Fraud Detection and Prevention

Department of Electronics and Communication

Rajiv Gandhi University of Knowledge Technologies

Nuzvid, India-521202

July 2020

Report submitted by

P. Jyothi(N150463)

V. Sushma(N150303)

K. Sri lahari(N150063)

Certificate

It is certified that the work contained in this term project entitled "**Fraud Detection and Prevention**" in MACHINE LEARNING using by "P. Jyothi, V. Sushma, K. Srilahari" has been carried out under the guidance of "P. Shravani madam'" and that it has not been submitted elsewhere for a degree.

Project Guide

P. Shravani,

Asst. Professor,

RGUKT NUZVID.

July 2020

ACKNOWLEDGEMENT

The success in this project would not have been possible but for the timely help and guidance rendered by many people. We wish to express our sincere thanks to all those who has assisted us in one way or the other for the completion of the project.

We thank to the project guide P. Shravani, Assistant Professor Department of Electronics and Communication Engineering for guiding all through the project works, giving a right direction and shape to learning by extending his expertise and experience in the education. Really, we are indebted for her excellent and enlightened guidance.

We thank to all the teaching, non-teaching staff of the Department of ECE who gave all possible help to bring project work to the present shape.

We thank all who contributed directly or indirectly in successfully carrying out the work.

P. Jyothi(N150463)

V. Sushma(N150303)

K. Srilahari(N150063)

ABSTRACT

Name of the students	:P. Jyothi, V. Sushma, K. Srilahari
ID No	: N150463 ,N150303 ,N150063
Degree for which submitted	: B. Tech
Department	: ECE
Term project title	: Fraud Detection And Prevention
Project guide	: P.Shravani
Month and year of thesis submission	: july 2020

It is vital that credit card companies are able to identify fraudulent credit card transactions so that customers are not charged for items that they did not purchase. Such problems can be tackled with Data Science and its importance, along with Machine Learning, cannot be overstated. This project intends to illustrate the modelling of a data set using machine learning with Credit Card Fraud Detection.

The Credit Card Fraud Detection Problem includes modelling past credit card transactions with the data of the ones that turned out to be fraud. This model is then used to recognize whether a new transaction is fraudulent or not. Our objective here is to detect 100% of the fraudulent transactions while minimizing the incorrect fraud classifications.

Credit Card Fraud Detection is a typical sample of classification. In this process, we have focused on analysing and pre-processing data sets as well as the deployment of multiple anomaly detection algorithms such as Local Outlier Factor and Isolation Forest algorithm on the PCA transformed Credit Card Transaction.

CONTENTS

CHAPTERS

1 Introduction

2 Methodology

2.1 Architectures

2.2 Proposed System

3 System Analysis

3.1 System Requirement Specification

3.1.1 Functional Requirements

3.1.2 Non-Functional Requirements

3.1.3 External Interface Requirements

3.1.4 System Requirements

3.2 Feasibility Study

3.2.1 Technical Feasibility

3.2.2 Operational Feasibility 3.2.3 Economical Feasibility

4 System Design

4.1 Introduction

4.2 System Modules

4.3 Data Flow Diagrams

4.4 UML Diagrams

4.4.1 Use-Case Diagram

4.4.2 Sequence Diagram
4.4.3 Activity Diagram
4.4.4 State Chart Diagram
5 System Implementation
5.1 Sample Code
6 Results
7 System Testing
7.1 Introduction
7.2 Testing Strategies
7.2.1 Unit Testing
7.2.2 Validation Testing
7.2.3 System Testing
7.2.4 Output Testing
8 CONCLUSION
9 FUTURESCOPE
10 REFERENCES

