A Project Report

on

MONEY LAUNDERING DETECTION USING MACHINE LEARNING METHODS

Submitted in partial fulfillment of the requirements for the award of degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

17WH1A0513	K. THANMAI
17WH1A0530	G. NYMISHA SANKAR
17WH1A0534	G. SRIVIDYA

Under the esteemed

guidance of

Mr. K. NARESH

Assistant Professor

Department of Computer Science & Engineering

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited EEE.ECE.CSE.IT B.Tech Courses,

Accredited to NAAC with 'A' Grade)

(Approved by AICTE, New Delhi and Affiliated JNTUH,Hyderabad)

Bachupally, Hyderabad – 500090

MAY, 2021

BVRIT HYDERABAD

College of Engineering for Women

(NBA Accredited EEE.ECE.CSE.IT B.Tech Courses,

Accredited to NAAC with 'A' Grad

(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the Project Work entitled "**MONEY LAUNDERING DETECTION USING MACHINE LEARNING METHODS**" is a bonafide work carried out by **Ms. K.THANMAI (17WH1A0513), Ms. G.NYMISHA SANKAR (17WH1A0530), Ms. G.SRIVIDYA (17WH1A0534),** in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide
Mr. K. Naresh
Asst. Professor, CSE

Head of the Department
Dr. Srinivasa Reddy Konda
Professor, CSE

External Examiner

DECLARATION

We hereby declare that the work presented in this project entitled **“MONEY LAUNDERING DETECTION USING MACHINE LEARNING METHODS”** submitted towards completion of Project Work in IV year of B.Tech., CSE at ‘BVRIT HYDERABAD College of Engineering For Women’ Hyderabad is an authentic record of our original work carried out under the guidance of Mr. K.Naresh, Assistant Professor, Department of CSE

Roll No.	**Name**	**Signature**
17WH1A0513	Ms. K. Thanmai	
17WH1A0530	Ms. G. Nymisha Sankar	
17WH1A0534	Ms. G. Srividya	

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr.K.V.N.Sunitha**, **Principal, BVRIT HYDERABAD College of Engineering for Women** for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. Srinivasa Reddy Konda, Head of Department of CSE, BVRIT HYDERABAD College of Engineering for Women** for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Ms. K. Naresh, Asst. Professor, CSE, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance and encouragement throughout the project.

We express our gratitude to our Major Project coordinator **Dr. Ganti Naga Satish, Professor, CSE**, **BVRIT HYDERABAD College of Engineering for Women** for his valuable suggestions in completing our work successfully.

Finally, We would like to thank all our faculty and Staff of CSE department who helped us directly or indirectly. Last but not least, we wish to acknowledge our **Parents** and **Friends** for giving moral strength and constant encouragement.

K. Thanmai	**(17wh1a0513)**
G. Nymisha Sankar	**(17wh1a0530)**
G. Srividya	**(17wh1a0534)**

TABLE OF CONTENTS

ABSTRACT

Money Laundering is the process of creating the appearance that large amounts of money obtained from serious crimes, such as drug trafficking or terrorist activity, originated from a legitimate source. Through money laundering, the launderer transforms the monetary proceeds derived from criminal activity into funds with an apparently legal source. The system that works against Money laundering is Anti-Money Laundering (AML) system. The existing system for Anti-Money Laundering accepts the bulk of data and converts it to large volumes reports that are tedious and topsy-turvy for a person to read without any help of software. To develop a structure to research in datamining, we create a taxonomy that combines research on patterns of observed fraud schemes with an appreciation of areas that benefit from a productive application of data mining. The aim of this study was to review research conducted in the field of fraud detection with an emphasis on detecting honey laundering and examine deficiencies based on data mining techniques. Which include a set of predefined rules and threshold values. In addition to this approach, data mining techniques are very convenient to detest money laundering patterns and detect unusual behavior. Therefore, unsupervised data mining technique will be more effective to detect new patterns of money laundering and can be crucial to enhance learning models based on classification methods. Of course, the development of new methods will be very useful to increase the accuracy of performance.

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

Detecting management fraud is a difficult task when using normal audit procedures. First, there is a shortage of knowledge concerning the characteristics of management fraud. Secondly, given its infrequency, most auditors lack the experience necessary to detect it. Finally, managers deliberately try to deceive auditors. For such managers, who comprehend the limitations of any audit, standard auditing procedures may prove insufficient. These limitations suggest that there is a need for additional analytical procedures for the effective detection of management fraud. It has also been noted that the increased emphasis on system assessment is at odds with the profession's position regarding fraud detection since most material frauds originate at the top levels of the organization, where controls and systems are least prevalent and efficient. Applying data mining to fraud detection as part of a routine financial audit can be challenging and, as we will explain, data mining should be used when the potential payoff is high. In general, when it comes to fraud detection for a given audit client, the audit team would make three major decisions: (1)What specific types of fraud (e.g., revenue recognition, understated liabilities, etc.) should be included in the audit plan for a particular client? (2) What sources of data (e.g., journal entries, emails, etc.) would be provided evidence of each type of fraud? (3) Which data mining technique(s) (e.g., directed or undirected techniques) would be the most effective for finding potential evidence of fraud in the selected data? Developing answers for each of these questions are significant individually, but, in combination, answering these questions is challenging.

Money Laundering

Money laundering is the process of taking cash earned from illicit activities such as drug trafficking, and making the cash appears to be earnings from a legal business activity. The money from the illicit activity is considered dirty and the process “launders” the money to make it look clean. Money laundering is the generic term used to describe the process by which criminals disguise the original ownership and control of the proceeds of criminal conduct by making such proceeds appear to have derived from a legitimate source. Illegally earned money needs laundering for the criminal organization to use it effectively. Dealing with large amounts of illegal cash is inefficient and dangerous. The criminals need a way to deposit the money in financial institutions, yet they can only do so if the money appears to come from legitimate sources. There are many ways to launder money. These methods span from the very simple to the very complex. One of the most common ways is to launder the money through a legitimate cash-based business owned by the criminal organization.

For instance, if the organization owns a restaurant, it might inflate the daily cash receipts to funnel its illegal cash through the restaurant and into the bank. Then they can distribute the funds to the owners out of the restaurant's bank account.

Steps of Money Laundering

Money-laundering is a dynamic three-stage process that requires

a. **Placement:** This is the movement of cash from its source. On occasion, the source can be easily disguised or misrepresented. This is followed by placing it into circulation through financial institutions, casinos, shops, bureau de change and other businesses, both local and abroad. The process of placement can be carried out through many processes.

b. **Layering:** The purpose of this stage is to make it more difficult to detect and uncover a laundering activity. It is meant to make the trailing of illegal proceeds difficult for the law enforcement agencies.

c. **Integration:** This is the movement of previously laundered money into the economy mainly through the banking system, and thus such monies appear to be normal business earnings. This is dissimilar to layering, for in the integration process detection and identification of laundered funds is provided through informants.

1.1 Objectives

This project attempts to Apply data mining to fraud detection as part of a routine financial audit. application of data mining to fraud detection becomes more manageable and will have a higher potential for a successful payoff. We also recognize that data mining techniques and associated software can have a steep learning curve. Further, if used improperly, data mining can produce many false positives and spurious patterns that will require auditors to expend time to subsequently investigate.

1.2 Methodology

To classify whether a transaction is fraud or legal, a large collection of transaction is required. The dataset is downloaded from Kaggle databse. In this section the methodology followed is discussed in detail.

1.2.1 DataSet

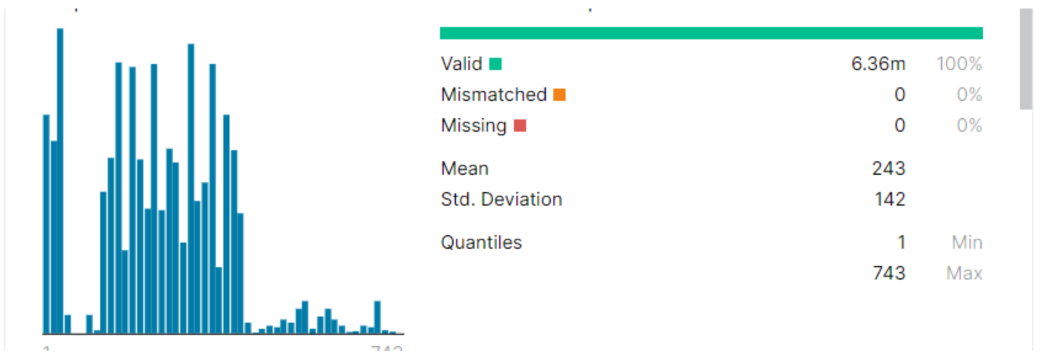
Proper and large dataset is required for all classification research during the training and the testing phase. The dataset for the project is downloaded from Kaggle. It is Each step represents an hour of simulation. This dataset is scaled down 1/4 of the original dataset which is presented in the paper "PaySim: A financial mobile money simulator for fraud detection". It contains a collection of 6353307 transactions. That produce an output of either 1 or 0. Where 1 indicates that a transaction is legal and on the other hand 0 indicates that the transaction is fraudulent.

Download the dataset for training

Before you start any training, you'll need a set of transactions to teach the model about the classes you want to recognize which we have already downloaded for the algorithm to predict.

Attributes of the Dataset

1. **Step:**
   Maps a unit of time in the real world. In this case 1 step is 1 hour of time.

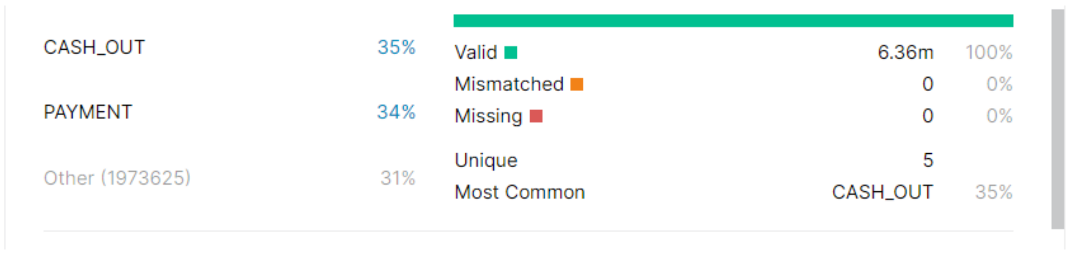


Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	243	
Std. Deviation	142	
Quantiles	1	Min
	743	Max

Fig 1: Step attribute statistics

2. Type:

Indicates the type of transaction being performed. The type of transactions that are recognised are, CASH-IN, CAHS-OUT, DEBT, PAYMENT, TRANSFER



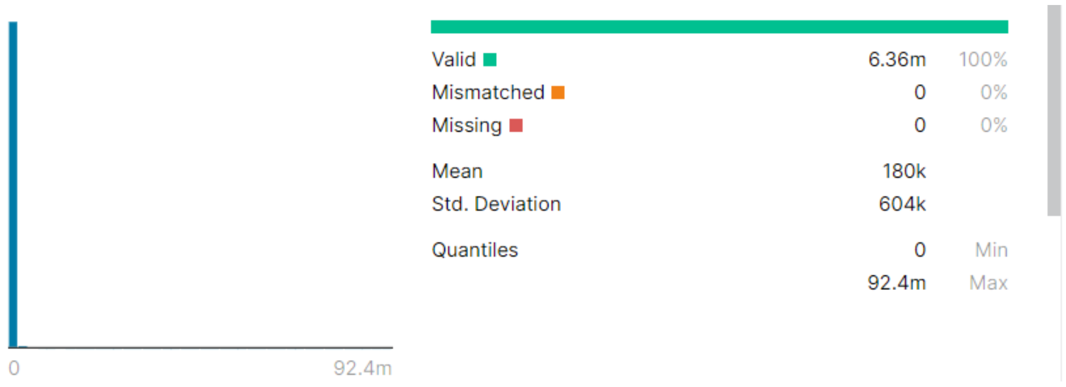
CASH_OUT	35%
PAYMENT	34%
Other (1973625)	31%

Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	5	
Most Common	CASH_OUT	35%

Fig 2: Type attribute statistics

3. Amount:

Indicates the amount of money being transacted.

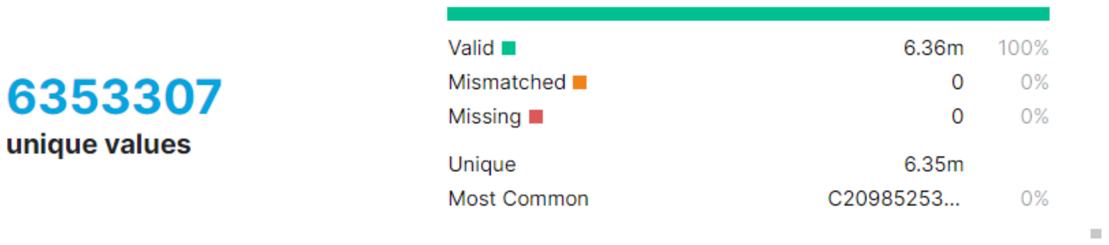


Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	180k	
Std. Deviation	604k	
Quantiles	0	Min
	92.4m	Max

Fig 3: Amount attribute statistics

4. nameOrg:

It indicates the name of the sender from where the transaction is being initiated.



