A

Major Project Report

on

CREDIT CARD FRAUD DETECTION

Submitted in Partial Fulfillment of
the Requirements for the Degree
of

Bachelor of Engineering

in

Computer Engineering

to

North Maharashtra University, Jalgaon

Submitted by

Rajeshwari Samadhan Sonawane
Gopal Anand Agrawal
Minal Ramakrishna Pawar
Vaishnavi Sunil Patil
Sakshi Dinesh Patil

Under the Guidance of

Prof. Dr. Satpalsing D. Rajput

DEPARTMENT OF COMPUTER ENGINEERING
SSBT's COLLEGE OF ENGINEERING AND TECHNOLOGY,
BAMBHORI, JALGAON - 425 001 (MS)

2020 - 2021

SSBT's COLLEGE OF ENGINEERING AND TECHNOLOGY, BAMBHORI, JALGAON - 425 001 (MS)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the major project entitled *credit card fraud detection*, submitted by

Rajeshwari Samadhan Sonawane
Gopal Anand Agrawal
Minal Ramakrishna Pawar
Vaishnavi Sunil Patil
Sakshi Dinesh Patil

in partial fulfillment of the degree of *Bachelor of Engineering* in *Computer Engineering* has been satisfactorily carried out under my guidance as per the requirement of North Maharashtra University, Jalgaon.

Date: July 13, 2021
Place: Jalgaon

Prof. Dr. Satpalsing D. Rajput
Guide

Prof. Dr. Girish K. Patnaik
Head

Prof. Dr. K. S. Wani
Principal

Acknowledgements

This project required a lot of guidance and assistance from many people. All that have done is only due to such supervision and assistance and would not forget to thank them. Like to express my deep gratitude and sincere thanks to all who helps us in this major project. Would like to thanks to Principal Prof. Dr. K. S. Wani, SSBTCOET for having provided us facilities to complete work. Deep gratitude goes to Prof. Dr. G. K. Patnaik, head of the department and major project guide Dr. Satpalsing D. Rajput , for granting us opportunity to conduct this major project work as well as his valuable suggestions and guidance at the time of need. Sincerely thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staffs of Computer Department of SSBT's COET which helped us in our project work. Also would like to extend our sincere esteems to all staff in laboratory for their timely support. I'm also thankful to my Parents. Last but not least thankful to God.

Rajeshwari Samadhan Sonawane
Gopal Anand Agrawal
Minal Ramakrishna Pawar
Vaishnavi Sunil Patil
Sakshi Dinesh Patil

Contents

List of Figures

Abstract

Credit card fraud detection is presently the most frequently occurring problem in the present world. This is due to the rise in both online transactions and e-commerce platforms. Credit card fraud generally happens when the card was stolen for any unauthorized purposes or even when the fraudster uses the credit card information for his use. In the present world, we are facing a lot of credit card problems. To detect fraudulent activities the credit card fraud detection system was introduced. This project aims to focus mainly on machine learning algorithms. An HMM is initially trained with the normal behavior of a cardholder. If an incoming credit card transition is not accepted by the trained HMM with sufficiently high probability, it is considered to be fraudulent. At the same time, we try to ensure that genuine transactions are not rejected.

Chapter 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 behavior 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 credit card for personal reasons while the owner and the card-issuing authorities are unaware of the fact that the card is being used. Fraud detection involves monitoring the activities of populations of users to estimate, perceive or avoid objectionable behavior, which consists of fraud, intrusion, and defaulting. This is a 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 time.

1.1 Background

Fraud act as an unlawful or criminal deception intended to result in financial or personal benefit. It is a deliberate act that is against the law, rule, or policy intending to attain unauthorized financial benefit. Numerous literature's about anomaly or fraud detection in this domain have been published already and are available for public usage. A comprehensive survey conducted by Clifton Phua and his associates has revealed that techniques employed in this domain include data mining applications, automated fraud detection, adversarial detection.

1.2 Motivation

Nowadays most of the transaction takes place online, meaning that credit card and other online payments system are involved. This method is convenient for the consumer. Consumers save time because they don't have to go to the store to make their purchases and companies save money by not owning physical stores and avoiding expensive rental payments. With careful designing, most of the problem can be eliminated but even the framework which was used to create the server is not perfect.

1.3 Problem definition

Credit card frauds can be made in many ways such as simple theft, application fraud, counterfeit cards, never received issue (NRI), and online fraud. In online fraud, the transaction is made remotely and only the cards details are needed. A manual signature, a PIN, or a card imprint are not required at the time of purchase. Though prevention mechanisms like CHIP and PIN decrease the fraudulent activities through simple theft, counterfeit cards. Online frauds are still increasing in both amount and number of transactions. When the fraudsters obtain a card, they usually use (spend) all of its available (unused) limit.

According to the statistics, they do this in four-five transactions, on average. Thus, for the fraud detection problem, although the typical prediction modeling performance measures are quite relevant, as indicated by the bank authorities, a performance criterion, measuring the loss that can be saved on the cards whose transactions are identified as fraud is more prominent. In other words, detecting fraud on a card having a larger available limit is more valuable than detecting fraud on a card having a smaller available limit. As a result, what we are faced with is a classification problem with variable declassification costs. Each false negative has a different declassification cost and the performance of the model should be evaluated over the total amount of saved available usable limits instead of the total number of frauds detected.

1.4 Scope

While we couldn't reach our goal of 100 percent 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 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. However, the output of these algorithm 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 the 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.

1.5 Objective

The objective of this project to reduce losses due to payment fraud.

- It is a user-friendly application.
- Higher accuracy of fraud detection.
- Less manual work needed for additional verification.
- Ability to identify new patterns and adapt to changes.
- Fewer false declines.

1.6 Selection of Life cycle Model for Development

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear- sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The waterfall model is the earliest SDLC approach that was used for software development.

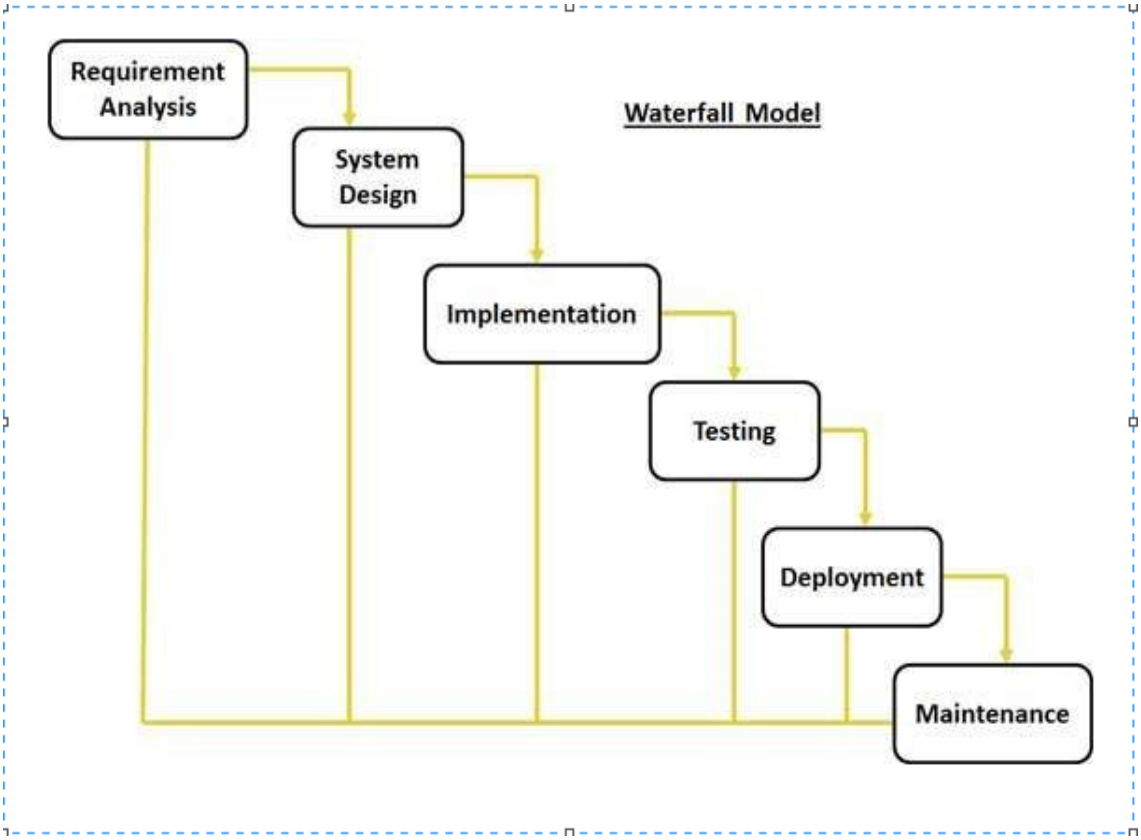


Figure 1.1: Life Cycle Model For credit card fraud detection

1.7 Organization of Report

The report is described in the following way.
Chapter 1 titled Introduction, describes Background, Motivation, Problem Definition, Scope, and Objective.
Chapter 2 titled Project planning and Management, Which presents Feasibility Study, Risk Analysis, Project Scheduling, Effort Allocation, and Cost Estimation.
Chapter 3 titled Analysis, which presents Software, Hardware, Functional, Non-functional, requirements, and Software requirement specification.
Chapter 4 titled Design presents System Architecture and UML diagrams.
Chapter 5 titled Conclusion and Future work, concludes with the major project.