1.INTRODUCTION

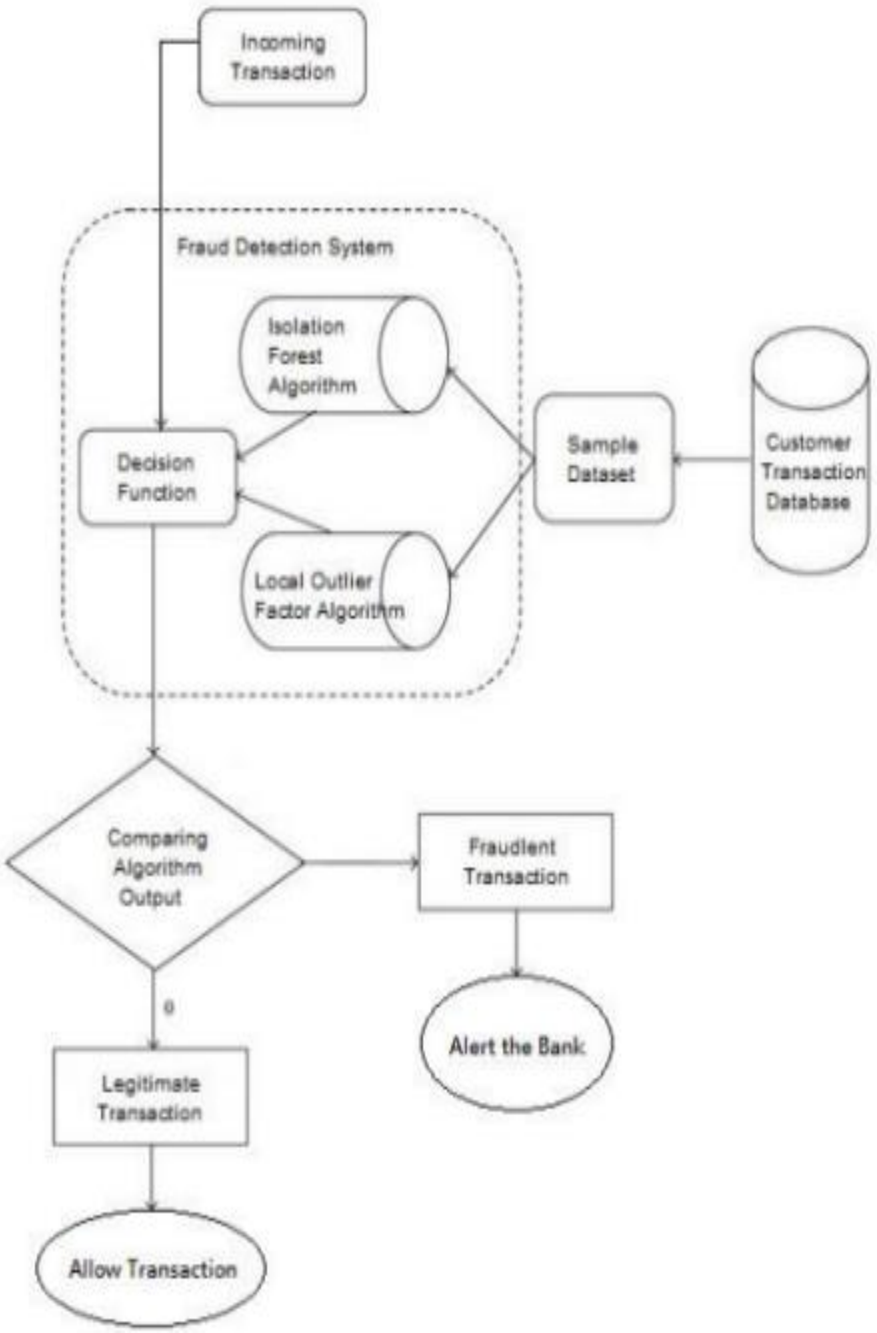
'Fraud' in credit card transactions is unauthorized and unwanted usage of an account by someone other than the owner of that account. Necessary prevention measures can be taken to stop this abuse and the behaviour of such fraudulent practices can be studied to minimize it and protect against similar occurrences in the future. In other words, Credit Card Fraud can be defined as a case where a person uses someone else’s credit card for personal reasons while the owner and the card issuing authorities are unaware of the fact that the card is being used.

This is very relevant problem that demands the attention of communities such as machine learning and data science where the solution to this problem can be automated. This problem is particularly challenging from the perspective of learning, as it is characterized by various factors such as class imbalance. The number of valid transactions far outnumber fraudulent ones. Also, the transaction patterns often change their statistical properties over the course of valid time.

These are not the only challenges in the implementation of a real-world fraud detection system, however. In real world examples, the massive stream of payment requests is quickly scanned by automatic tools that determine which transactions to authorize.

Fraud detection methods are continuously developed to defend criminals in adapting to their fraudulent strategies. These frauds are classified as:

- Credit Card Frauds: Online and Offline
- Card Theft
- Account Bankruptcy
- Device Intrusion
- Application Fraud
- Counterfeit Card
- Telecommunication Fraud

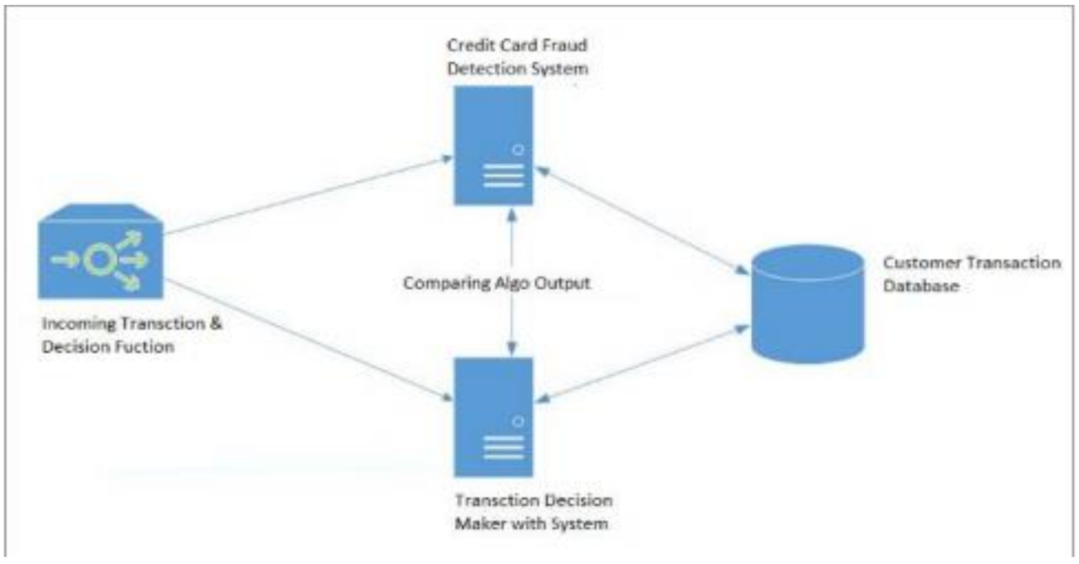

Incoming Transaction
Fraud Detection System
Isolation Forest Algorithm
Decision Function
Local Outlier Factor Algorithm
Sample Dataset
Customer Transaction Database
Comparing Algorithm Output
Fraudlent Transaction
Alert the Bank
0
Legitimate Transaction
Allow Transaction