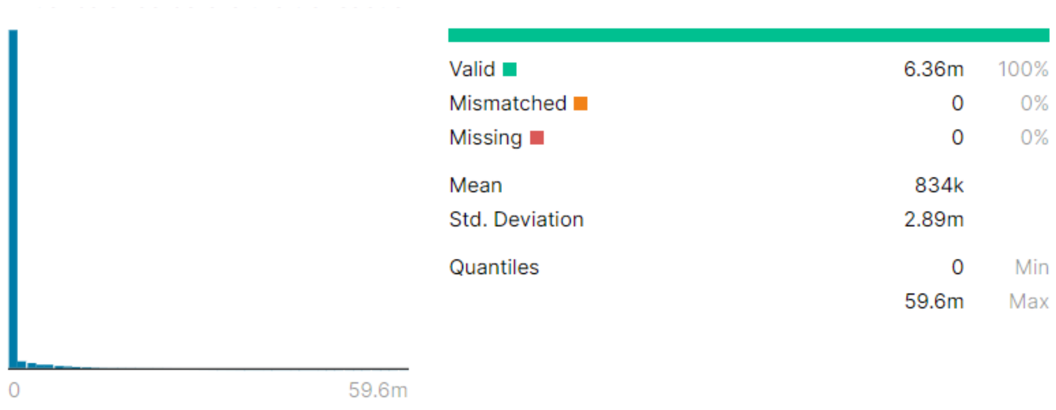
6353307
unique values

Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	6.35m	
Most Common	C20985253...	0%

Fig 4: nameOrg attribute statictics

5. oldbalanceOrg:

It indicates the balance of the sender of the amount before transaction.

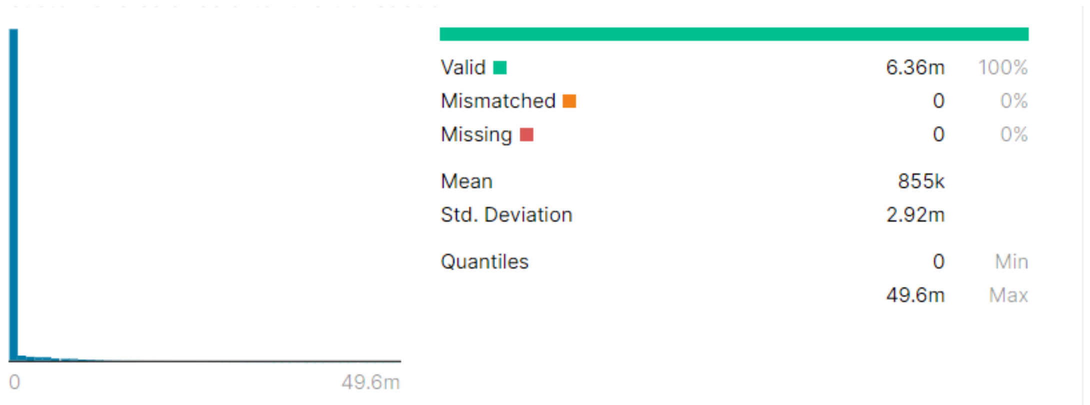


Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	834k	
Std. Deviation	2.89m	
Quantiles	0	Min
	59.6m	Max

Fig 5: oldbalanceOrg attribute statistics

6. newbalanceOrg:

It indicates the new balance if the sender after performing the transaction.

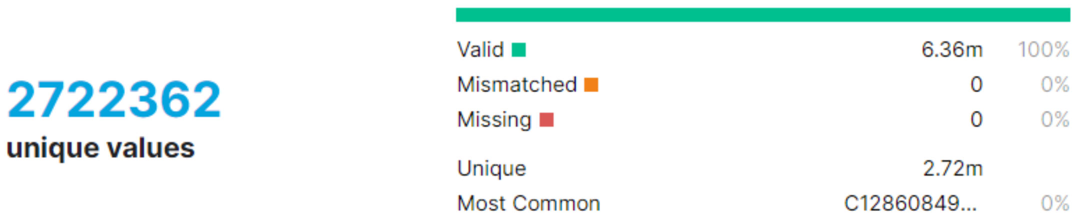


Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	855k	
Std. Deviation	2.92m	
Quantiles	0	Min
	49.6m	Max

Fig 6: newbalanceOrg attribute statistics

7. nameDest:

It indicates the name of the receiver of the amount that is being transferred.

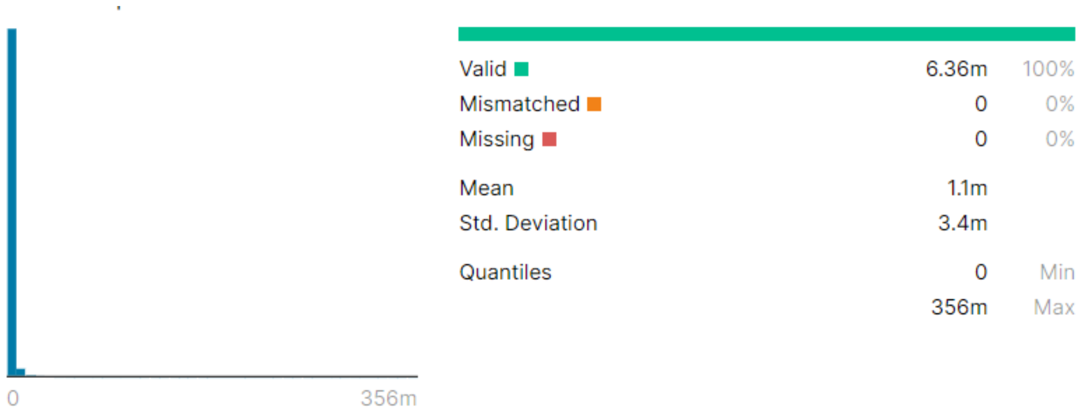


2722362
unique values

Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Unique	2.72m	
Most Common	C12860849...	0%

Fig 7: nameDest attribute statistics

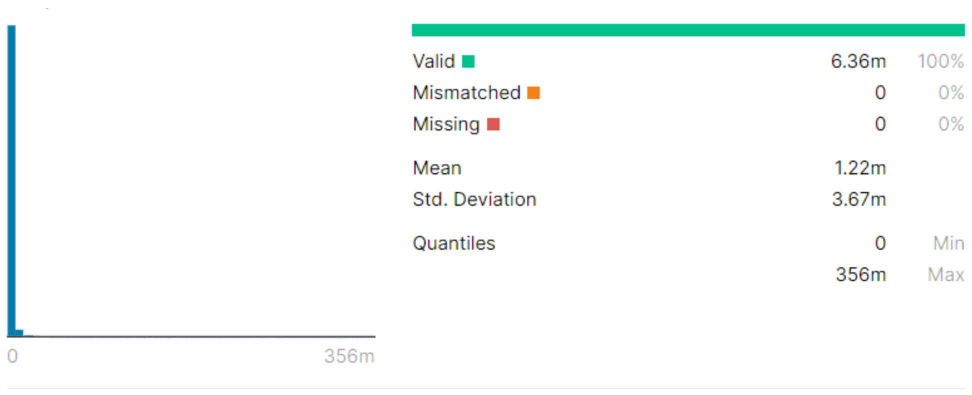
8. **oldbalanceDest:**
It indicates the old balance of the receiver before receiving the amount that is transferred.



Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	1.1m	
Std. Deviation	3.4m	
Quantiles	0	Min
	356m	Max

Fig 8: oldbalanceDest attribute statistics

9. **newbalanceDest:**
It indicates the new balance of the receiver after receiving the amount being transferred.

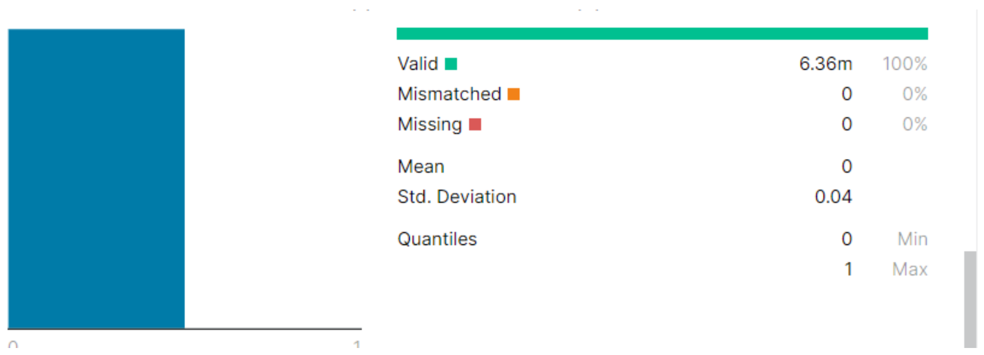


Valid 🟩	6.36m	100%
Mismatched 🟧	0	0%
Missing 🟥	0	0%
Mean	1.22m	
Std. Deviation	3.67m	
Quantiles	0	Min
	356m	Max

Fig 9: newbalanceDest attribute statistics

10. isFraud:

It indicates whether a transaction is fraudulent or not. If the outcome is 1 it means that the transaction is fraudulent and if the outcome is 0 is means that the transaction is not fraudulent.



Valid 🟩 6.36m 100%
Mismatched 🟧 0 0%
Missing 🟥 0 0%

Mean 0
Std. Deviation 0.04

Quantiles 0 Min
1 Max

Fig 10: isFraud attribute statistics

11. isFlaggedFraud:

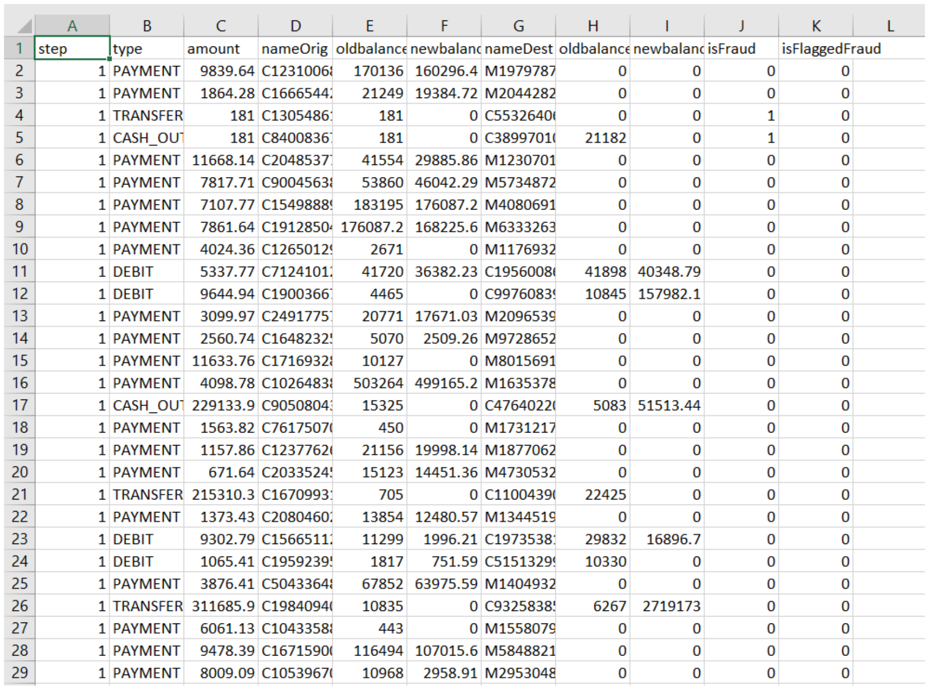
It acts as an indicator that there is a potential possibility for a transaction to be a fraudulent one if it holds a value of 1 and otherwise if it holds 0.

flags illegal attempts to transfer more than 200.000 in a single transaction.

