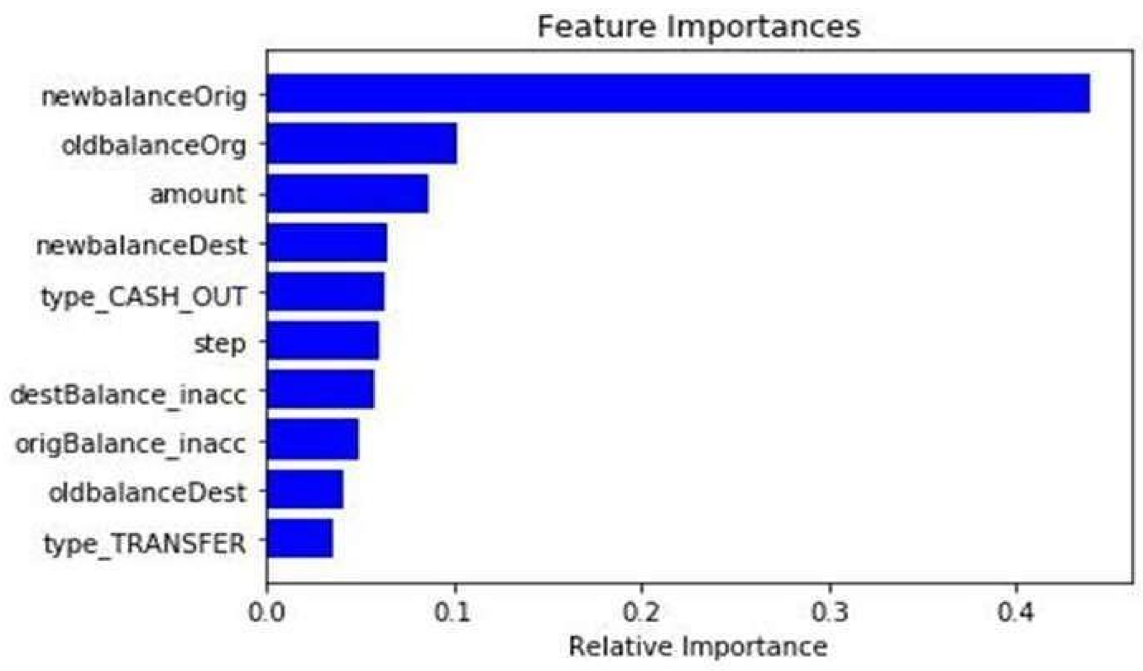


Figure 17: Random Forest Model Feature Importance

Therefore, the balance of the originator ("counterbalance") feature is critical to making the prediction as compared to all other variables.

For the Receiver-Operator Characteristics (ROC) curve and calculate Area-Under- Curve (AUC) for this model is depicted in the following figure.

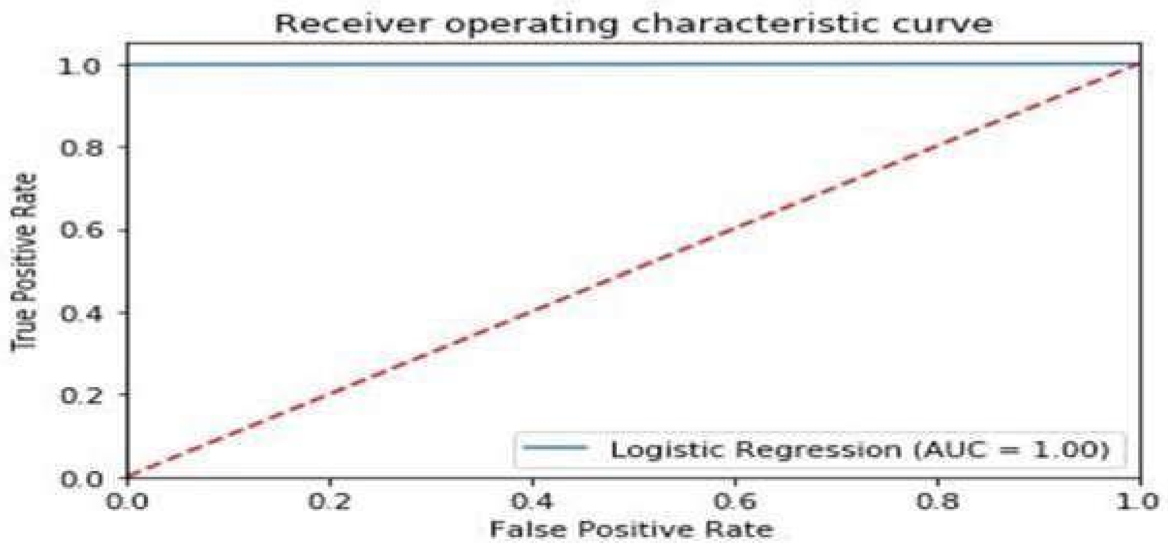


Figure 18 : ROC curve of Random Forest Model

4.2.3. ANALYSIS SUMMARY

We analyzed the financial transactions data and developed a machine learning model to detect fraud. The analysis included data cleaning, exploratory analysis and predictive modeling.