2.METHODOLOGY

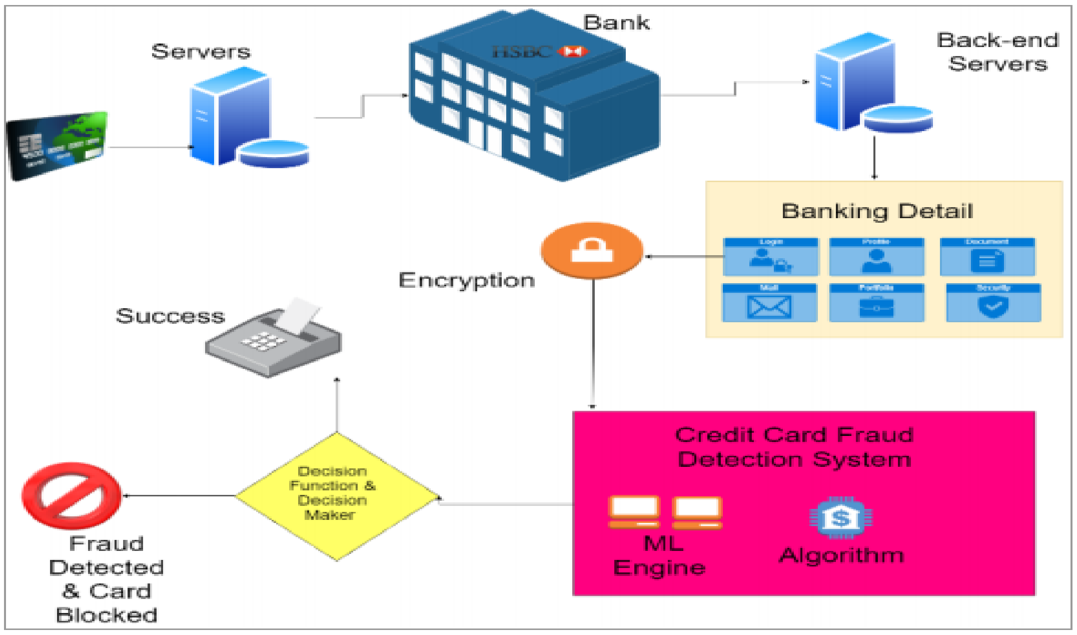
2.1) Architecture:

The approach that this paper proposes, uses the latest machine learning algorithms to detect anomalous activities, called outliers.

The basic rough architecture diagram can be represented with the following figure:


Credit Card Fraud Detection System
Comparing Algo Output
Customer Transaction Database
Incoming Transction & Decision Fuction
Transction Decision Maker with System

When looked at in detail on a larger scale along with real life elements, the full architecture diagram can be represented as follows:



2.2) Proposed System:

In proposed system, we present a Hidden Markov Model (HMM). Which does not require fraud signatures and yet is able to detect frauds by considering a cardholder’s spending habit. Card transaction processing sequence by the stochastic process of an HMM. The details of items purchased in Individual transactions are usually not known to any Fraud Detection System (FDS) running at the bank that issues credit cards to the cardholders. Hence, we feel that HMM is an ideal choice for addressing this problem.

An FDS runs at a credit card issuing bank. Each incoming transaction is submitted to the FDS for verification. FDS receives the card details and the value of purchase to verify, whether the transaction is genuine or not. The types of goods that are bought in that transaction are not known to the FDS. It tries to find any anomaly in the transaction based on the spending profile of the cardholder, shipping address, and billing address, etc. If the FDS confirms the transaction to be of fraud, it raises an alarm, and the issuing bank declines the transaction.

3.SYSTEM ANALYSIS

3.1) System Requirement Specification:

3.1.1) Functional Requirements:

Purpose:

Accuracy: It is defined as a portion of all the number of transactions which are identified correctly. That is the genuine transactions as genuine and fraud transactions as fraud.

Input:

Making transactions....

Output:

Checking Whether transaction is genuine or not (fraud or not).

3.1.2) Non-Functional Requirements:

Usability:

This system is easy to use. The user will reach to the summarized output with one button. Because one of the software's feature is timesaving.

Reliability:

This software will be developed with machine learning and deep learning techniques. Here user provided data will be compared with result and measure reliability with recent machine learning techniques, user gained data should be enough for reliability if enough data is obtained.

Performance:

Calculation time and Response time should be as little as possible, because one of the software's features is time saving. Our project response time is less than 30 seconds.

Supportability:

The system should require Python knowledge to maintenance. It does not need any server side requirements. User side problems will be fixed with an update and it also require code knowledge and network knowledge.

3.1.3) External Interface Requirements:

User Interfaces:

Some of the logical characteristics of user interface that the system needs are buttons in keyboard such as enter and escape which is used to direct and escape from a site respectively.

Hardware Interfaces:

In this interface we describe the characteristics of each interface between the software and hardware components of the system. The hardware interface that are used in this project are web cam, keyboard etc.

Software Interfaces:

In this interface we describe the connections between our project and other software components (identified by name and version), including databases, operating systems, tools, libraries, and integrated commercial components. In this project the libraries that used are Six pip, tensor flow…, etc. softwares used are python 3.4.7 and opencv.

Communication Interfaces:

In our project the requirement used for communication purpose is web browser.

3.1.4) System Requirements:

All computer software needs certain hardware components or other software resources to be present on a computer. These pre-requisites are known as system. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources. There are two types of requirements. They are hardware and software.

Hardware Requirements:

The most common set of requirements defined by any operating system is the physical computer resources, also known as hardware, A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating system in newer versions of software, system requirements tend to increase over time.

Software Requirements:

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optional functioning of an application. These requirements or prerequisites are generally not included in the software installation package and need to be installed separately before the software is installed.

The software requirements that are used in our project are:

- Python 3.4.7
- numpy
- Pandas
- seaborn
- scipy
- skylearn

3.2) Feasibility Study

3.2.1) Technical Feasibility:

This is carried out to check the technical feasibility, that is, the technical requirement of the system. Any system developed must not have a high demand on the available technical resources. Internet is the only technical requirement for this project.