Fig 11: isFlaggedFraud Attribute statistics

Training the dataset using following records

The dataset contains 11 attributes and 6353307 records. Among the 11 attributes **"isFraud"** is the class attribute.



	step	type	amount	nameOrig	oldbalance	newbalanc	nameDest	oldbalance	newbalanc	isFraud	isFlaggedFraud
2	1	PAYMENT	9839.64	C1231006	170136	160296.4	M1979787	0	0	0	0
3	1	PAYMENT	1864.28	C1666544	21249	19384.72	M2044282	0	0	0	0
4	1	TRANSFER	181	C1305486	181	0	C5532640	0	0	1	0
5	1	CASH_OUT	181	C8400836	181	0	C3899701	21182	0	1	0
6	1	PAYMENT	11668.14	C2048537	41554	29885.86	M1230701	0	0	0	0
7	1	PAYMENT	7817.71	C9004563	53860	46042.29	M5734872	0	0	0	0
8	1	PAYMENT	7107.77	C1549888	183195	176087.2	M4080691	0	0	0	0
9	1	PAYMENT	7861.64	C1912850	176087.2	168225.6	M6333263	0	0	0	0
10	1	PAYMENT	4024.36	C1265012	2671	0	M1176932	0	0	0	0
11	1	DEBIT	5337.77	C7124101	41720	36382.23	C1956008	41898	40348.79	0	0
12	1	DEBIT	9644.94	C1900366	4465	0	C9976083	10845	157982.1	0	0
13	1	PAYMENT	3099.97	C2491775	20771	17671.03	M2096539	0	0	0	0
14	1	PAYMENT	2560.74	C1648232	5070	2509.26	M9728652	0	0	0	0
15	1	PAYMENT	11633.76	C1716932	10127	0	M8015691	0	0	0	0
16	1	PAYMENT	4098.78	C1026483	503264	499165.2	M1635378	0	0	0	0
17	1	CASH_OUT	229133.9	C9050804	15325	0	C4764022	5083	51513.44	0	0
18	1	PAYMENT	1563.82	C7617507	450	0	M1731217	0	0	0	0
19	1	PAYMENT	1157.86	C1237762	21156	19998.14	M1877062	0	0	0	0
20	1	PAYMENT	671.64	C2033524	15123	14451.36	M4730532	0	0	0	0
21	1	TRANSFER	215310.3	C1670993	705	0	C1100439	22425	0	0	0
22	1	PAYMENT	1373.43	C2080460	13854	12480.57	M1344519	0	0	0	0
23	1	DEBIT	9302.79	C1566511	11299	1996.21	C1973538	29832	16896.7	0	0
24	1	DEBIT	1065.41	C1959239	1817	751.59	C5151329	10330	0	0	0
25	1	PAYMENT	3876.41	C5043364	67852	63975.59	M1404932	0	0	0	0
26	1	TRANSFER	311685.9	C1984094	10835	0	C9325838	6267	2719173	0	0
27	1	PAYMENT	6061.13	C1043358	443	0	M1558079	0	0	0	0
28	1	PAYMENT	9478.39	C1671590	116494	107015.6	M5848821	0	0	0	0
29	1	PAYMENT	8009.09	C1053967	10968	2958.91	M2953048	0	0	0	0

Fig 12: Dataset Sample

1.2.2 Existing Models:

1. **Clustering:**

Clustering is a process that classifies the data into different groups, and the members of each group have the most similarities to each other and the members of each group have the least similarities to another group. The best performance of a clustering algorithm will be apparent when the clusters are away from each other as far as possible. In anti-money laundering, clustering is typically used for grouping transactions with bank accounts in different clusters that have the most similarities with each other. These techniques help us to detect patterns for suspicious transaction sequence or present models to identify the accounts or the riskier customers. One of the most important challenges facing the clustering of financial transactions is the size and the amount of data, for example, we are facing thousands or millions of transactions per unit time in this method. Table 1 shows different clustering methods used.

Source	Methods	Title	The main objective	Technology / algorithms / methods
(Zhu 2006)	Clustering	An outlier detection model based on cross dataset comparison for financial surveillance	The money laundering detection using comparison every customer transaction with the same customer's transaction history	Creating a profile for customers
(Zengan 2009)	Clustering	Application of cluster-based local outlier factor algorithm in anti-money laundering	The money laundering detection using clustering techniques combination and Outliers	Cluster-based local outlier factor algorithm
(Wang and Dong 2009)	Clustering	Research on money laundering detection based on improved minimum spanning tree clustering and its application	Clustering method based on improved minimum spanning tree is created based on the criterion of dissimilarity, which the minimum spanning tree is built based on these criteria, and this tree is clustered into k clusters.	Minimum spanning tree
(Le Khac and Kechadi 2010)	Clustering	Application of data mining for anti-money laundering detection: A case study	Providing a solution based on the knowledge that detects the patterns of money laundering by combining data mining techniques and natural computing.	Neural networks Genetic algorithm Heuristics

Table 1: Classification of money laundering detection methods using clustering

2. Rule-based methods:

We can observe two approaches in data mining, classification - prediction and clustering approach. Rule-based methods are considered classification and prediction methods. In rule-based methods, we are facing a set of rules that are expressed in the language of logic, and actually, we use a series of logical rules to classify the factors.

Source	Methods	Title	The main objective	Technology / algorithms / methods
(Khan et al. 2013)	Prediction	A Bayesian approach for suspicious financial activity reporting	It creates a model of the user's past activities, and this model will be a gauge of future customer activities. If the transaction or customer financial activities have significant deviations to the pattern, the new transaction of the user will be the suspicious money laundering activities	Bayesian networks
Rajput et al.(2014)	Classification and prediction	Ontology Based Expert-System for Suspicious Transactions Detection	In this study, the provided approach to detect suspicious activities is founded by monitoring independent transactions.	Ontology / Semantic Web
Khanuja and Adane 2014	Classification	Forensic Analysis for Monitoring Database Transactions	Providing a methodology for continuous monitoring of the monetary and financial systems by preset instructions	Dempster – Shafer Theory

Table 2: Rule-Based Methods in Money Laundering Detection

3. Support vector machines:

SVM is a supervised learning method, which is used for classification. Support vector machines similar to neural networks can obtain approximations with the desired degree of accuracy for each multivariable function. So, it is very useful to model the nonlinear and complex systems and processes, including detecting activities related to financial fraud. SVM goal is to find a separator super-vector of data points belonging to two classes, with a maximal margin. From a the geometric perspective, it is the gap between the super-vector and the nearest training samples. From another perspective, the margin is defined as the amount of space or separation between the two classes, which is defined by the super -vector.

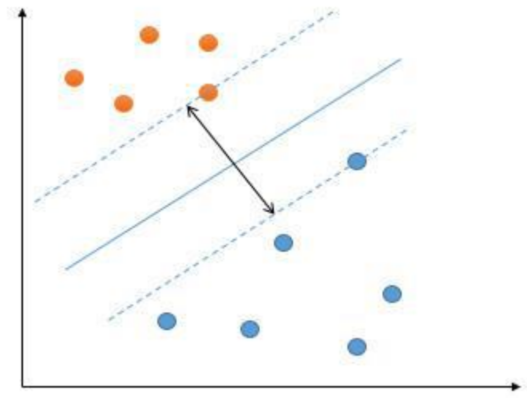


Fig 13: Sample of Support Vector Machine

Source	Methods	Title	The primary objective	Technology / algorithms / methods
(Tang and Yin 2005)	Classification	Developing an intelligent data discriminating system of anti-money laundering based on SVM	Using statistical learning theory to improve the anti money laundering..	Statistical learning theory Support Vector Machine
(Keyan and Tingting 2011)	Classification	An improved support-vector network model for anti-money laundering	Selecting a classification parameter suitable to detect suspicious activities using support vector machines is vital.	Support Vector Machine Cross Validation

Table 3: Classification of Methods Based on Support Vector Machines for Money Laundering Detection

4. **Social networks**:

In recent years, the Social Network theory has attracted increasing attention. Social network analysis regarding data mining is called link analysis or link mining. For the modelling of social networks, the relationship between entities will be displayed in the form of links in a graph.

A social network represents relationships between social entities such as friends, professionals or writers. In the last decade, due to the increasing growth of communication technologies and Web-based services, the growth and penetration of these networks have been widespread, and many people had the experience and interacted with social networks. Through online social networks, a huge amount of facilities will be provided for interaction and cooperation between independent individuals, regardless of the geographical distance between them. Each network is a massive database of millions of users and their activities. Unfortunately, due to the liberalization of the use of most social networks, the information contained on these networks is a good place to delinquent users. Therefore, network analysis will be instrumental to explore relationships or suspicious transactions for money laundering detection. Social networks are dynamic. New links show a new inter action between objects.In the prediction of links, a snapshot of the social network at the time “t” will be placed at our disposal, and we are asked to predict the edges that will be added to this network, in the period t to t + 1. In this case, we are looking forward to use the real attributes of the model and to expose the development that can model the evolution of a social network.

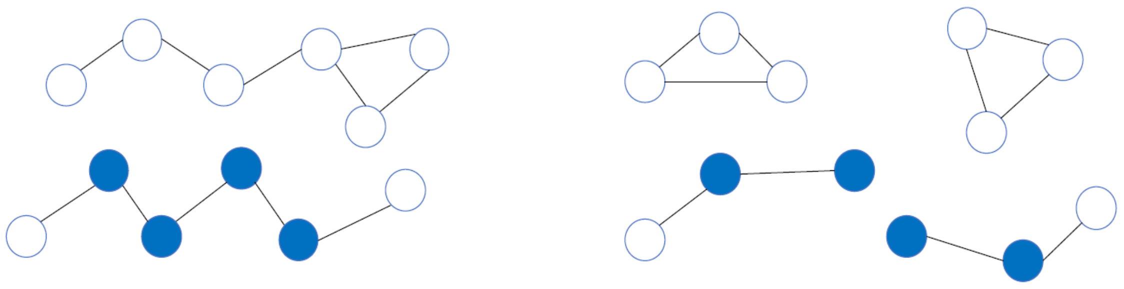


Fig 14: Networks with Different Relationships

Source	Methods	Title	The primary objective	Technolog y / algorithms / methods
(Dreżewski, Sepielak, and Filipkowski 2015)	Social networks analysis	The application of social network analysis algorithms in a system supporting money laundering detection	Presenting a new application for money laundering detection in a great deal of banking data. In this study, researchers have added a social network analysis component to their old system	Money Launderin g Detection System (MLDS)
(Colladon and Remondi 2017)	Social networks analysis	Using social network analysis to prevent money laundering	Introducing a new approach to sorting and mapping relationships between data and presenting a prediction model based on social networks criteria to assess the risk of profiles that are involved in the business. The social network analysis is used to predict the involvement of accounts for customers who are involved in processes of money laundering.	Decision support systems

Table 4: The Methods of Money Laundering Detection Using Social Network Analysis

1.2.3 Proposed Models

Logistic Regression Model:

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

The dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no). Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Types of Logistic Regression

a. **Binary or Binomial:** In such a kind of classification, a dependent variable will have only two possible types either 1 and 0

b. **Multinomial:** In such a kind of classification, dependent variable can have 3 or more possible unordered types or the types having no quantitative significance.

c. **Ordinal:** In such a kind of classification, dependent variable can have 3 or more possible ordered types or the types having a quantitative significance.

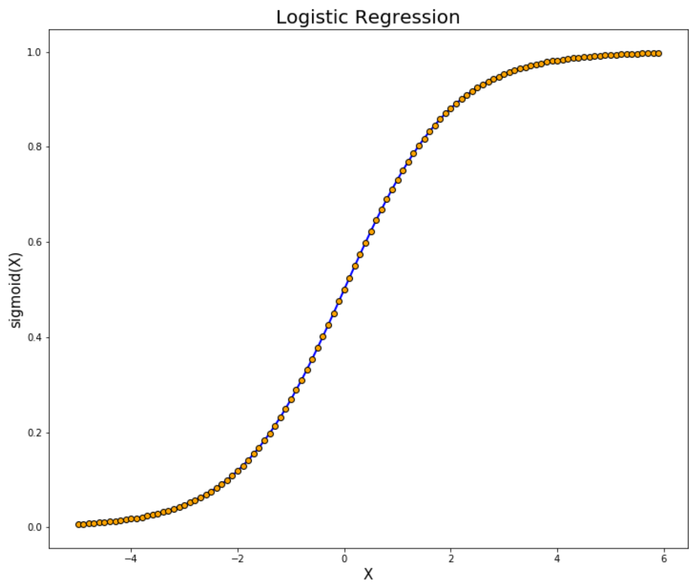


Fig 15: Sample of Logistic Regression Model

Decision Tree Model:

The structure of a decision tree is a tree topology similar to a flowchart. The highest node in the tree is the root node, and the leaf nodes represent categories and distribution of categories. The decision tree is a classification or prediction technique that each non-leaf test node test specifies a feature and every branch out from this node shows the result of this test. Unlike neural networks, decision trees deal with the production rules. The prediction obtained from a tree is explained in the form of a series rule in a decision tree, a while the result of prediction is only expressed in neural networks and how to achieve them is hidden in the network itself. Also, unlike neural networks, there is no requirement for the data to be necessarily numerical in the decision tree. Decision forest or random forest is a collection of several decision trees that avoids instability and excessive risk education (bias) that may occur in a single tree.

Split Creation

1. **Part one: Calculating Gini Score:** We have just discussed this part in the previous section.

2. **Part two: Splitting a dataset:** It may be defined as separating a dataset into two lists of rows having index of an attribute and a split value of that attribute. After getting the two groups - right and left, from the dataset, we can calculate the value of split by using Gini score calculated in first part. Split value will decide in which group the attribute will reside.

3. **Part three: Evaluating all Splits:** Next part after finding Gini score and splitting dataset is the evaluation of all splits.

For this purpose, first, we must check every value associated with each attribute as a candidate split. Then we need to find the best possible split by evaluating the cost of the split. The best split will be used as a node in the decision tree.

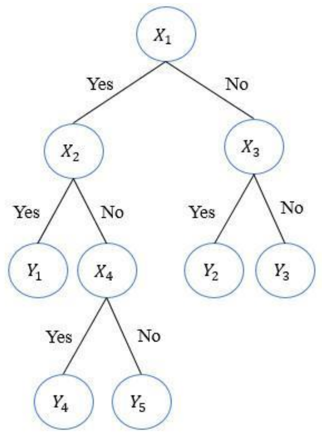


Fig 16: Sample of Decision Tree Model

Random Forest Model:

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

Working of Random Forest Model:

We can understand the working of Random Forest algorithm with the help of following steps –

Step-1: First, start with the selection of random samples from a given dataset.