1.8 Summary

In this first chapter, Introduction about credit card fraud detection. In the next chapter, Project Planning and Management is presented.

Chapter 2

Project Planning and Management

Project planning is part of project management, which relates to the use of schedules such as Gantt chart stop landed subsequently report progress within the project environment. Project planning can be done manually or by the use of project management software. Project management is the practice of initiating, planning, executing, controlling, and closing the work of a team to achieve specific goals and meet specific success criteria at the specified time. The primary challenge of project management is to achieve all of the project goals within the given constraints. Every client has strategic goals and the projects that we do advance those goals. Project management is important because it ensures there's rigor in architecting projects properly so that they fit well within the broader context of our client's strategic frameworks. Good project management ensures that the goals of projects closely align with the strategic goals of the business.

The organization of the chapter is as follows: Section 2.1 describes the Feasibility Study, Risk Analysis is presented in Section 2.2, Section 2.3 describes Project Scheduling, Section 2.4 describes Effort Allocation, and Section 2.5 describes Cost Estimation. Finally, the summary is presented in the last section.

2.1 Feasibility study

The feasibility study is the process of planning and developing a new system or improves the existing system. It means the evaluation or analysis of the project. A well-designed feasibility study should provide a historical background of the project. A feasibility study is done by investing in the existing system in the area under investigation and generating ideas about the project. A feasibility study is done by evaluating the existing system of the area undertakes investigation and generating ideas for a new style.

There are few types of feasibility are exist. So developers should take care of this feasibility:-

1. Technical Feasibility
2. Economic Feasibility
3. Operational Feasibility

2.1.1 Technical feasibility

The project is technically feasible can be built using the existing available technologies. The technology required to detect fraudulent transactions is available and hence it is technically feasible and extended. The portal is developed using HTML, CSS as front end and Python, MySQL as back end. Windows used for the development of the portal is very easy to use. All the required hardware and software are easily available in the market. Hence, the portal is technically feasible.

2.1.2 Economic feasibility

The project is economically feasible as the necessary hardware and software are available in the market at a low cost. As the data samples increase, which consumes more time and processing power. In that case, a better processor might be needed.

The amount of funds that pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system is well within the budget and this was achieved because most of the technologies used are freely available.

2.1.3 Operational feasibility

The project is operationally feasible as the user having basic knowledge about computers and the Internet. The proposed system has a user-friendly user interface so the user does not have any knowledge about the domain can also work efficiently.

2.2 Risk Analysis

Risk analysis is and management is a series of steps that help a software team to understand and managing certainty. Many problems can plague as of ware project. A risk is a potential problem it might happen, it might not. But, regardless of the outcome, it's good to identify it, assess its probability of occurrences estimate its impact, and establish a contingency plan should the problem occurred. Recognizing what can go wrong is the first step, called Risk identification. Once information is established, risks are ranked, by probability and impact. Finally, a plan is developed to manage those risks with high probability and high impact. When risks are analyzed it is important to quantify the level of uncertainty and the degree of loss is associated with each risk to accomplish this.

2.2.1 Project Risk

The location must be appropriate. GPS performance going in the right way. As we are dealing with money, the model shows that uncertainty is regarded as the biggest risk factor. Any bug in the system can result in many fraud transactions which cause a huge loss of money. That's why risk is very high.

2.2.2 Technical Risk

Threatens the quality and the timelines of the software to be produced. If a technical risk becomes a reality, an implementation may become difficult or impossible.

2.2.3 Business Risk

Threatens the viability of the software to be built. Business risk often jeopardizes the project or the product.

2.3 Project Scheduling

Project Scheduling is a mechanism to communicate what tasks need to get done and which organizational resources will be allocated to complete those tasks in what timeframe. Project scheduling activity is used to schedule the software development lifecycle. Project scheduling is important to note, however, that the schedule evolves over time. During the early stages of project planning microscopic schedule is developed. The schedule identifies all major software engineering activities. Also identifies the product functions to which they are applied. So, the Gantt chart is as follow:

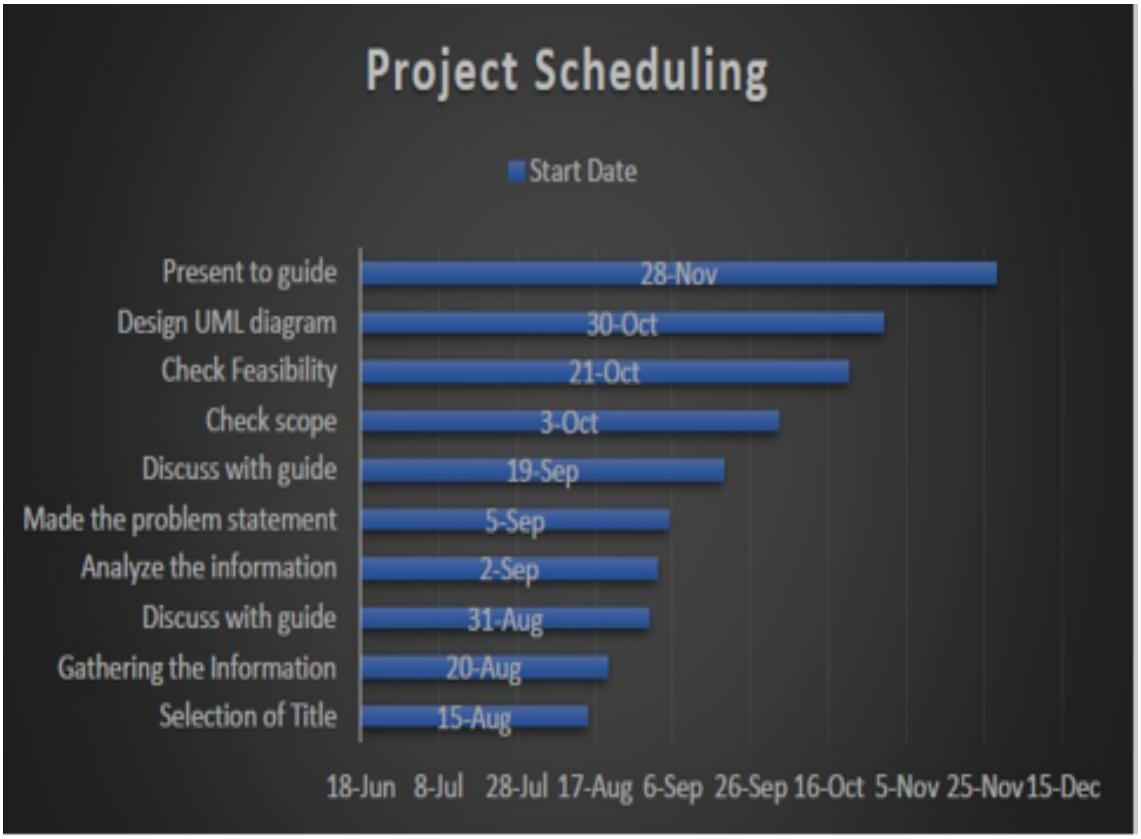


Figure 2.1: Project Scheduling Diagram

2.4 Effort allocation

Effort allocation is a feature that is great for indicating how much of an assigned day a task should take. Before, it was only possible to enter and view how much effort a task would take as a percentage (i.e. what percent of your day the task would take). The project is developed by a combination of the effort of the team. So the whole project is divided into modules and the number of modules is allotted to team members. After completion of each module, it will be linked from one module to another module to form a complete project.