3.2.2) Operational Feasibility:

It is mainly concerned with issues like whether the system will be used if it is developed and implemented. This proposed system is really benefitable when compared to the existed system. Users using this will stay contented. This project can be further implemented.

3.3.3) Economical Feasibility:

This is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into development of this project. And there is no organization economically depended for this project.

4.SYSTEM DESIGN

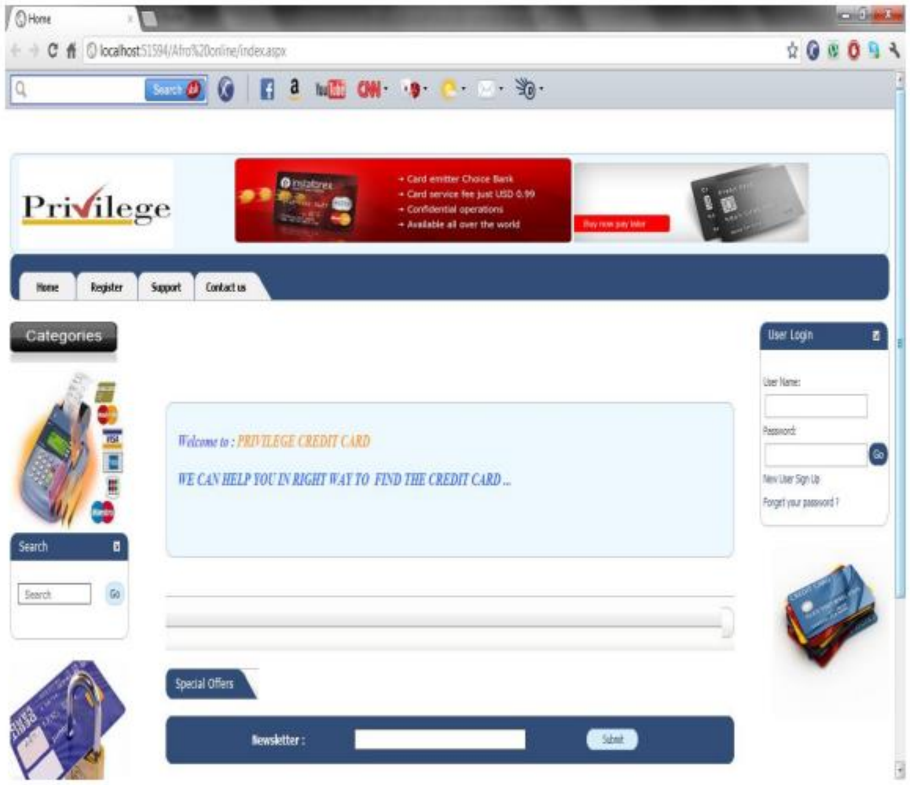
4.1) Introduction:

System design is the process of defining the architecture, modules, interfaces and data file system to satisfy specified requirements. System design could be seen as the application of system theory to product development.

4.2) System Modules:

The following are the modules used in the project: Admin and User modules. In the Admin module, the administrator takes care of the user accounts and all the genuine and fraud transactions. Whenever, a fraud is detected, immediately a notice to the user will be given through an email. In case, if a genuine user's transaction is detected as fraud and then if the user requests the administrator, the blocked credit-card will be released by the administrator and user can use it again normally.

In the bellowed diagram the administrator can login with his credentials to manage his/her personal account as well as carry out other functionalities.



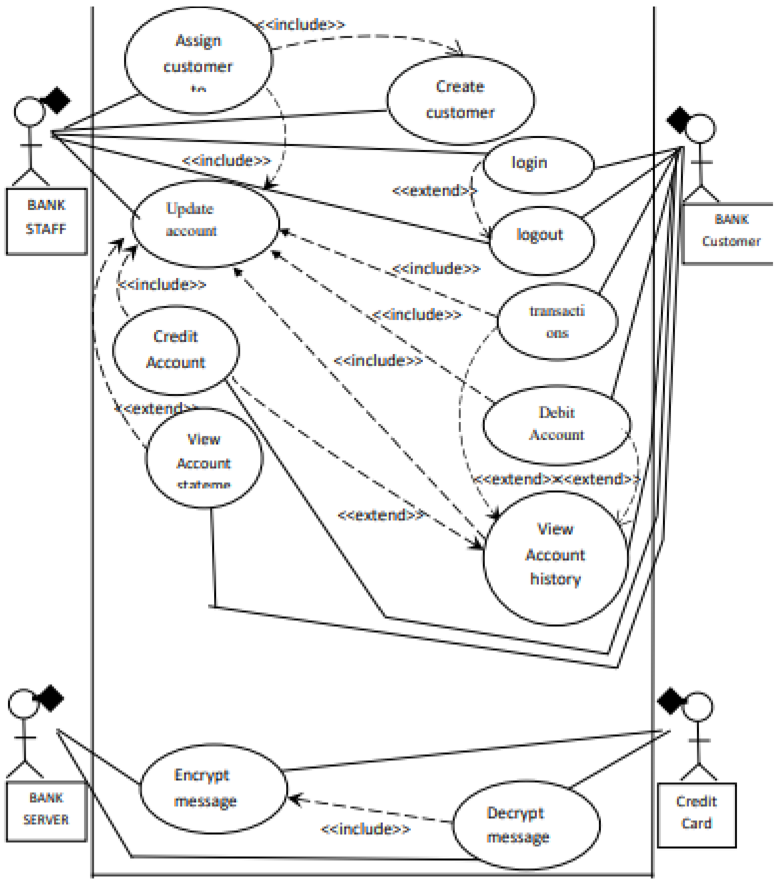
Welcome to : PRIVILEGE CREDIT CARD

WE CAN HELP YOU IN RIGHT WAY TO FIND THE CREDIT CARD ...