Step-2: Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.

Step-3: In this step, voting will be performed for every predicted result.

Step-4: At last, select the most voted prediction result as the final prediction result.

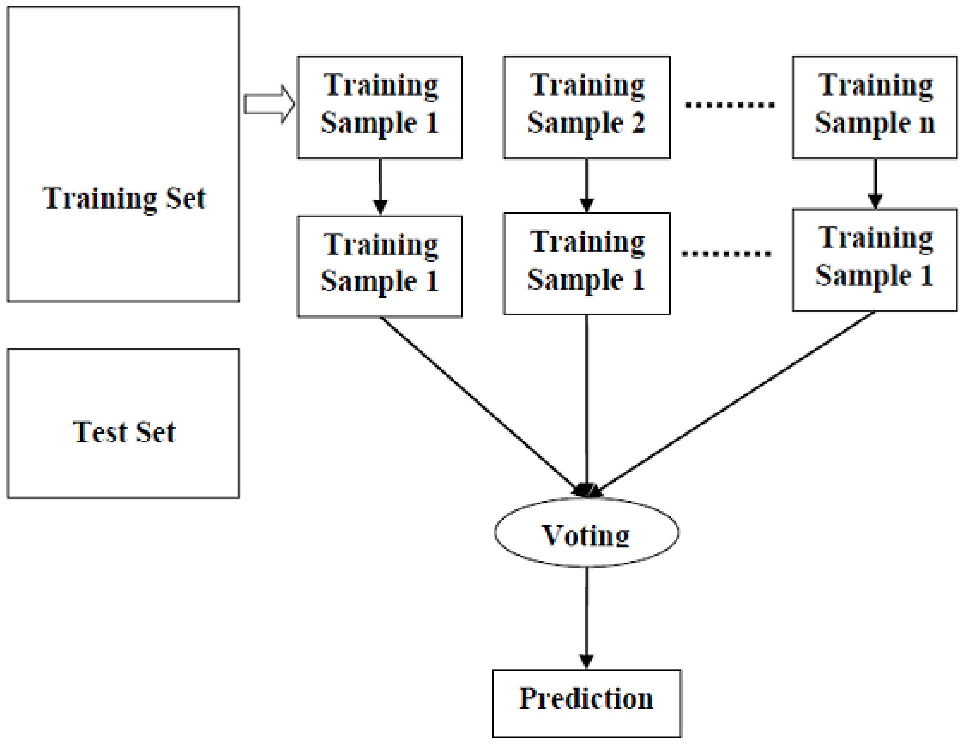


Fig 17: Sample of Random Forest Model

Convolutional Neural Network Model:

CNN architectures vary with the type of the problem at hand. The proposed model consists of three convolutional layers each followed by a maxpooling layer. The final layer is fully connected MLP. ReLu activation function is applied to the output of every convolutional layer and fully connected layer. The first convolutional layer filters the input image with 32 kernels of size 3x3. After max pooling is applied, the output is given as an input for the second convolutional layer with 64 kernels of size 4x4. The last convolutional layer has 128 kernels of size 1x1 followed by a fully connected layer of 512 neurons. The output of this layer is given to softmax function which produces a probability distribution of the four output class. The model is trained using adaptive moment estimation (Adam) with batch size of 100 for 1000 epochs.

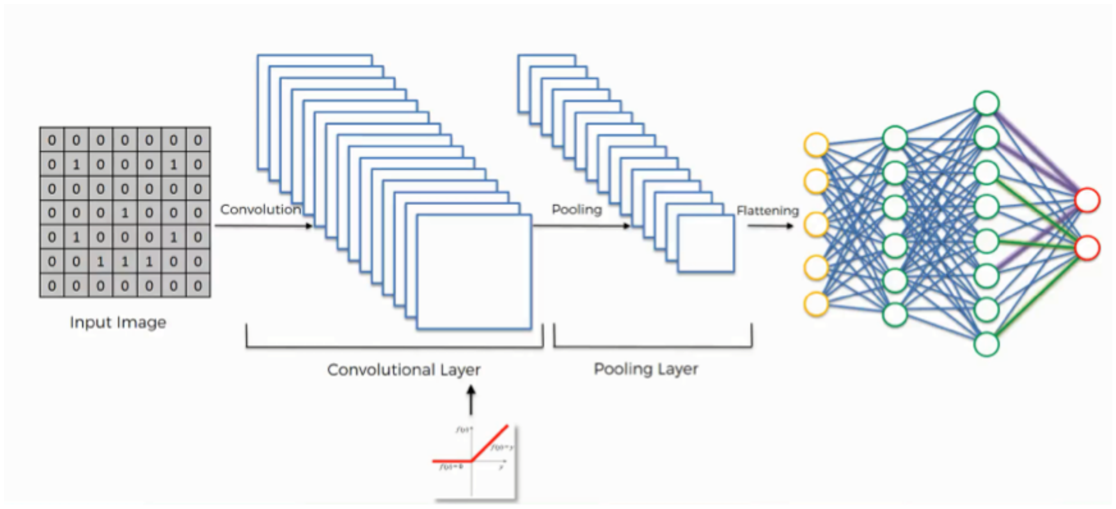


Fig 18: Convolutional Neural Network

Steps in CNN:

There are four steps in CNN model-

1. Convolution:

The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information. In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel to then produce a feature map.

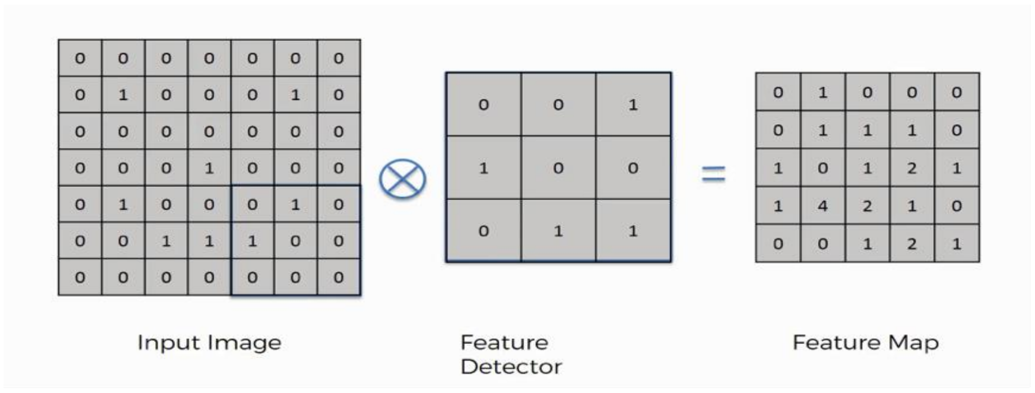


Fig 19: Convolution in CNN Model

2. Max Pooling:

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned

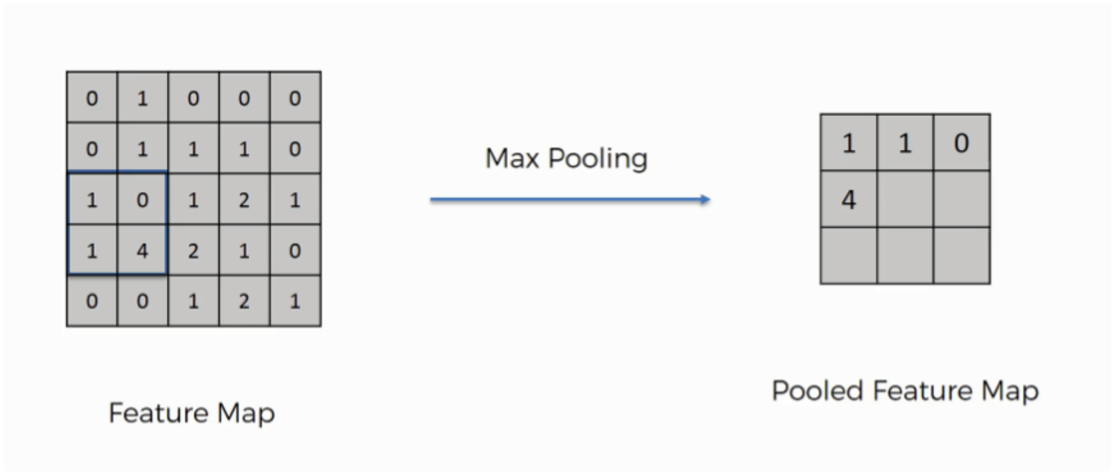


Fig 20: Max Pooling in CNN Model

3. Flattening:

Flattening is the process of converting all the resultant 2 dimensional arrays into a single long continuous linear vector

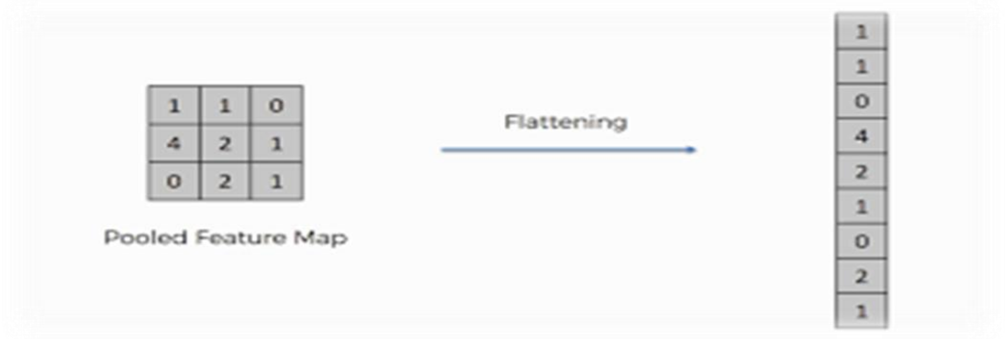


Fig 21: Flattening in CNN Model

4. Full Connection

At the end of a CNN, the output of the last Pooling Layer acts as a input to the so called Fully Connected Layer.

There can be one or more of these layers ("fully connected" means that every node in the first layer is connected to every node in the second layer).

As you see from the image below, we have three layers in the full connection step:

- Input layer
- Fully-connected layer
- Output layer

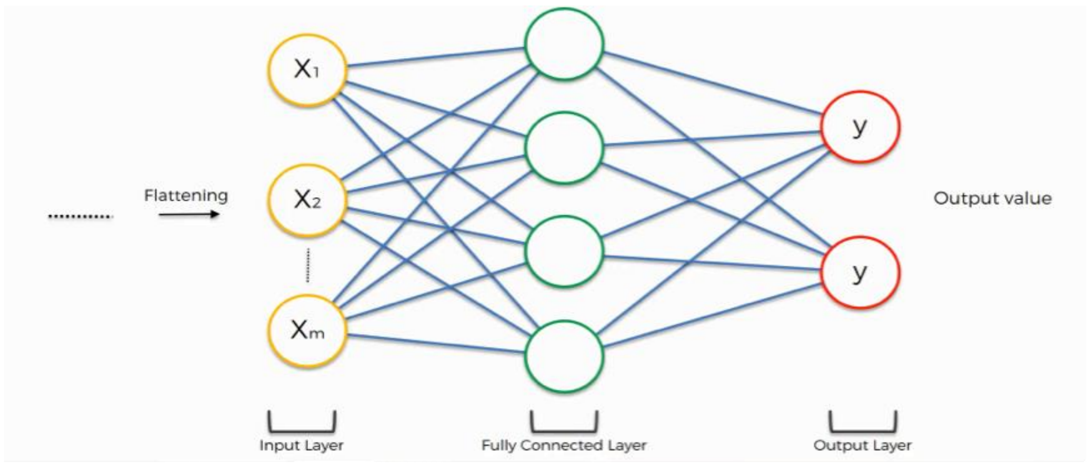


Fig 22: Full Connection in CNN Model

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirements Gathering

2.1.1 Software Requirements

Programming Language	Python 3.6
Graphical User Interface	Tkinter
Dataset	Money Laundering Detection dataset
Packages	Tensorflow, Numpy, Pandas, Matplotlib, Scikit-learn, keras, Seaborm, imblearn
Framework	Anaconda
Tool	Jupyter Notebook

2.1.2 Hardware Requirements

Operating System	Windows 10
Processor	Intel Core i3-2348M
CPU Speed	2.30 GHz
Memory	4 GB (RAM)

2.2 Technologies Description

2.2.1 Python:

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors.

This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python

2.2.2 Tkinter:

Tkinter is the most commonly used library for developing GUI (Graphical User Interface) in Python. It is a standard Python interface to the Tk GUI toolkit shipped with Python. As Tk and Tkinter are available on most of the Unix platforms as well as on the Windows system, developing GUI applications with Tkinter becomes the fastest and easiest. Graphical User Interface(GUI) is a form of user interface which allows users to interact with computers through visual indicators using items such as icons, menus, windows, etc. It has advantages over the Command Line Interface(CLI) where users interact with computers by writing commands using keyboard only and whose usage is more difficult than GUI. Tkinter is the inbuilt python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python as it is simple and easy to work with. You don't need to worry about the installation of the Tkinter module separately as it comes with Python already. It gives an object-oriented interface to the Tk GUI toolkit.
Widgets in Tkinter are the elements of GUI application which provides various controls (such as Labels, Buttons, ComboBoxes, CheckBoxes, MenuBars, RadioButtons and many more) to users to interact with the application.