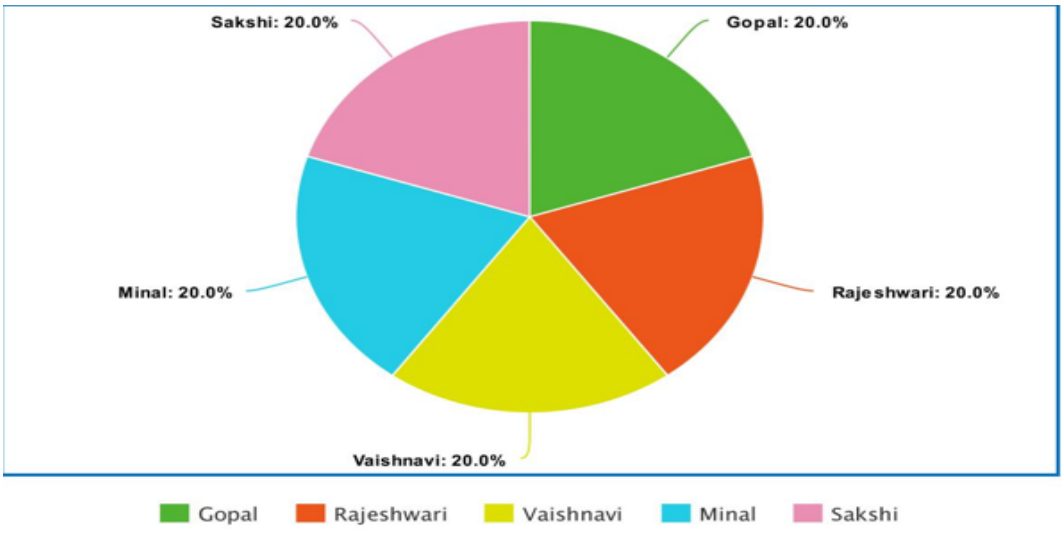


Figure 2.2: Chart of Effort Allocation

2.5 Cost Estimation

A cost estimate is the approximation of the cost of a program, project, or operation. The cost estimate is the product of the cost estimating process. The cost estimate has a single total value and may have identifiable component values. A problem with a cost overrun can be avoided with a credible, reliable, and accurate cost estimate. A cost estimate or is the professional who prepares cost estimates. There are different types of cost estimators, whose title may be preceded by a modifier, such as building estimator, oral ethical estimator, or chief estimator. For any software project, it is necessary to know the total cost required for the development of the project and the development time requires. Cost Estimation is needed before development is initiated. The cost of any software project is calculated by the formula.

C = aLb

Where; C = cost of project

a = 1.4 (constant)

b = 0.93 (constant)

L = size of code

2.6 summary

In this chapter, the Planning of the project of management is discussed. In the next chapter Analysis of the project is explained.

Chapter 3

Analysis

The analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it. The analysis phase is used to ensure that everyone understands the vision of the project; it also defines a clear scope. This will help with decisions on nice-to-have features that may be mentioned along the way but that may not be necessary to meet your initial project goals. These are usually the culprits in never-ending implementation projects.

The organization of the chapter is as follow: Section 3.1 describes Requirements Collection and Identification, Section 3.2 Software Requirement Specification, Section 3.3 describes Software Requirement Specification, Section 3.3.1 describes Product Features, Section 3.3.2 describes Functional Requirements, Section 3.3.3 Non- Functional Requirements, Section 3.3.4 External Interfaces, Section 3.4 Summary.

3.1 Hardware Requirements

1. Processor type: Pentium III-compatible processor or faster.
2. Processor speed: Minimum: 1.0 GHz, Recommended: 2.0 GHz or faster
3. RAM: 512 MB or more
4. HARD DISK: 20GB or More
5. Monitor: VGA or higher resolution 800x600 or higher resolution
6. Pointing device: Microsoft Mouse or compatible pointing device
7. CD-ROM: Actual requirements will vary based on system configuration and the applications and features chosen to install.

3.2 Software Requirement

1. Application software
2. Framework: Django
3. Database: MySQL
4. Back End: Python
5. Front End: HTML, CSS, JavaScript
6. Operating System: Windows 7 or Higher, Linux (Ubuntu)

3.3 Software requirement specification

This section describes both the functional and non-functional requirements of the system. The functional requirement section defines the system's external interface, general requirements, performance, design constraints, etc. The credit card fraud detection system has been developed to alert the customer regarding the fraud of their credit card. After the payment process, the transactions performed are verified whether the performed transaction is a real or fraud transaction and minimizes the false alert by implementing a genetic algorithm.

3.3.1 Product Features

In this project, we are using functions such as :

- The proposed method overcomes the low accuracy forecast problem.
- Fast and reliable solution is attained
- Utilizing the latest AI methods, the fraudulent transactions are recognized and the false alerts are reduced.
- The detection of the fraud use of the card is found much faster than the existing system.
- In the case of the existing system even the original cardholder is also checked for fraud detection. But in this system no need to check the original user as we maintain a log.
- The log which is maintained will also be proof for the bank for the transaction made.
- We can find the most accurate detection using this technique.
- This reduces the tedious work of an employee in the bank.

3.3.2 Functional Requirement

Functional Requirement describes what the portal has to do. Functional Requirements are as follows:

- Users get accurate predictions about Positive, Negative, and Recovered Cases which are available on one portal.
- System extracts the information from the Database and provides this information to the users according to accurate predictions.
- The database server stores the information about all the predicted cases.

3.3.3 Non-Functional Requirement

Non-functional Requirement is mostly quality requirement. That stipulates how well the Portal does, what it has to do. Other than functional requirements, in practice, this would in detail analysis of issues such as availability, security, usability, and maintainability.

Non-Functional requirements are as follows.

- Portal is unique and simplified.
- Prediction of cases is easy to capture.
- This portal is designed for a user-friendly environment so that the peoples do not face any difficulty regarding the accurate prediction of cases.

3.3.4 External Interfaces User Interfaces

The system performance is adequate. However, Credit Card Detection System is working with the user internet connection.

3.4 Summary

In this third chapter, Analysis of credit card fraud detection system is presented. In the next chapter, Design is presented.

Chapter 4

Design

Project design is an early phase of the project where a project's key features, structure, criteria for success, and major deliverables are all planned out. The point is to develop one or more designs that can be used to achieve the desired project goals stockholders can then choose the best design to use for the actual execution of the project. The project design phase might generate a variety of different output, including sketches, flowcharts. A project design is a strategic organization of ideas materials and processes to achieve a goal. Project managers rely on a good design to avoid pitfalls and provide parameters to maintain crucial aspects of the project, like the schedule and the budget.

The organization of the chapter is as follows section 4.1 describes system Architecture. Data flow diagrams are presented in section 4.2 section, Section 4.3 describe UML diagrams

4.1 System architecture

The system architecture provides details on how the components or modules are integrated and is described with the help of a unified modeling program figure that shows the system architecture in detail.

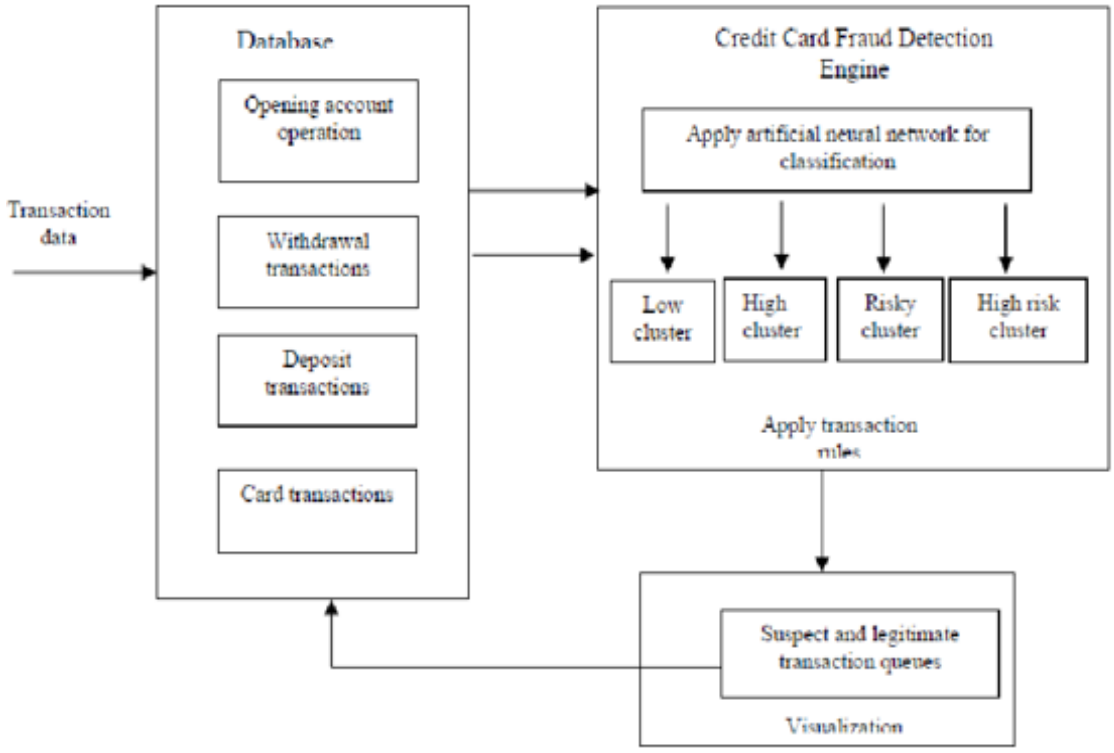


Figure 4.1: System Architecture for credit card fraud detection

4.2 Data flow diagram

A data flow diagram (DFD) is a graphical representation of the flow of the student information system. A data flow diagram can also be used for the visualization of data processing. DFD shows the integration between the system and outside entities. The context level DFD is then exploded to show more details of the system being modeled In the DFD there are four symbols :