4.3) UMl Diagrams:

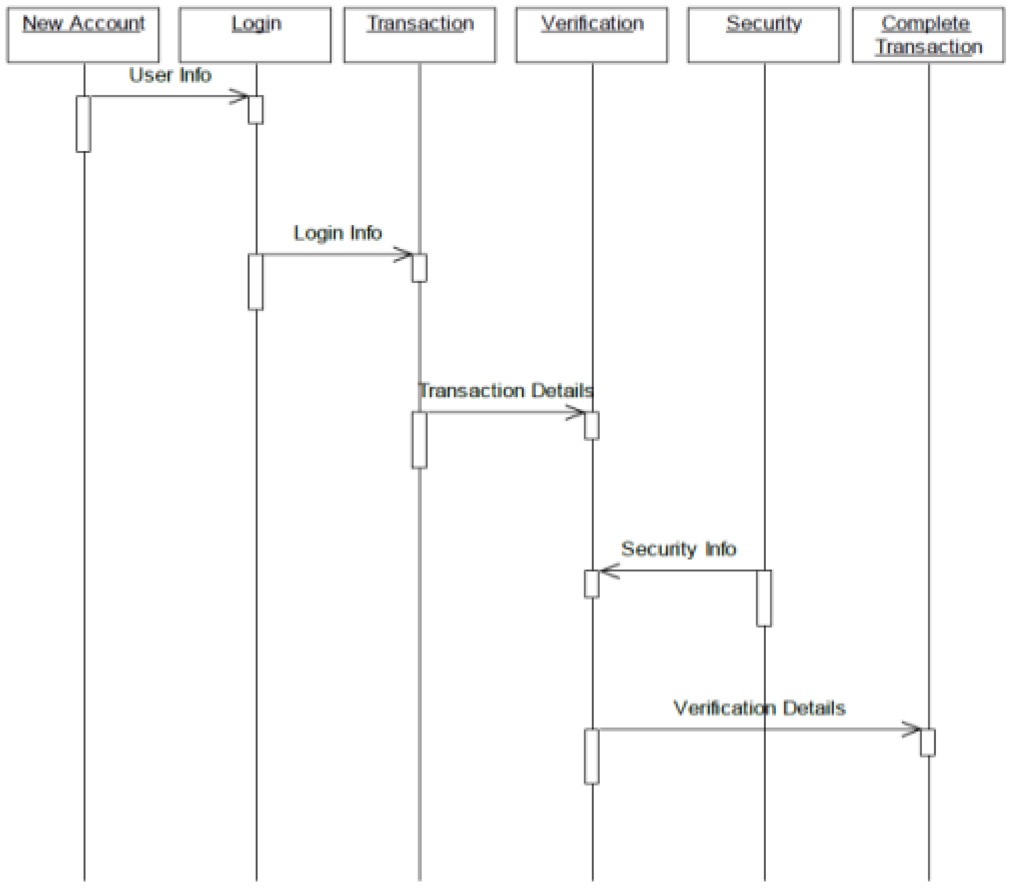
4.3.1) Use-Case Diagrams:

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.



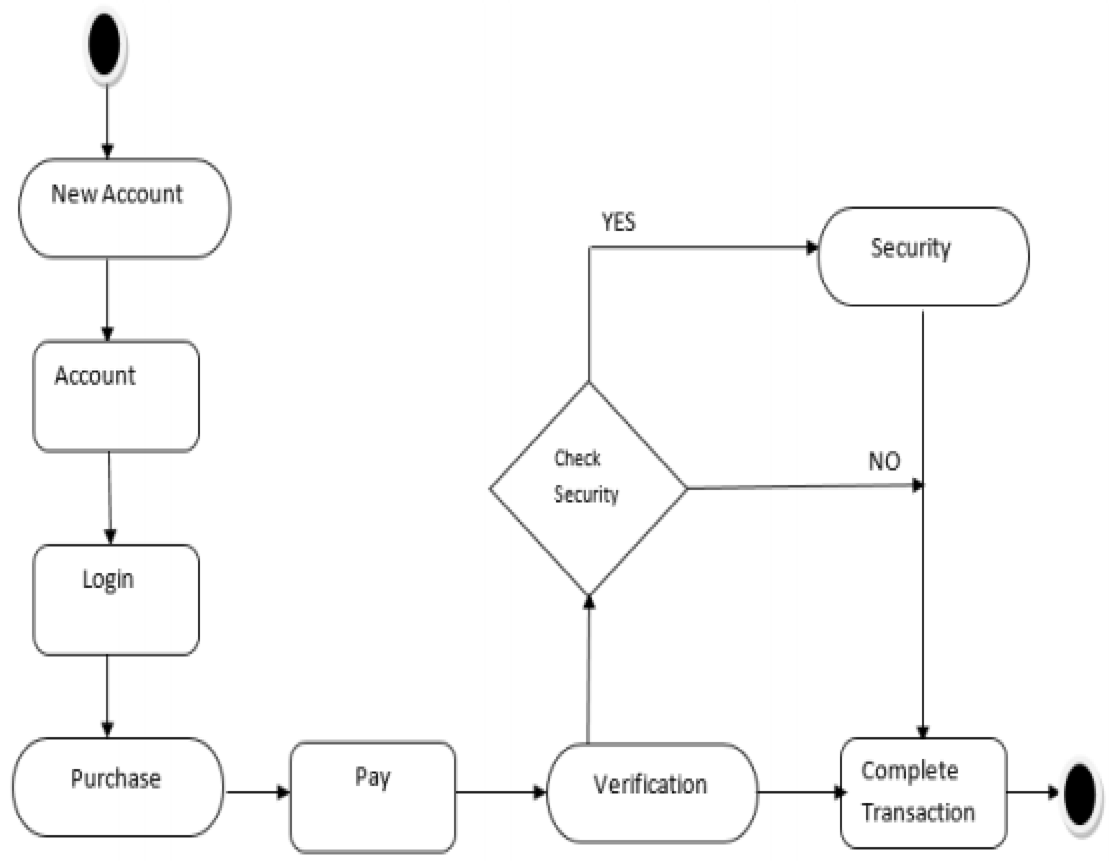
4.3.2) Sequence Diagram:

A sequence diagram shows object interactions arranged in time sequence, it depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.