2.2.3 Seaborn:

Seaborn is a visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas. Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variables for better understanding of dataset. Plots are basically used for visualizing the relationship between variables. Those variables can be either be completely numerical or a category like a group, class or division. Seaborn divides plot into the below categories:

- Relational plots: This plot is used to understand the relation between two variables.
- Categorical plots: This plot deals with categorical variables and how they can be visualized.
- Distribution plots: This plot is used for examining univariate and bivariate distributions
- Regression plots: The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analyses.
- Matrix plots: A matrix plot is an array of scatterplots.
- Multi-plot grids: It is an useful approach is to draw multiple instances of the same plot on different subsets of the dataset.

2.2.4 Tensorflow:

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

TensorFlow is basically a software library for numerical computation using data flow graphs where:

- nodes in the graph represent mathematical operations.
- edges in the graph represent the multidimensional data arrays (called tensors) communicated between them. (Please note that tensor is the central unit of data in TensorFlow).

2.2.5 Numpy:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.
It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

2.2.6 Pandas:

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

2.2.7 Malplotlib:

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible.

You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery. For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython.

For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

2.2.8 Scikit – Learn:

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis
- Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

2.2.9 Imblearn:

imb-Learn is a Python module that helps in balancing the datasets which are highly skewed or biased towards some classes. Thus, it helps in resampling the classes which are otherwise oversampled or under sampled. If there is a greater imbalance ratio, the output is biased to the class which has a higher number of examples. The following dependencies need to be installed to use imbalanced-learn:

- scipy(>=0.19.1)
- numpy(>=1.13.3)
- scikit-learn(>=0.23)
- joblib(>=0.11)
- keras 2 (optional)
- tensorflow (optional)

The resampling of data is done in 2 parts:

Estimator: It implements a fit method which is derived from scikit-learn. The data and targets are both in the form of a 2D array

Re-sampler: The fit resample method resample the data and targets into a dictionary with a key-value pair of data resampled and targets resampled.

The Imbalanced Learn module has different algorithms for oversampling and under sampling:
We will use the built-in dataset called the make classification dataset which return

- x: a matrix of n_samples * n_features and
- y: an array of integer labels.

2.2.10 Anaconda:

Anaconda is an open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing. Anaconda works for R and python programming language. Spyder(sub-application of Anaconda) is used for python. Opencv for python will work in spyder. Package versions are managed by the package management system called conda. To begin working with Anaconda, one must get it installed first. Follow the below instructions to Download and install Anaconda on your system:

2.2.11 Jupyter Notebook:

The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text.
Jupyter Notebook is maintained by the people at Project Jupyter.
Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

3.DESIGN

3.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

3.2 Flow Diagram

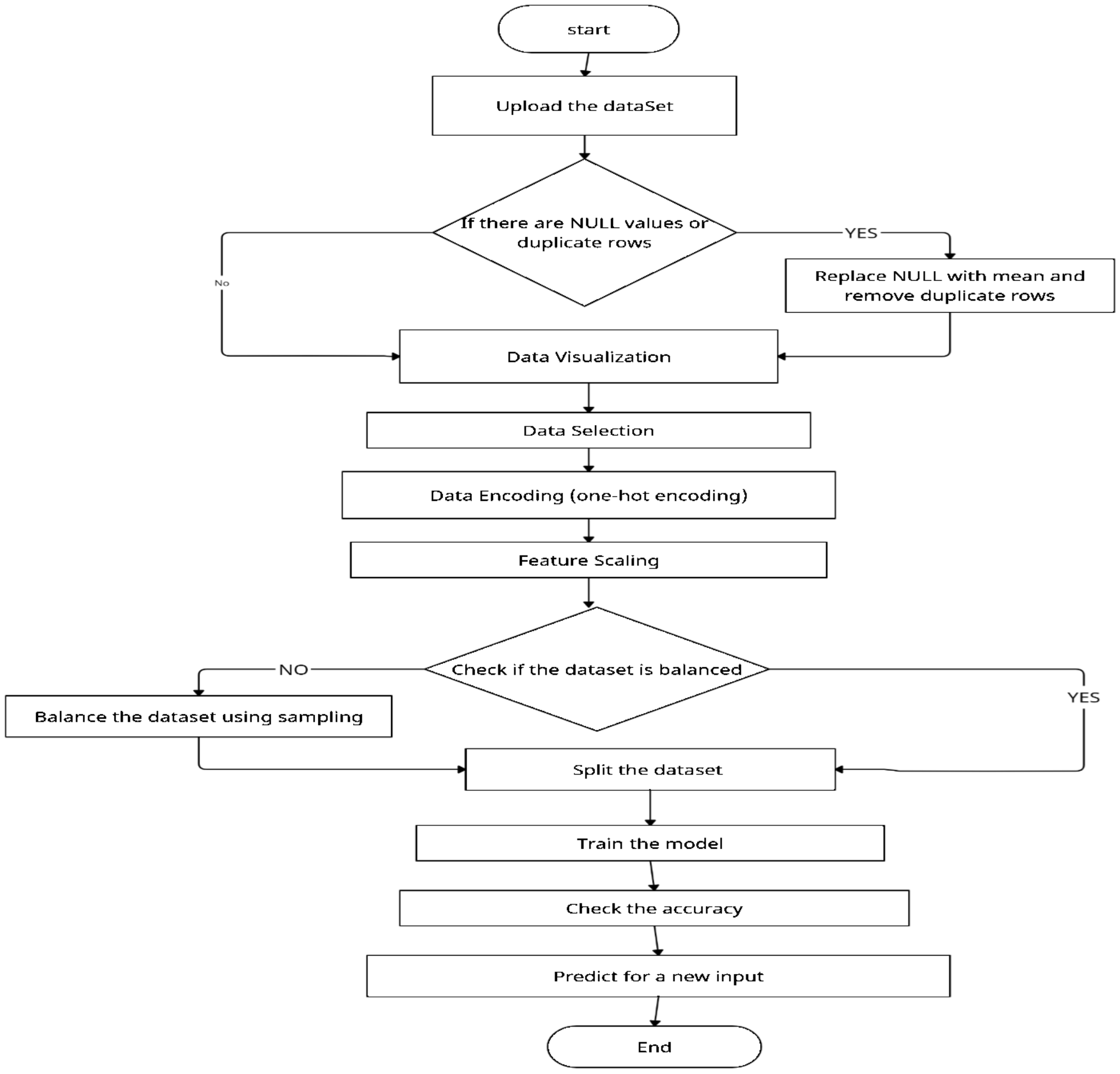

start
Upload the dataSet
If there are NULL values or duplicate rows
YES
No
Replace NULL with mean and remove duplicate rows
Data Visualization
Data Selection
Data Encoding (one-hot encoding)
Feature Scaling
Check if the dataset is balanced
NO
YES
Balance the dataset using sampling
Split the dataset
Train the model
Check the accuracy
Predict for a new input
End

Fig 23: Flow Diagram for Money Laundering Detection

3.3 System Architecture

Web applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

Like desktop applications, web applications are made up of many parts and often contain mini programs and some of which have user interfaces. In addition, web applications frequently require an additional markup or scripting language, such as HTML, CSS, or JavaScript programming language. Also, many applications use only the Python programming language, which is ideal because of its versatility