1. A square defines a source(originator) or destination of system data.
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

4.2.1 Level 0 Data Flow Diagram

The figure shows the Data Flow Diagram of Level 0. It consists of A square defines a source as a user, an arrow identifies data flow as information, A circle or a bubble represents a process of credit card fraud detection.

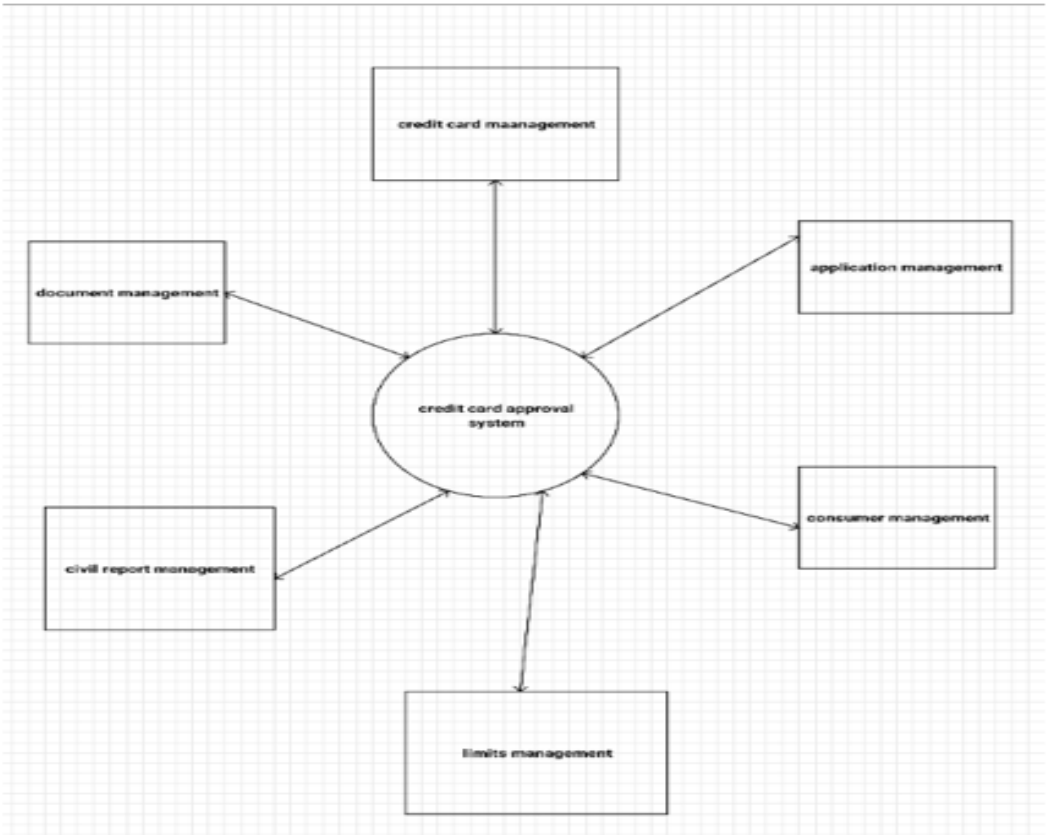


Figure 4.2: Level 0 Data Flow Diagram for credit card fraud detection

4.2.2 Level 1 Data Flow Diagram

Figure , shows the Data Flow Diagram of Level 1. It consists of A square defines a source as user, an arrow identifies data flow as information, A circle or a bubbleRepresents a process of credit card fraud detection

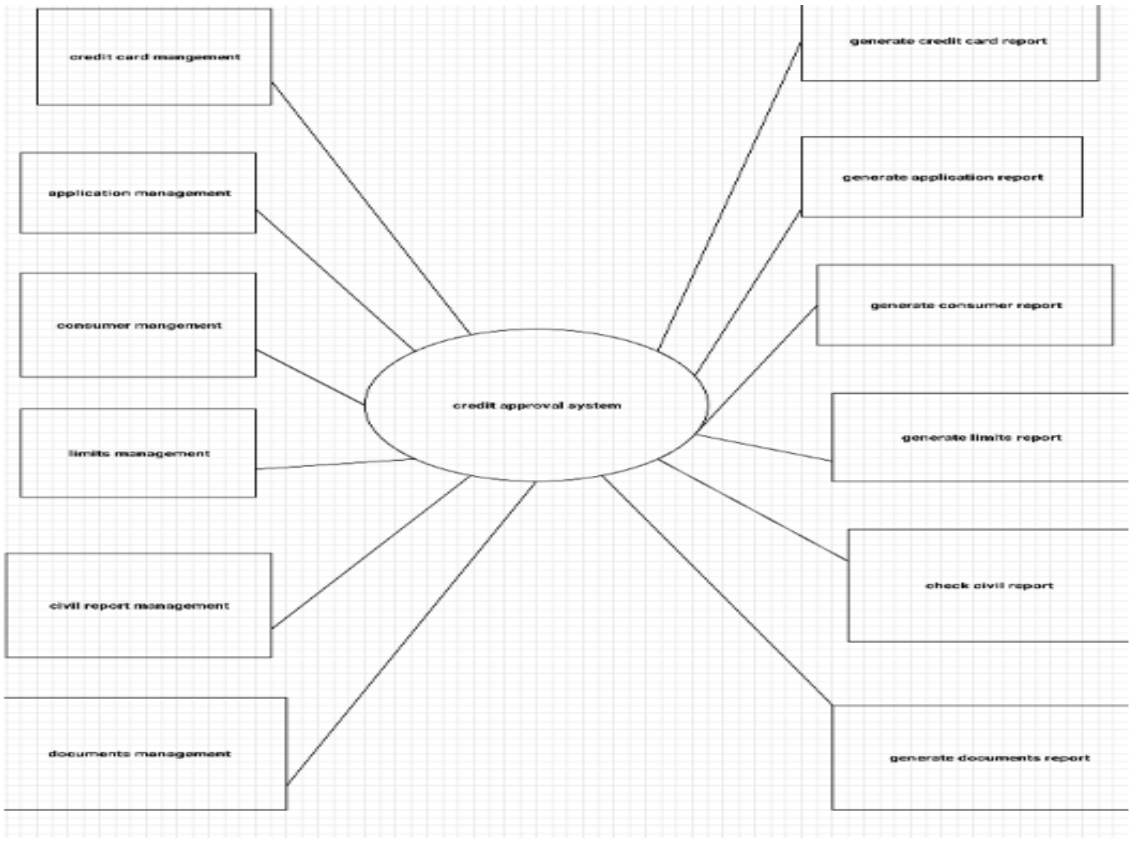


Figure 4.3: Level 1 Data Flow Diagram for credit card fraud detection

4.3 UML Diagrams

The Unified Modeling Language is a standardized modeling language enabling developers to specify, visualize, construct and document artifacts of a software system. UML diagram identifies functionalities of the system diagram is a diagram based on the UML to visually represent the system along with its main actors, roles, actions classes. UML is an important aspect involved in object-oriented software development. It uses graphic notation to create visual models of a software system.

4.3.1 Use Case Diagram

A use case diagram simply depicts the interaction between the system and entities external to the system. these entities are called actors which have a specific role in the system. the figure shows the use cases for the proposed system. It consists of actors like cardholder, payment gateway, merchants.