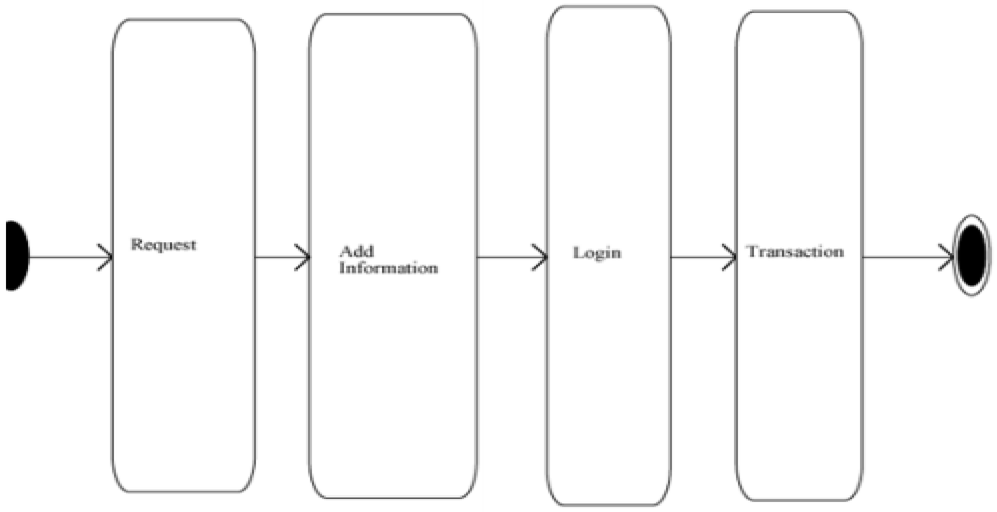
4.3.3) Activity Diagrams:

Activity diagrams ae graphical representations of workflow of stepwise activities and actions with support for choice, iteration and concurrency.



4.3.4) State chart Diagram:

A state diagram is a type of diagram used in computer science and related fields to describe the behaviour of system. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times is a reasonable abstraction.



5.SYSTEM IMPLEMENTATION

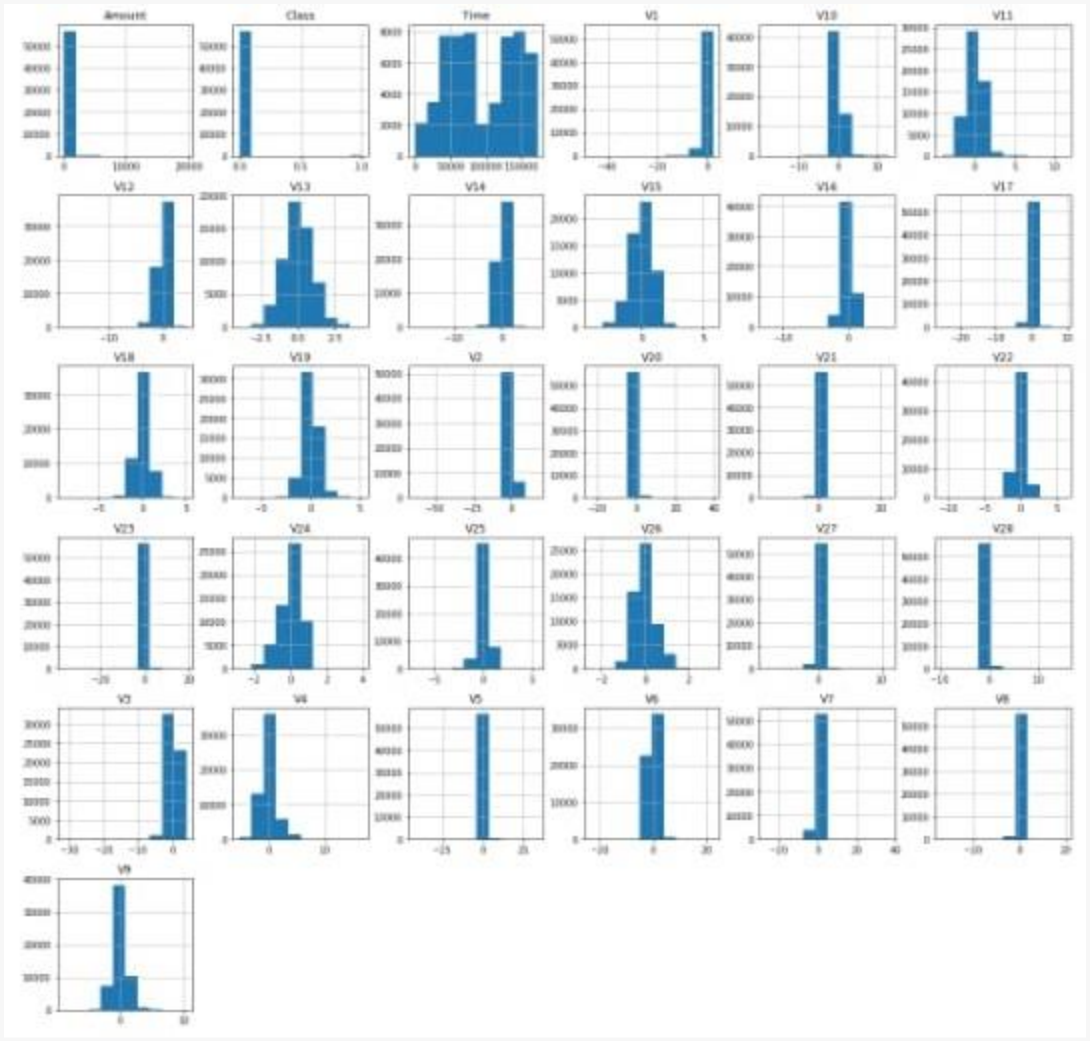
```python
import numpy as np
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import sys
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
import sklearn
data = pd.read_csv('../input/creditcard.csv')
print(data.columns)
data.shape
data = data.sample(frac = 0.2, random_state = 1)
print(data.shape)
# plot the histogram of each parameter
data.hist(figsize = (20, 20))
plt.show()
# determine the number of fraud cases
fraud = data[data['Class'] == 1]
valid = data[data['Class'] == 0]

outlier_fraction = len(fraud) / float(len(valid))
print(outlier_fraction)

print('Fraud Cases: {}'.format(len(fraud)))
print('Valid Cases: {}'.format(len(valid)))
# correlation matrix
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))

sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
# get the columns from the dataframe
columns = data.columns.tolist()

# filter the columns to remove the data we do not want
columns = [c for c in columns if c not in ['Class']]

# store the variable we will be predicting on which is class
target = 'Class'

# X includes everything except our class column
X = data[columns]
# Y includes all the class labels for each sample
# this is also one-dimensional
Y = data[target]

# print the shapes of X and Y
```