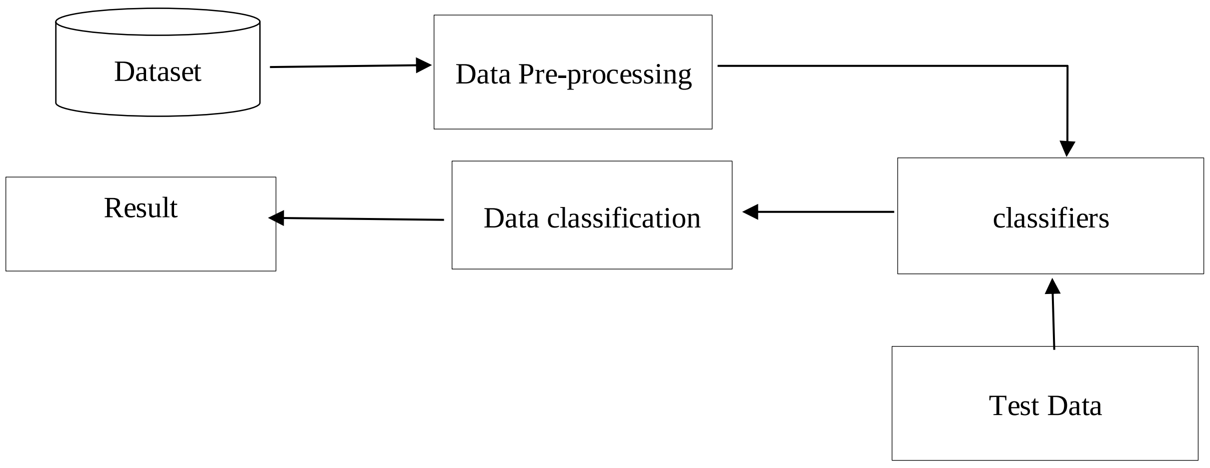


Fig 24: System Architecture for money laundering detection

3.4 Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

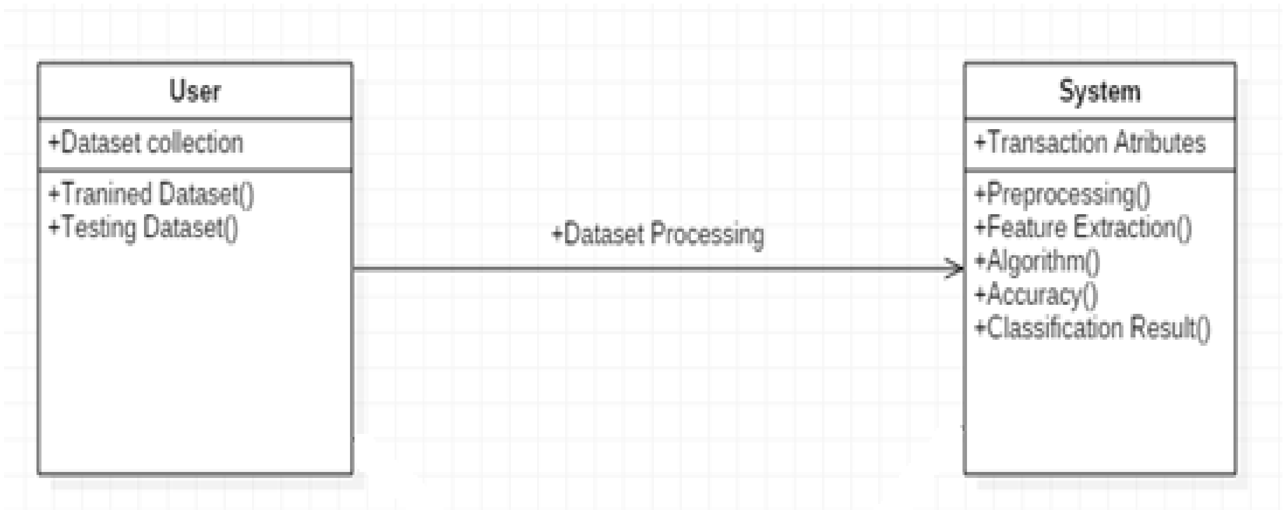


Fig 25: Class Diagram for Money Laundering detection

3.5 Activity Diagram

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control

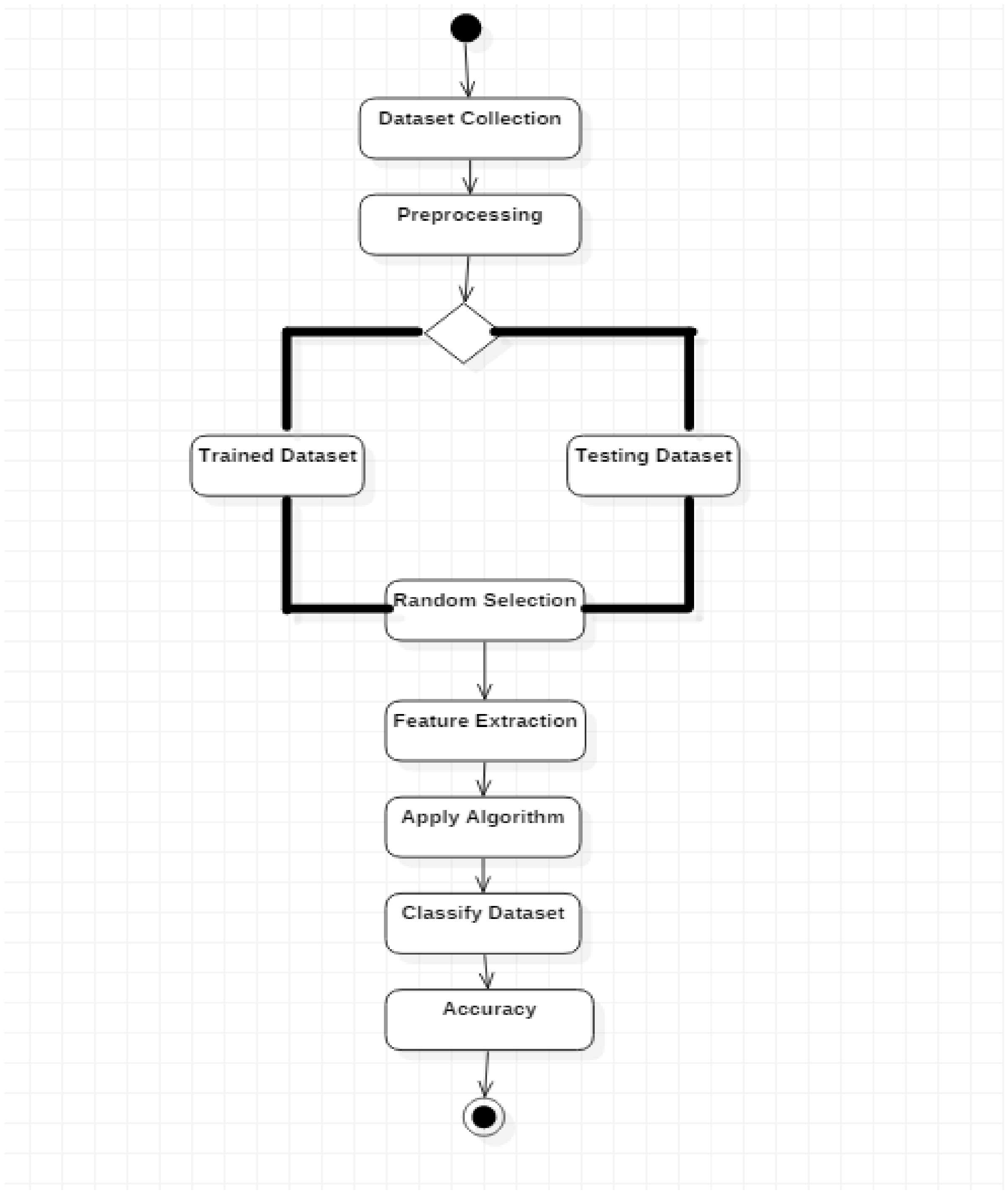


Fig 26: Activity Diagram for Money Laundering Detection

3.6 Sequence Diagram

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

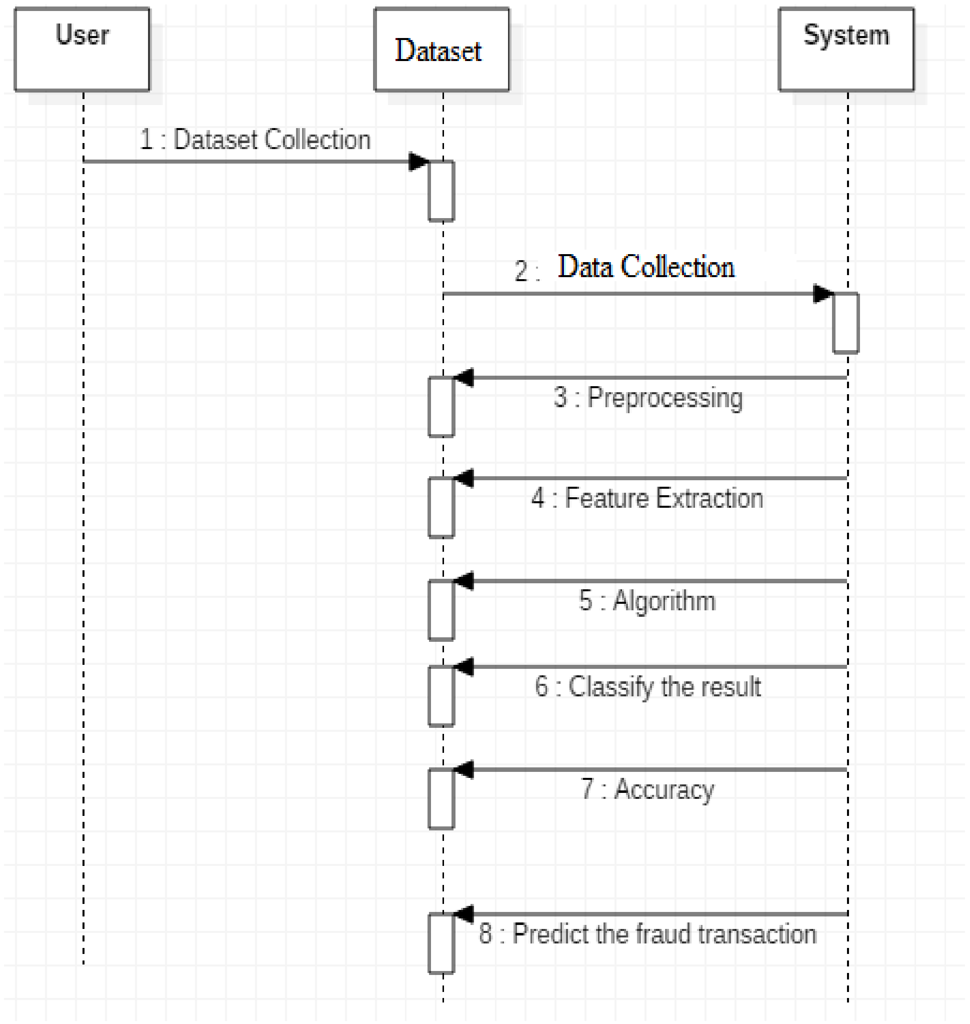


Fig 27: Sequence Diagram for Money Laundering detection

3.7 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.
Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.
These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.
Hence to model the entire system, a number of use case diagrams are used.

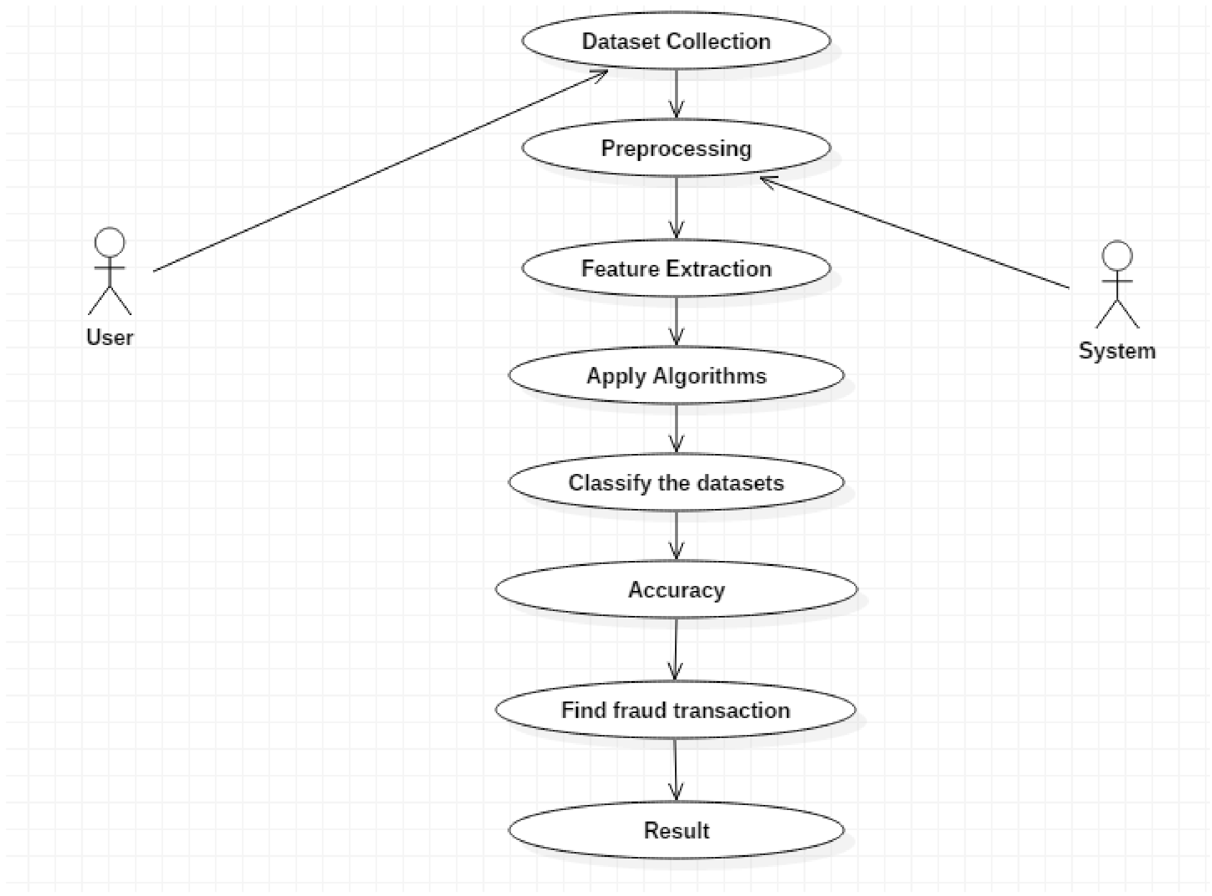


Fig 28: Use Case Diagram for Money Laundering Detection

3.8 Collaboration Diagram

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

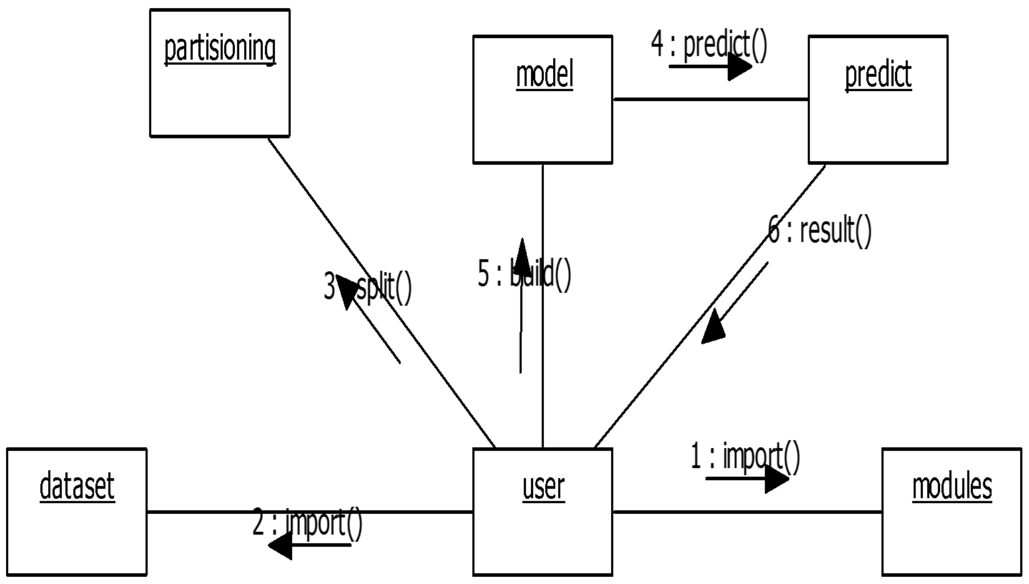


Fig 29: Collaboration Diagram for Money Laundering Detection

4.IMPLEMENTATION OF THE PROPOSED PROJECT

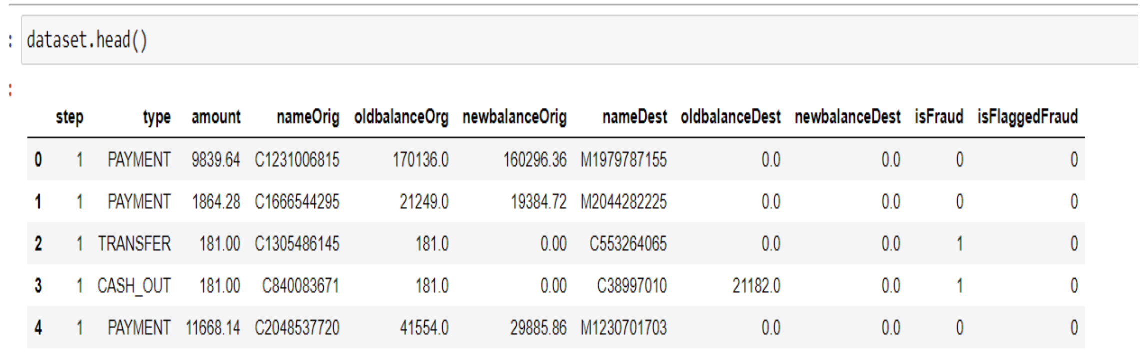
4.1 Source Code

Importing Required Modules

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from keras.models import Sequential,model_from_json
from keras.layers import Dense
from keras.optimizers import RMSprop
import tkinter
from sklearn.metrics import accuracy_score
import warnings
import tensorflow as tf
warnings.filterwarnings("ignore")
%matplotlib inline
Installing Keras
```

Uploading and understanding the dataset

```
dataset =  pd.read_csv("PS_20174392719_1491204439457_log-1.csv")
dataset.head()
dataset.info()
dataset.describe()
```