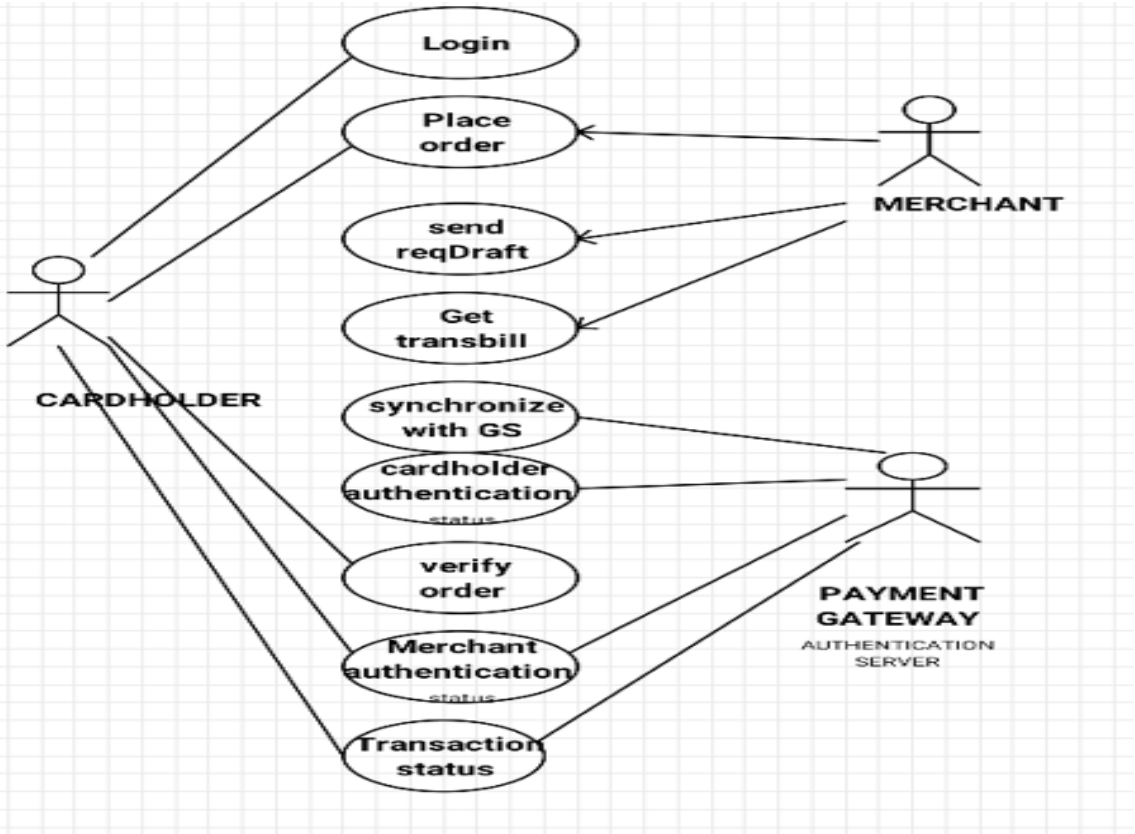


Figure 4.4: UseCase Diagram for credit card fraud detection

4.3.2 Class diagram

A class diagram is used to represent a static view of the system. It mainly uses the classes, interfaces and their relationship. figure shows the class diagram for the proposed system which contains classes like role class, permission class, document class, user class, consumer class, limit class, credit card class, etc.

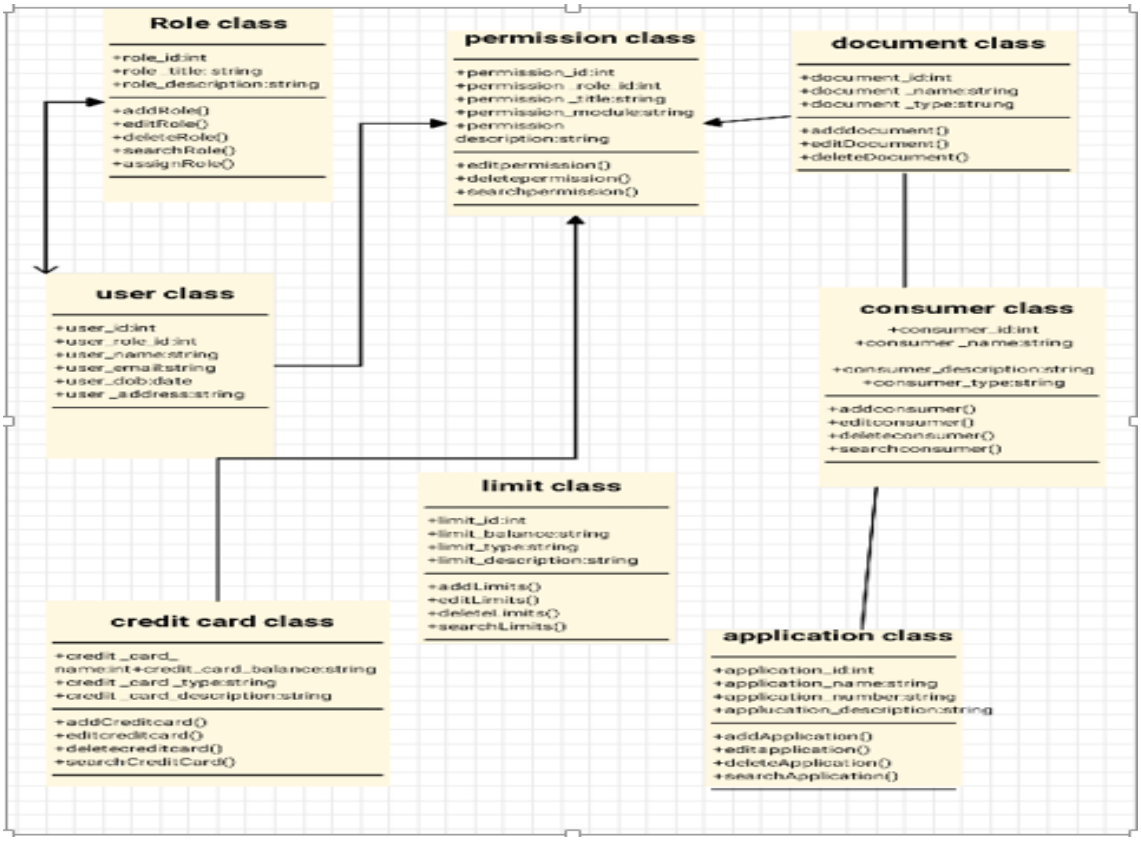


Figure 4.5: Class diagram for credit card fraud detection

4.3.3 Sequence diagram

A sequence diagram simply depicts interaction between object in sequential manner. Purpose of Sequence diagram is to show functionality. Figure shows the sequence diagram for purposed system which contain client , webpage, snapshot, file system, etc.