In the data cleaning, we checked for missing values, converted data types and summarized the variables in the data. In an exploratory analysis, we looked at the class imbalance, and deep-dived into each of the variables, in particular transaction type, transaction amount, balance and time step. We identified derived variables that can help with fraud detection. We also plotted various graphs to better visualize the data and come up with insights.

In predictive modeling, we experimented with Logistic Regression and Random Forest algorithms. We observed that Random Forest performs best for this application with almost 100% precision and recall scores. We tried to improve the logistic regression results by under sampling, but the results were the same because of a lot of the data is excluded. We ensured that there is no overfilling in the models through cross-validation.

We can conclude that fraud detection in financial transactions is successful in this labeled datasets, and the best algorithm for this purpose is Random Forest.

CHAPTER 5
CONCLUSION

5.CONCLUSION

In conclusion, we successfully developed a framework for detecting fraudulent transactions in financial data. This framework will help understand the nuances of fraud detection such as the creation of derived variables that may help separate the classes, addressing class imbalance and choosing the right machine learning algorithm.

We experimented with two machine learning algorithms – Logistic Regression and Random Forest. The Random Forest algorithm gave far better results than Logistic Regression indicating tree-based algorithms work well for transactions data with well- differentiated classes. This also emphasizes the usefulness of conducting rigorous exploratory analysis to understand the data in detail before developing machine learning models. Through this exploratory analysis, we derived a few features that differentiated the classes better than the raw data.

5.1 RECOMMANDATIONS

Through this project, we demonstrated that it is possible to identify fraudulent transactions in financial transactions data with very high accuracy despite the high-class imbalance. We provide the following recommendations from this exercise -

- Fraud detection in transactions data where transaction amount and balances of the recipient and originator are available can be best performed using tree-based algorithms like Random Forest.

- Using dispersion and scatter plots to visualize the separation between fraud and non-fraud transactions is essential to choose the right features.

- To address the high-class imbalance typical in fraud detection problems, sampling techniques like under sampling, oversampling, SMOTE can be used. However, there are limitations in terms of computation requirements with these approaches, especially when dealing with big data sets.

- To measure the performance of fraud detection systems, we need to be careful about choosing the right measure. The recall parameter is a good measure as it captures whether a good number of fraudulent transactions are correctly classified or not. We should not rely only on accuracy as it can be misleading.

REFERENCES

1. E. Ngai et.al., *The Application of Data Mining Techniques in Financial Fraud Detection:*

   A Classification Framework and an Academic Review of Literature, Decision Support Systems. 50, 2011, 559–569
2. Albashrawi et.al., *Detecting Financial Fraud Using Data Mining Techniques: A Decade Review from 2004 to 2015*, Journal of Data Science 14(2016), 553-570
3. TESTIMON @ NTNU, *Synthetic Financial Datasets for Fraud Detection*, Kaggle, retrieved from https://www.kaggle.com/ntnu-testimon/paysim1
4. Jaya kumar et.al., *A New Procedure of Clustering based on Multivariate Outlier Detection*. Journal of Data Science 2013; 11: 69-84
5. Jans et.al, *A Business Process Mining Application for Internal Transaction Fraud Mitigation*, Expert Systems with Applications 2011; 38: 13351–13359
6. Phua et.al., *Minority Report in Fraud Detection: Classification of Skewed Data*. ACM SIGKDD Explorations Newsletter 2004; 6: 50-59.
7. Dharwa et.al., *A Data Mining with Hybrid Approach Based Transaction Risk Score Generation Model (TRSGM) for Fraud Detection of Online Financial*

Transaction, International Journal of Computer Applications 2011; 16: 18-25.

8. Sahin et.al., *A Cost-Sensitive Decision Tree Approach for Fraud Detection*, Expert Systems with Applications 2013; 40: 5916–5923.

9. Sorournejad et.al., *A Survey of Credit Card Fraud Detection Techniques: Data and Technique Oriented Perspective,* 2016

10. Wedge et.al., *Solving the False Positives Problem in Fraud Prediction Using Automated Feature Engineering*, Machine Learning and Knowledge Discovery in Databases, pp 372-388, 2018