```
: dataset.head()
:
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Fig 30: Sample of the dataset

Data Cleaning

```
print('Null Values =',dataset.isnull().values.any())
dataset = dataset.drop_duplicates(keep='first').copy()
dataset.shape
```

Data Visualization

```
f, ax = plt.subplots(figsize=(12, 10))
plt.title('correlation of transaction features')

sns.heatmap(dataset.corr(),linewidths=0.25,vmax=1.0,    square=True,    cmap="YlGnBu",
linecolor='black', annot=True)
dataset.corr()['isFraud'].sort_values(ascending=False)
```

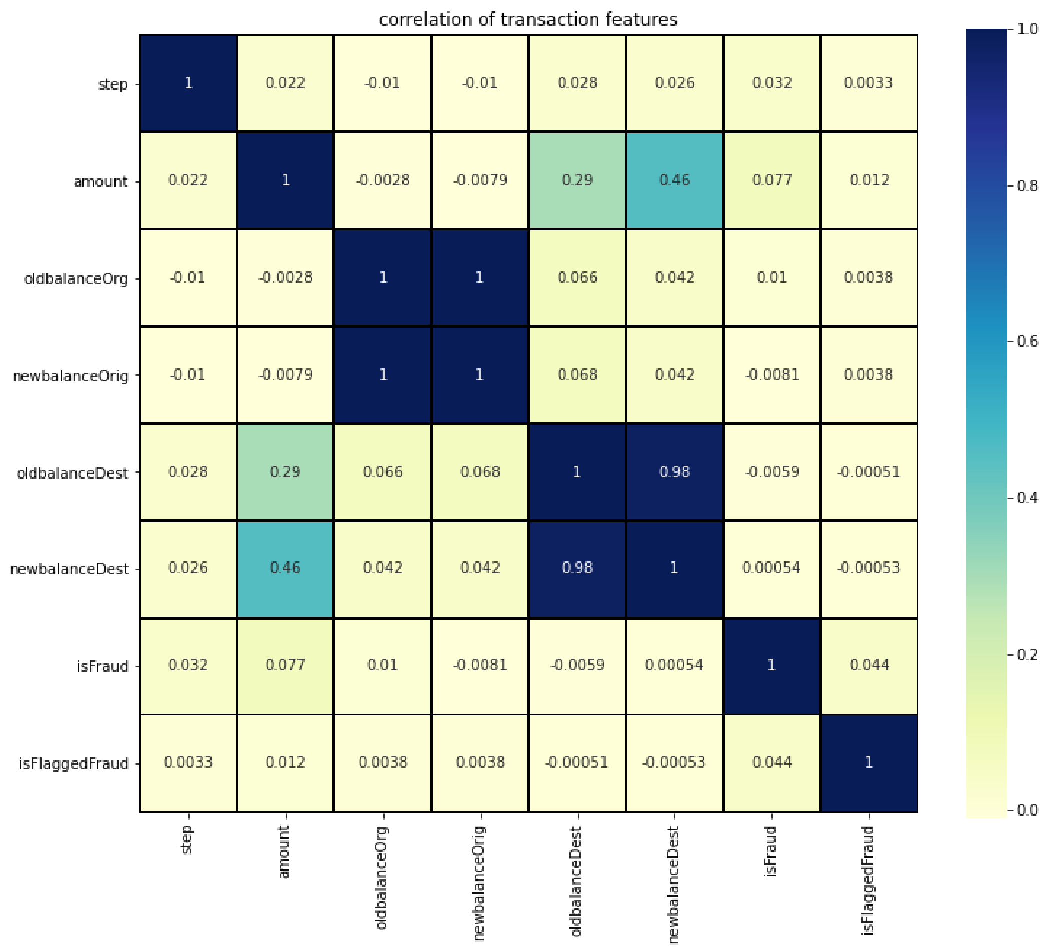


Fig 31: Heapmap

```
dataset.plot.scatter(x='type', y='isFraud')
```

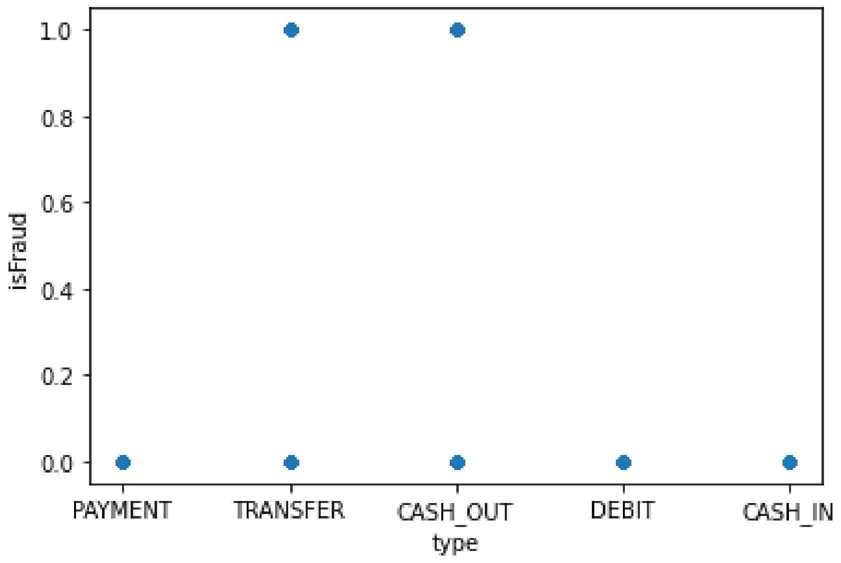
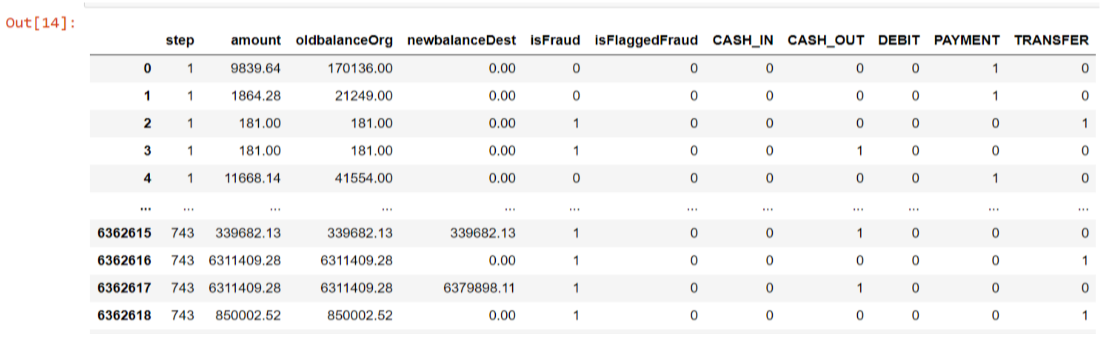


Fig 32: Scatter Plot

Feature Selection and Encoding

```
dataset.drop('nameOrig', axis=1, inplace=True)
dataset.drop('nameDest', axis=1, inplace=True)
dataset.drop('oldbalanceDest', axis=1, inplace=True)
dataset.drop('newbalanceOrig', axis=1, inplace=True)
dum = pd.get_dummies(dataset['type'])
dataset_encoded = pd.concat([dataset,dum],axis=1)
dataset_encoded.drop(['type'],axis=1, inplace=True)
dataset_encoded
```



Out[14]:

	step	amount	oldbalanceOrg	newbalanceDest	isFraud	isFlaggedFraud	CASH_IN	CASH_OUT	DEBIT	PAYMENT	TRANSFER
0	1	9839.64	170136.00	0.00	0	0	0	0	0	1	0
1	1	1864.28	21249.00	0.00	0	0	0	0	0	1	0
2	1	181.00	181.00	0.00	1	0	0	0	0	0	1
3	1	181.00	181.00	0.00	1	0	0	1	0	0	0
4	1	11668.14	41554.00	0.00	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
6362615	743	339682.13	339682.13	339682.13	1	0	0	1	0	0	0
6362616	743	6311409.28	6311409.28	0.00	1	0	0	0	0	0	1
6362617	743	6311409.28	6311409.28	6379898.11	1	0	0	1	0	0	0
6362618	743	850002.52	850002.52	0.00	1	0	0	0	0	0	1

Fig 33: Encoded dataset

Splitting data

```
dataset_balanced = dataset_encoded.sample(n=20000)
dataset_balanced.isFraud.value_counts().plot.bar()
print(dataset_balanced.isFraud.value_counts())
```

```
x=dataset_balanced.drop(['isFraud'],axis=1)
y=dataset_balanced['isFraud']
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size=0.3, random_state=100)
```

Sampling the training dataset

```
print('Before OverSampling, the shape of train_X: {}'.format(X_train.shape))
print('Before OverSampling, the shape of train_y: {} \n'.format(y_train.shape))
print("Before OverSampling, counts of label '1': {}".format(sum(y_train==1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train==0)))
sm = SMOTE(random_state=10, sampling_strategy= 1.0)
x_train_res, y_train_res = sm.fit_resample(X_train, y_train)
print('After OverSampling, the shape of train_X: {}'.format(x_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))
print("After OverSampling, counts of label '1': {}".format(sum(y_train_res==1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res==0)))
```

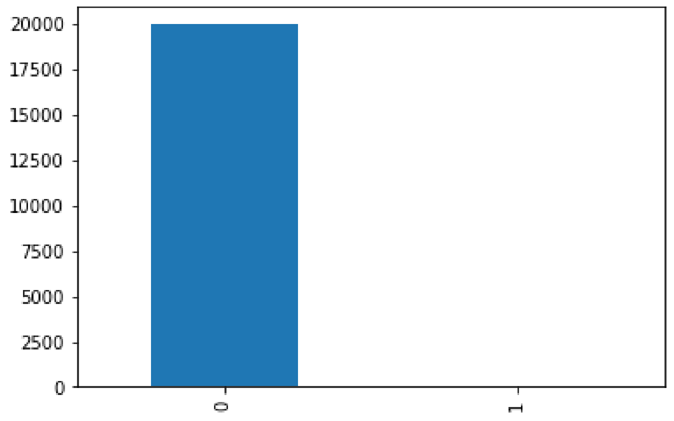


Fig 34: Graphical Representation of Imbalanced Target class

Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train_scaled = sc.fit_transform(x_train_res)
x_test_scaled = sc.transform(X_test)
```

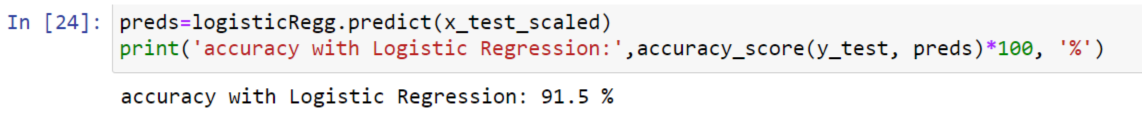
Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logisticRegg=LogisticRegression()
logisticRegg.fit(x_train_scaled, y_train_res)

preds=logisticRegg.predict(x_test_scaled)
print('accuracy with Logistic Regression:',accuracy_score(y_test, preds)*100, '%')
```

import sklearn.metrics as metrics
preds = logisticRegg.predict(x_test_scaled)
rmsle=metrics.mean_squared_log_error(y_test, preds)
#rmsle = np.sqrt(mean_squared_log_error(y_test, preds))
rmsle

print('Confusion matrix', '\n', confusion_matrix(y_test, preds), '\n')
print('Classification report', '\n', classification_report(y_test, preds), '\n')



```
In [24]: preds=logisticRegg.predict(x_test_scaled)
         print('accuracy with Logistic Regression:',accuracy_score(y_test, preds)*100, '%')

         accuracy with Logistic Regression: 91.5 %
```

Fig 35: Accuracy for Logistic Regression

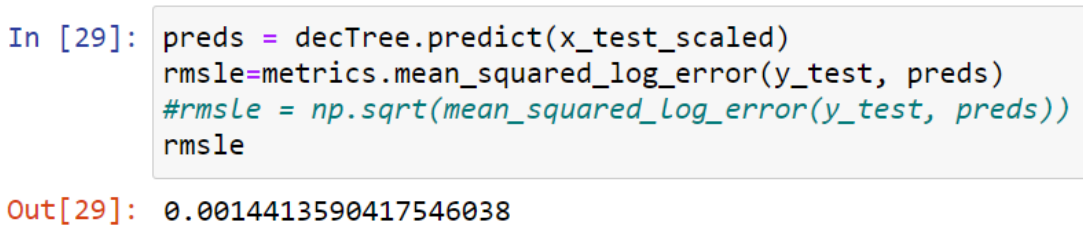
Decision Tree

from sklearn.tree import DecisionTreeClassifier
decTree=DecisionTreeClassifier()
decTree.fit(x_train_scaled, y_train_res)

preds=decTree.predict(x_test_scaled)
print('accuracy with Decision Tree:',accuracy_score(y_test, preds)*100, '%')

preds = decTree.predict(x_test_scaled)
rmsle=metrics.mean_squared_log_error(y_test, preds)
#rmsle = np.sqrt(mean_squared_log_error(y_test, preds))
rmsle

print('Confusion matrix', '\n', confusion_matrix(y_test, preds), '\n')
print('Classification report', '\n', classification_report(y_test, preds), '\n')



```
In [29]: preds = decTree.predict(x_test_scaled)
         rmsle=metrics.mean_squared_log_error(y_test, preds)
         #rmsle = np.sqrt(mean_squared_log_error(y_test, preds))
         rmsle

Out[29]: 0.0014413590417546038
```