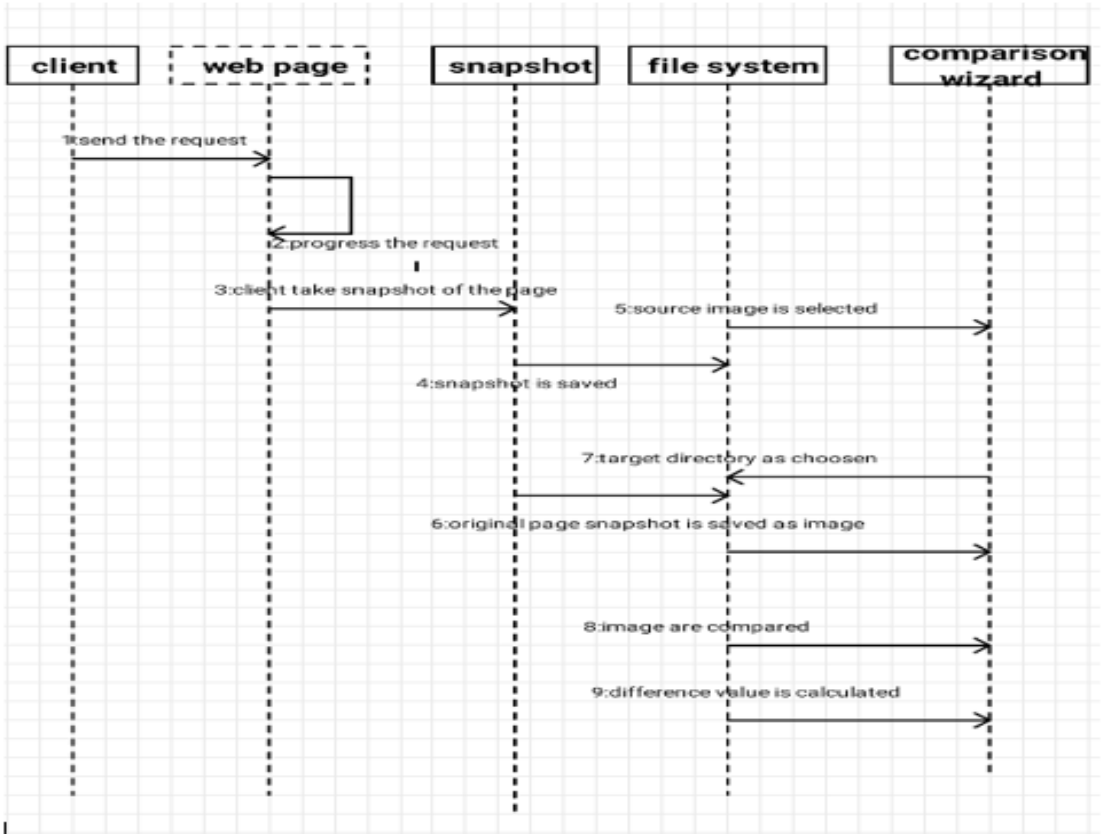


Figure 4.6: Sequence diagram For credit card fraud detection

4.3.4 State diagram

A state chart diagram is used to model the dynamic nature of a system. They dene different states of n object during its lifetime and those states are changed by events. State chart diagrams are useful to model the reactive system. the figure, shows the state diagram for a proposed system which contain the states like purchase, credit card information verification, etc.

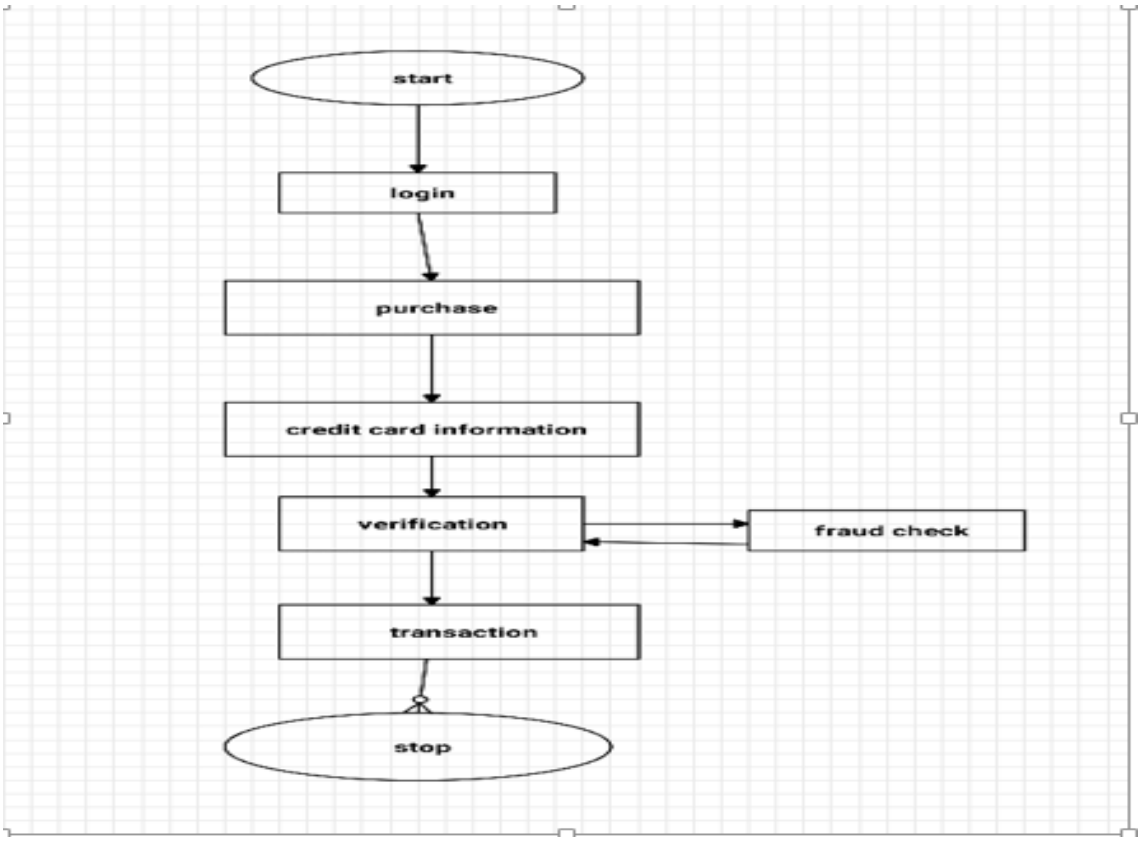


Figure 4.7: State Diagram for credit card fraud detection

4.3.5 Component Diagram

A component diagram depicts how components are wired together to form larger components or software systems they are used to illustrate the structure of arbitrarily complex systems fig diagram for the system which consists the components like credit card approval system, security, persistence, database connector.

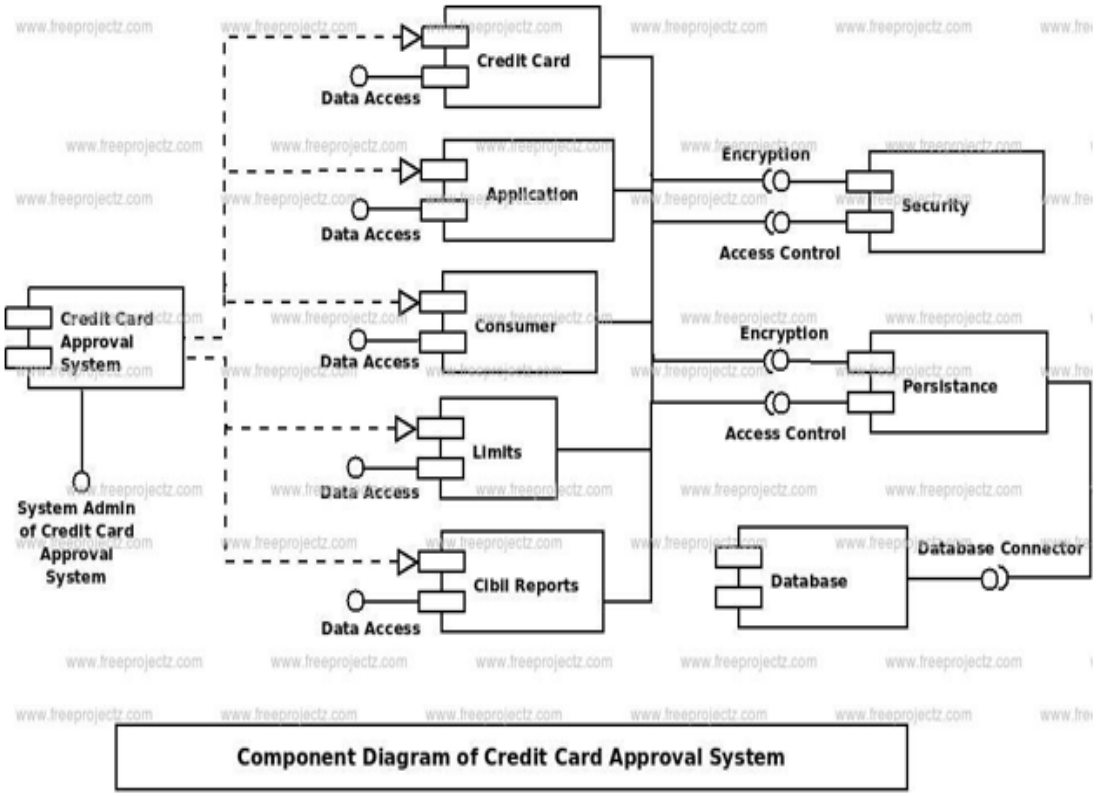


Figure 4.8: Component Diagram for credit card fraud detection

4.3.6 Deployment Diagram

A deployment diagram represents the configuration of runtime processing node and the components that live in them fig shows deployment diagram for a system which consist the node like legal pattern database, machine algorithm, frequent mining, legal pattern database.