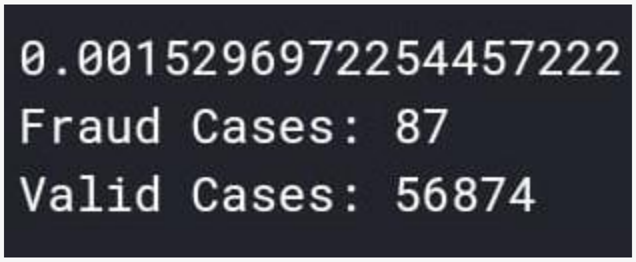
```
print(X.shape)
print(Y.shape)
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
# define a random state
state = 1

# define the outlier detection methods
classifiers = {
    # contamination is the number of outliers we think there are
    'Isolation Forest': IsolationForest(max_samples = len(X),
                                        contamination = outlier_fraction,
                                        random_state = state),
    # number of neighbors to consider, the higher the percentage of outliers the
higher you want to make this number
    'Local Outlier Factor': LocalOutlierFactor(
    n_neighbors = 20,
    contamination = outlier_fraction)
}
```

Fit the model

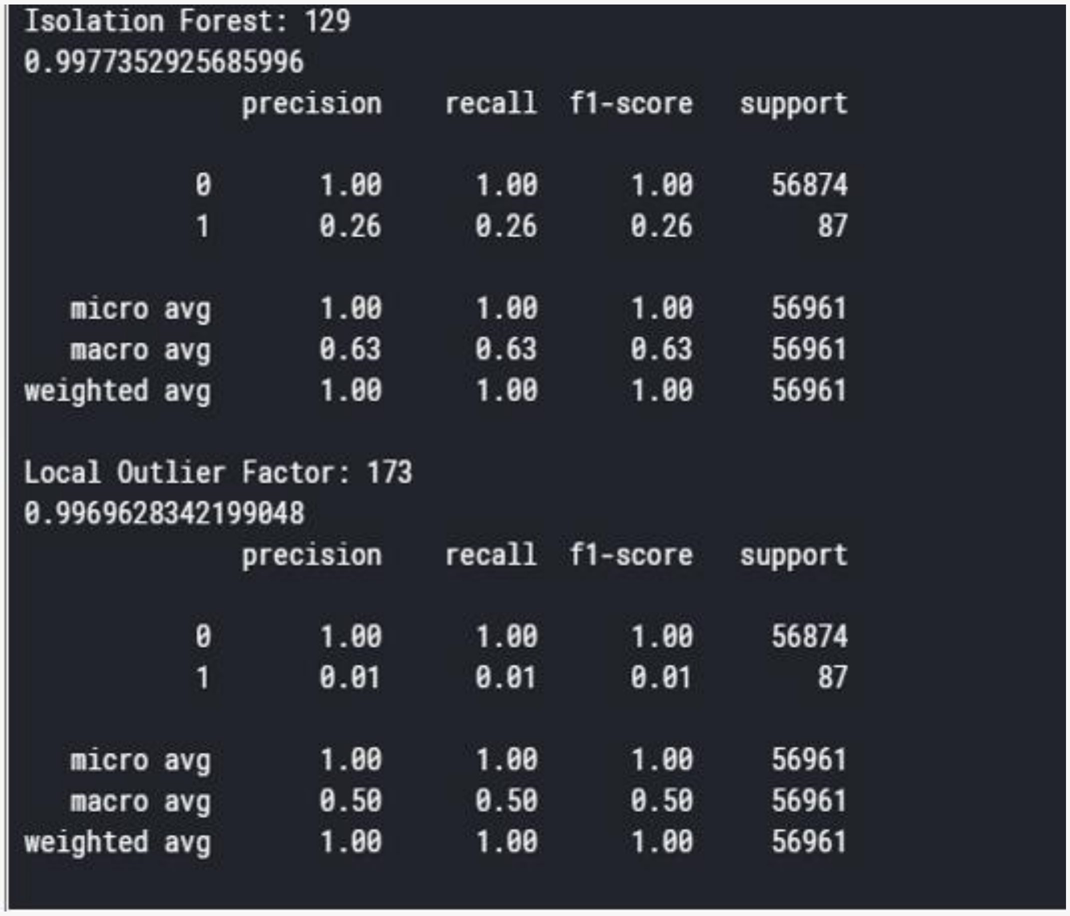
```
n_outliers = len(fraud)

for i, (clf_name, clf) in enumerate(classifiers.items()):

    # fit the data and tag outliers
    if clf_name == 'Local Outlier Factor':
        y_pred = clf.fit_predict(X)
        scores_pred = clf.negative_outlier_factor_
    else:
        clf.fit(X)
        scores_pred = clf.decision_function(X)
        y_pred = clf.predict(X)

    # reshape the prediction values to 0 for valid and 1 for fraud
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1

    # calculate the number of errors
    n_errors = (y_pred != Y).sum()
     # classification matrix
    print('{}: {}'.format(clf_name, n_errors))
    print(accuracy_score(Y, y_pred))
    print(classification_report(Y, y_pred))
```