Fig 36: Accuracy for Decision Tree

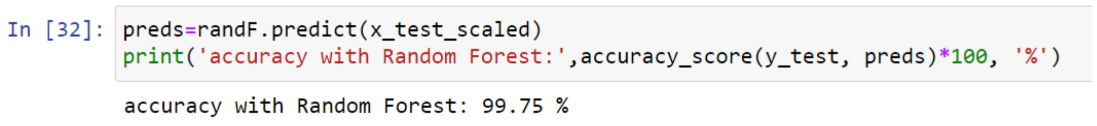
Random Forest

from sklearn.ensemble import RandomForestClassifier
randF=RandomForestClassifier()
randF.fit(x_train_scaled, y_train_res)

preds=randF.predict(x_test_scaled)

print('accuracy with Random Forest:',accuracy_score(y_test, preds)*100, '%')
preds = randF.predict(x_test_scaled)
rmsle=metrics.mean_squared_log_error(y_test, preds)
#rmsle = np.sqrt(mean_squared_log_error(y_test, preds))
rmsle

print('Confusion matrix', '\n', confusion_matrix(y_test, preds), '\n')
print('Classification report', '\n', classification_report(y_test, preds), '\n')



```
In [32]: preds=randF.predict(x_test_scaled)
         print('accuracy with Random Forest:',accuracy_score(y_test, preds)*100, '%')

         accuracy with Random Forest: 99.75 %
```

Fig 37: Accuracy for Random Forest

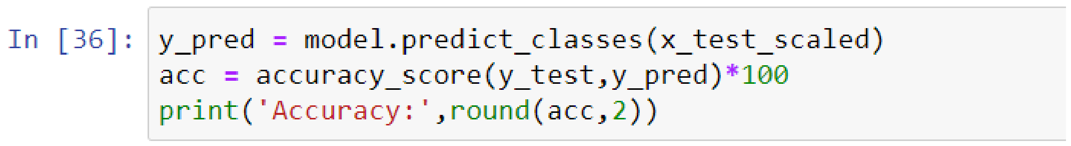
Neural Networks

''' Initializing the model '''
model = Sequential()
''' Adding the input layer and the first hidden layer '''
model.add(Dense(6, kernel_initializer = 'uniform', activation = 'relu', input_shape=(None,10)))
''' Adding the second hidden layer '''
model.add(Dense(6, kernel_initializer = 'uniform', activation = 'relu'))
''' Adding the output layer '''
model.add(Dense(1, kernel_initializer = 'uniform', activation = 'sigmoid'))
''' Compiling and fitting the model '''
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
model_info = model.fit(x_train_scaled, y_train_res, batch_size = 10, epochs = 10)

y_pred = model.predict_classes(x_test_scaled)
acc = accuracy_score(y_test,y_pred)*100
print('Accuracy:',round(acc,2))

rmsle=metrics.mean_squared_log_error(y_test, y_pred)
#rmsle = np.sqrt(mean_squared_log_error(y_test, preds))
rmsle

print('Confusion matrix', '\n', confusion_matrix(y_test, y_pred), '\n')
print('Classification report', '\n', classification_report(y_test, y_pred), '\n')

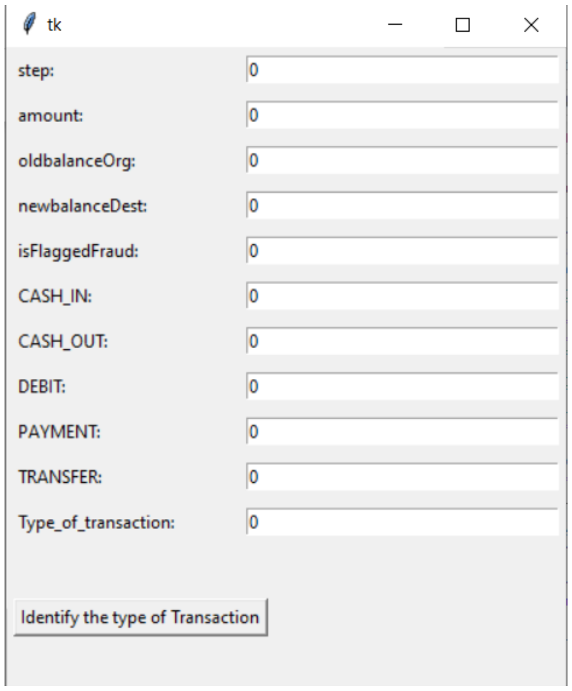


```
In [36]: y_pred = model.predict_classes(x_test_scaled)
         acc = accuracy_score(y_test,y_pred)*100
         print('Accuracy:',round(acc,2))
```

Fig 38: Accuracy for Neural Network

Graphical User Interface

```
from tkinter import *
fields                                                                                                                       =
('step','amount','oldbalanceOrg','newbalanceDest','isFlaggedFraud','CASH_IN','CASH_OUT','D
EBIT','PAYMENT','TRANSFER','Type_of_transaction')
def final_balance(entries):
    st = (float(entries['step'].get()) / 100) / 12
    amnt = float(entries['amount'].get())
    oldbal = float(entries['oldbalanceOrg'].get())
    newbal = float(entries['newbalanceDest'].get())
    flag = float(entries['isFlaggedFraud'].get())
    cin = float(entries['CASH_IN'].get())
    cout = float(entries['CASH_OUT'].get())
    dbt = float(entries['DEBIT'].get())
    paymnt = float(entries['PAYMENT'].get())
    trans = float(entries['TRANSFER'].get())
    x_new=np.reshape(np.asarray([st,amnt,oldbal,newbal,flag,cin,cout,dbt,paymnt,trans]),(1,10))
    y_pred = model.predict_classes(x_new)
    print(y_pred)
    entries['Type_of_transaction'].delete(0,END)
    if y_pred==1:
        entries['Type_of_transaction'].insert(0,"The transaction done is Fraudlent!!")
    else:
        entries['Type_of_transaction'].insert(0,"No Money Laundering!!!!")
def makeform(root, fields):
    entries = {}
    for field in fields:
        row = Frame(root)
        lab = Label(row, width=22, text=field+": ", anchor='w')
        ent = Entry(row)
        ent.insert(0,"0")
        row.pack(side = TOP, fill = X, padx = 5 , pady = 5)
        lab.pack(side = LEFT)
        ent.pack(side = RIGHT, expand = YES, fill = X)
        entries[field] = ent
    return entries
root = Tk()
ents = makeform(root, fields)
root.bind('<Return>', (lambda event, e = ents: fetch(e)))
b1 = Button(root, text = 'Identify the type of Transaction',command=(lambda e = ents:
final_balance(e)))
b1.pack(side = LEFT, padx = 5, pady = 5)
root.mainloop()
```



step: 0
amount: 0
oldbalanceOrg: 0
newbalanceDest: 0
isFlaggedFraud: 0
CASH_IN: 0
CASH_OUT: 0
DEBIT: 0
PAYMENT: 0
TRANSFER: 0
Type_of_transaction: 0

Fig 39: Sample GUI

4.2 Testing

Testing is a set of activities that can be planned in advanced and conducted systematically. A strategy for software testing must accommodation low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements. Software testing is one element of verification and validation. Verification refers to the set of activities that ensure that software correctly implements as specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements. The main objective of software is testing to uncover errors. To fulfill this objective, a series of test steps unit, integration, validation and system tests are planned and executed. Each test step is accomplished through a series of systematic test technique that assist in the design of test cases. With each testing step, the level of abstraction with which software is considered is broadened.

Unit Testing

This testing method considers a module as single unit and checks the unit at interfaces and communicates with other modules rather than getting into details at statement level.

System testing

Here all the pre tested individual modules will be assembled to create the larger system and tests are carried out at system level to make sure that all modules are working in synchronous with each other.

Integration Testing

Testing is a major quality control measure employed during software development. Its basic function is to detect errors. Sub functions when combined may not produce than it is desired. Global data structures can represent the problems. Integrated testing is a systematic technique for constructing the program structure while conducting the tests

Regression testing

Each time a new module is added as a part of integration as the software changes. Regression testing is an actually that helps to ensure changes that do not introduce unintended behavior as additional errors.

4.3 Test Cases

Testcase ID	Test Scanario	Expected Result
TC01	Check whether all the required fields are given input	compare number of inputs with expected inputs
TC02	Check whether the inputs are matching with datatypes	Check the input type with expected type
TC03	Check whether find the type of transaction buton is working	redirected to a function on clicking the button

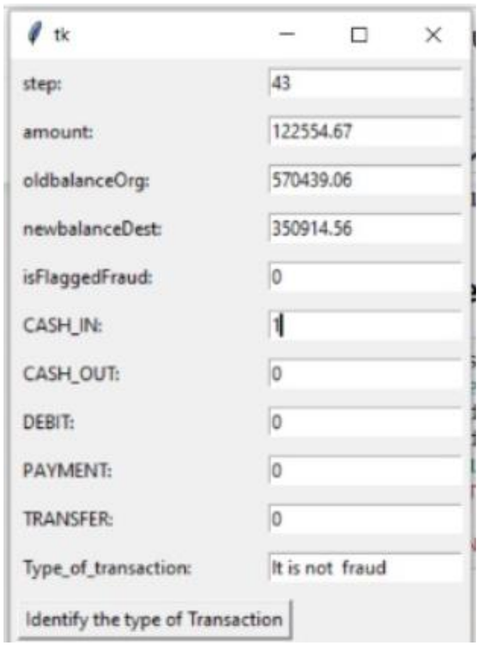
Table 5: Test cases

4.4 Execution Screen Shots

Input 1:

Step	43
Type	CASH_IN
Amount	122554.67
oldbalanceOrg	570439.06
newbalanceOrg	350914.56
isFlaggedFraud	0

Output:



tk

step:	43
amount:	122554.67
oldbalanceOrg:	570439.06
newbalanceDest:	350914.56
isFlaggedFraud:	0
CASH_IN:	1
CASH_OUT:	0
DEBIT:	0
PAYMENT:	0
TRANSFER:	0
Type_of_transaction:	It is not fraud

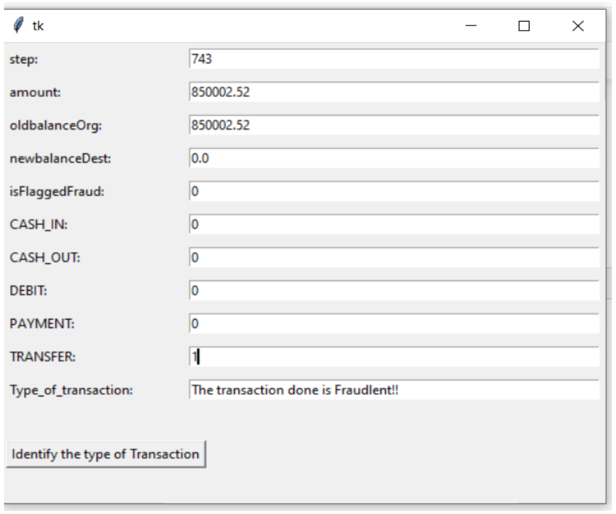
Identify the type of Transaction

Fig 40: Out put for Input 1

Input 2:

Step	743
Type	TRANSFER
Amount	850002.52
oldbalanceOrg	850002.52
newbalanceDest	0.0
isFlaggedFraud	0

Output



tk

step:	743
amount:	850002.52
oldbalanceOrg:	850002.52
newbalanceDest:	0.0
isFlaggedFraud:	0
CASH_IN:	0
CASH_OUT:	0
DEBIT:	0
PAYMENT:	0
TRANSFER:	1
Type_of_transaction:	The transaction done is Fraudlent!!

Identify the type of Transaction

Fig 41: Out put for input 2

CONCLUSION AND FUTURE SCOPE

Data mining is a process to extract knowledge from existing data. It is used as a tool in banking and finance, in general, to discover useful information from the operational and historical data to enable better decision-making. It is an interdisciplinary field, the confluence of Statistics, Database technology, Information science, Machine learning, and Visualization. It involves steps that include data selection, data integration, data transformation, data mining, pattern evaluation, knowledge presentation. Banks use data mining in various application areas like marketing, fraud detection, risk management, money laundering detection and investment banking. According to what was mentioned in the previous parts, detecting activities related to money laundering is necessary and inevitable for the economy, industries, banks and financial institutions.

The future scope for this project is – This activity of indentifying suspecious activity can be extended to be implemented in electronic payment applications to detect fraudulant transactions immediately.

REFERENCES

- https://www.ripublication.com/ijaer17/ijaerv12n20_120.pdf
- http://www.ijarset.com/upload/2015/august/1_IJARSET_manjunath.pdf
- https://www.researchgate.net/publication/43508814_Knowledge-based_anti-money_laundering_A_software_agent_bank_application