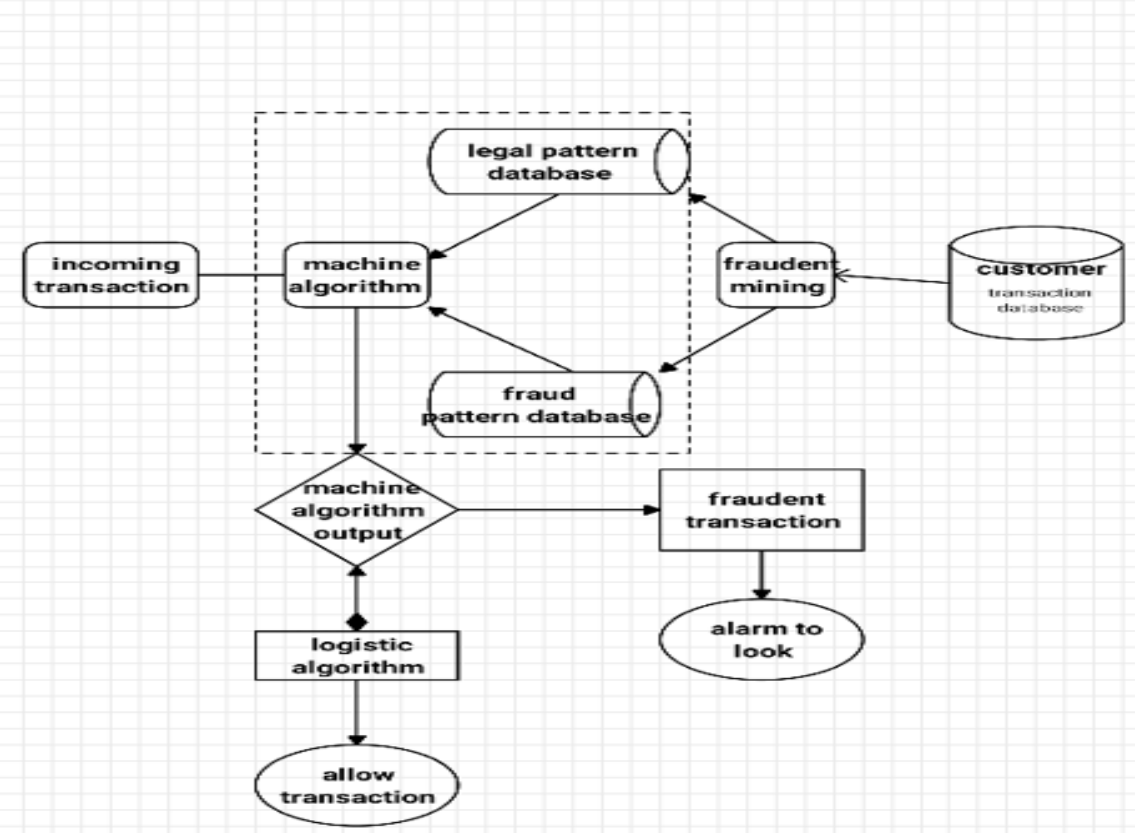


Figure 4.9: Component Diagram for credit card fraud detection

4.4 Summary

In this fourth chapter, the Design of credit card fraud detection is presented. In the next chapter, the Conclusion and Future Work are presented.

Chapter 5

Coding & Implementation

Important phase in system development is the successful implementation of the new system design. Implementation includes all those activities that take place to convert from the old system to the new system.Project Implementation is a practice of executing or carrying out a project under a certain plan in order to complete this project and produce desired results. Such a practice encompasses all processes and activities involved in getting the project plan fulfilled and accomplishing project goals and objectives. Implementation details, how system is implement in module wise. Executing the individual module and then combine to solution.

The organization of this Chapter is as follows. Section 5.1 describes Algorithm. The Software And Hardware For Development represented in Section 5.2. Finally, The Summary is described in the last Section 5.3.

5.1 Algorithm

1. In the real world, massive streams of payments are requested by the users.
2. So, in the existing system, these requests are analyzed by automatic tools that determine which transaction to authorize.
3. We are using a Logistic Regression machine-learning algorithm to analyze all the authorized transactions and report suspicious ones.
4. When users request any transaction request these requests will be scanned by the decision function.
5. The decision function will compare the transaction data with the output of the machine learning model.
6. If the transaction will be encountered to be legitimate then it will allow it. otherwise, it will alert the bank.
7. The system will automatically generate a report for a fraudulent transactions.
8. These reports are investigated by the professionals by actually contacting the card-holders to confirm the transaction was genuine or not.
9. The investigators provide feedback to the automated system.
10. These feedback is used to train and update the algorithm eventually to improve fraud detection performance.

5.2 Software & Hardware for development

Software and hardware give an idea about what are the necessary things that are needed for the proposed system, which plays a very important role in the development Of any system. This chapter deals with what are the hardware components that are needed for the system, application, software that is required for the development of the system and the functional requirement of the system. Front end tools helps to visualize the system while Back end helps in activities which are not visible to the end user.

Django is a widely used free, open-source, and high-level web development framework. It provides a lot of features to the developers "out of the box," so development can be rapid. Jupyter Notebook is Uses data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more And some technology are used data processing, dada handling, feature engoneering, onehot encoding, model bulding. Pandas, Numpy, Sklearn, Pinkle,Matploatlab Python Libraries are a set of useful functions that eliminate the need for writing codes from scratch. The portal is developed by using this Technology and languages. The proposed computerization is developed using MySQL as Backend and CSS, HTML anf Javascript as a Frontend. Main operating system required for making portal is Windows 10.

5.3 Coding

Standard coding practices are needed to ensure that the code is readable, understandable and easily modifiable. This project has defined standards and guidelines to be followed while pseudo coding. These standards were followed during the development of the application to produce code that is more consistent and to make code maintenance.

A. Logistic Regression

- Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

3. Creditcard Fraud Detection using Logistic Regression

July 13, 2021

[1]:
```
import numpy as np
import pandas as pd
import sklearn
import scipy
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report,accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn.svm import OneClassSVM
from pylab import rcParams
```

[2]:
```
creditcard_data = pd.read_csv('D://Users//gopal//Downloads//creditcard.csv')
```

[3]:
```
creditcard_data.head(10)
```

[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	

	V8	V9	...	V21	V22	V23	V24	V25	\
0	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	
1	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	
2	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	
3	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.175575	0.647376	
4	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.141267	-0.206010	
5	0.260314	-0.568671	...	-0.208254	-0.559825	-0.026398	-0.371427	-0.232794	
6	0.081213	0.464960	...	-0.167716	-0.270710	-0.154104	-0.780055	0.750137	
7	-3.807864	0.615375	...	1.943465	-1.015455	0.057504	-0.649709	-0.415267	

```
8  0.851084 -0.392048  ... -0.073425 -0.268092 -0.204233  1.011592  0.373205
9  0.069539 -0.736727  ... -0.246914 -0.633753 -0.120794 -0.385050 -0.069733

        V26       V27       V28  Amount  Class
0 -0.189115  0.133558 -0.021053  149.62      0
1  0.125895 -0.008983  0.014724    2.69      0
2 -0.139097 -0.055353 -0.059752  378.66      0
3 -0.221929  0.062723  0.061458  123.50      0
4  0.502292  0.219422  0.215153   69.99      0
5  0.105915  0.253844  0.081080    3.67      0
6 -0.257237  0.034507  0.005168    4.99      0
7 -0.051634 -1.206921 -1.085339   40.80      0
8 -0.384157  0.011747  0.142404   93.20      0
9  0.094199  0.246219  0.083076    3.68      0

[10 rows x 31 columns]
```

[4]:
```
creditcard_data.describe()
```

[4]:
```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean    94813.859575  1.165980e-15  3.416908e-16 -1.373150e-15  2.086869e-15
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   9.604066e-16  1.490107e-15 -5.556467e-16  1.177556e-16 -2.406455e-15
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05
mean   ...  1.656562e-16 -3.444850e-16  2.578648e-16  4.471968e-15
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01
```

```
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00

                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000
mean   5.340915e-16  1.687098e-15 -3.666453e-16 -1.220404e-16      88.349619
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000

              Class
count  284807.000000
mean        0.001727
std         0.041527
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000

[8 rows x 31 columns]
```

[5]:
```
creditcard_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
```

```
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

[6]:
```
creditcard_data.count()
```

[6]:
```
Time      284807
V1        284807
V2        284807
V3        284807
V4        284807
V5        284807
V6        284807
V7        284807
V8        284807
V9        284807
V10       284807
V11       284807
V12       284807
V13       284807
V14       284807
V15       284807
V16       284807
V17       284807
V18       284807
V19       284807
V20       284807
V21       284807
V22       284807
V23       284807
V24       284807
V25       284807
V26       284807
```