6.RESULTS





```
0.0015296972254457222
Fraud Cases: 87
Valid Cases: 56874
```



```
Isolation Forest: 129
0.9977352925685996
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56874
           1       0.26      0.26      0.26        87

   micro avg       1.00      1.00      1.00     56961
   macro avg       0.63      0.63      0.63     56961
weighted avg       1.00      1.00      1.00     56961

Local Outlier Factor: 173
0.9969628342199048
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56874
           1       0.01      0.01      0.01        87

   micro avg       1.00      1.00      1.00     56961
   macro avg       0.50      0.50      0.50     56961
weighted avg       1.00      1.00      1.00     56961
```

7.SOFTWARE TESTING

7.1) INTRODUCTION:

Software testing or Quality Assurance strategies describe how to mitigate product risks of stakeholders at the test level, which kinds of testing are to be done and which entry criteria will apply. They're made based on development design documents.

7.2) Types of testing strategies:

The various types of testing are:

- White Box Testing
- Black Box Testing
- Unit Testing
- Integration Testing
- Validation Testing
- Output Testing
- User Acceptance Testing

7.2.1) Unit Testing:

In this testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing.

The module of the system is tested separately. This testing is carried out during programming stage itself. In this testing step each module is found to working satisfactorily as regard to the expected output from the module. There are some validation checks for fields also. It is very easy to find error debut in the system.

7.2.2) Validation Testing:

At the culmination of the black box testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests. That is, validation tests begin, validation testing can be defined many ways but a simple definition is that validation succeeds when the software functions in manner that can be reasonably expected be the customer. After validation test has been conducted one of the two possible conditions exists. The functions or performance characteristics confirm to specification and are accepted.

7.2.3) System Testing:

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

7.2.4) Output Testing:

After performance of the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specific format. Asking the user about the format required by system tests the output displayed or generated by the system under consideration.

Here the output format is considered the of screen display. The output format on the screen is found to be correct as the format was designed in the system phase according to the user need. For the hard copy also the output comes out as specified by the user. Hence the output testing does not result in any correction in the system.

8.CONCLUSIONS

Credit card fraud is without a doubt an act of criminal dishonesty. This article has listed out the most common methods of fraud along with their detection methods and reviewed recent findings in this field. This paper has also explained in detail, how machine learning can be applied to get better results in fraud detection along with the algorithm, pseudocode, explanation its implementation and experimentation results.

While the algorithm does reach over 99.6% accuracy, its precision remains only at 28% when a tenth of the data set is taken into consideration. However, when the entire dataset is fed into the algorithm, the precision rises to 33%. This high percentage of accuracy is to be expected due to the huge imbalance between the number of valid and number of genuine transactions.

9)FUTURE SCOPE

- While we couldn’t reach out goal of 100% accuracy in fraud detection, we did end up creating a system that can, with enough time and data, get very close to that goal. As with any such project, there is some room for improvement here.

- The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result.

- This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.

- More room for improvement can be found in the dataset. As demonstrated before, the precision of the algorithms increases when the size of dataset is increased. Hence, more data will surely make the model more accurate in detecting frauds and reduce the number of false positives. However, this requires official support from the banks themselves.

10)REFERENCES

[1] “Credit Card Fraud Detection Based on Transaction Behaviour -by John Richard D. Kho, Larry A. Vea” published by Proc. of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, November 5-8, 2017

[2] CLIFTON PHUA1, VINCENT LEE1, KATE SMITH1 & ROSS GAYLER2 “ A Comprehensive Survey of Data Mining-based Fraud Detection Research” published by School of Business Systems, Faculty of Information Technology, Monash University, Wellington Road, Clayton, Victoria 3800, Australia

[3] “Survey Paper on Credit Card Fraud Detection by Suman” , Research Scholar, GJUS&T Hisar HCE, Sonepat published by International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issue 3, March 2014

[4“Credit Card Fraud Detection-by Ishu Trivedi, Monika, Mrigya, Mridushi” published by International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 1, January

[5] “Credit Card Fraud Detection through Parenclitic Network AnalysisBy Massimiliano Zanin, Miguel Romance, Regino Criado, and SantiagoMoral” published by Hindawi Complexity Volume 2018, Article ID 5764370, 9 pages

[6] “Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy” published by IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 29, NO. 8, AUGUST 2018