```
V27       284807
V28       284807
Amount    284807
Class     284807
dtype: int64
```

[7]:
```
creditcard_data.isnull().values.any()
```

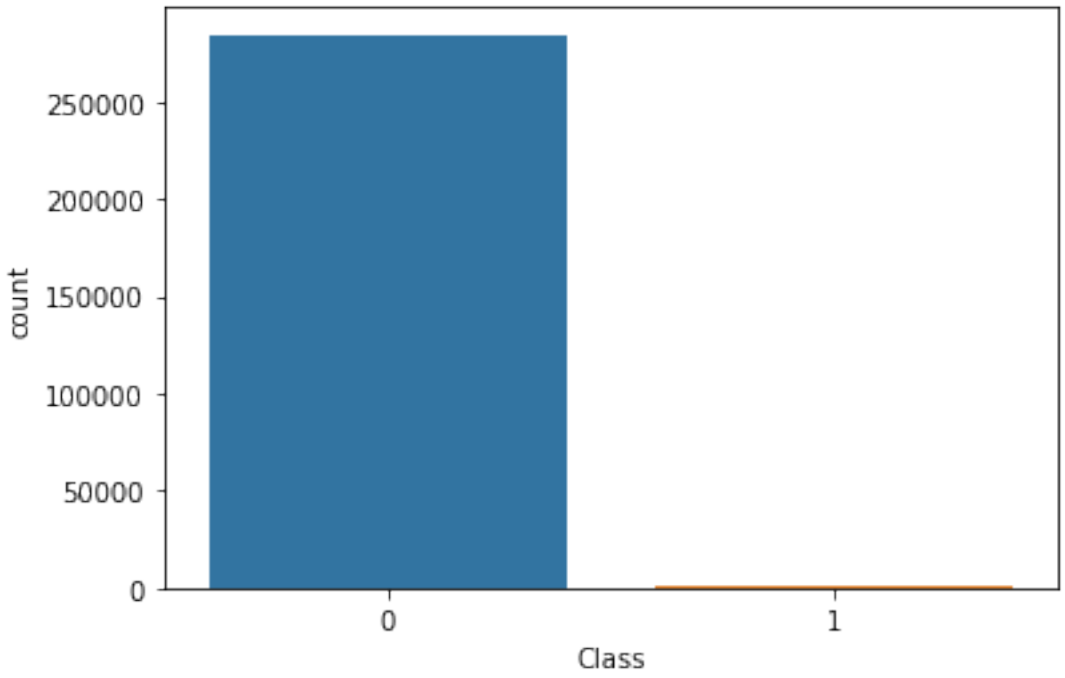
[7]:
```
False
```

[8]:
```
creditcard_data.isnull().sum()
```

[8]:
```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

[9]:
```
sns.countplot(x = 'Class',data = creditcard_data)
```

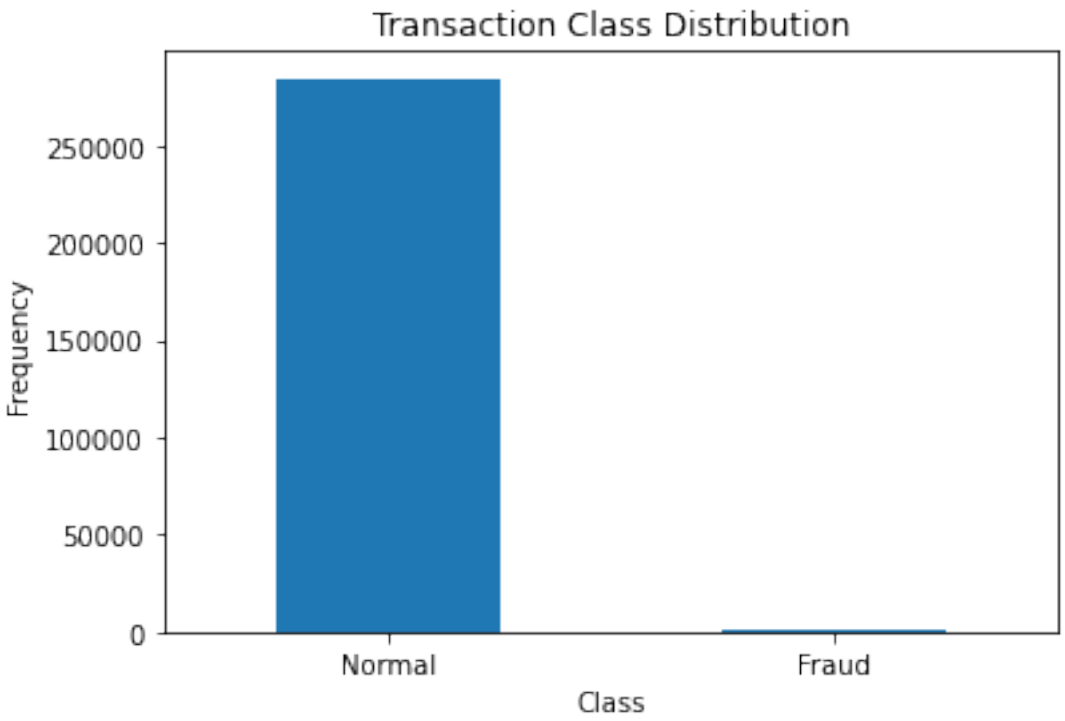
[9]: <AxesSubplot:xlabel='Class', ylabel='count'>



[10]:
```
LABELS = ["Normal", "Fraud"]
```

[11]:
```
count_classes = pd.value_counts(creditcard_data['Class'], sort = True)
count_classes.plot(kind = 'bar', rot=0)
plt.title("Transaction Class Distribution")
plt.xticks(range(2), LABELS)
plt.xlabel("Class")
plt.ylabel("Frequency")
```

[11]: Text(0, 0.5, 'Frequency')



[12]:
```
fraud = creditcard_data[creditcard_data['Class']==1]
normal = creditcard_data[creditcard_data['Class']==0]
```

[13]:
```
# Here 'shape' means number of columns in first row.
# 'fraud' variable prints the total number of fraud transactions,
# 'normal' variable prints total number of not fraud transactions.
```

[14]:
```
print(fraud.shape,normal.shape)
```

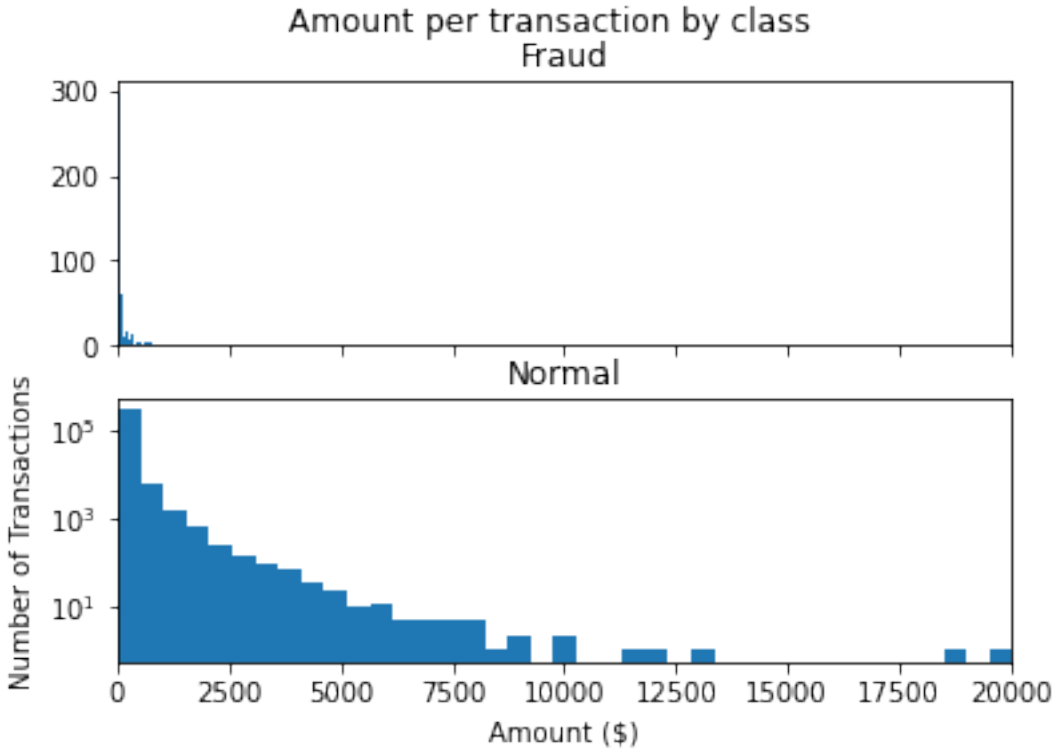
```
(492, 31) (284315, 31)
```

[15]:
```
normal.Amount.describe()
```

[15]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

[16]:
```
# Amount per transaction by class
```

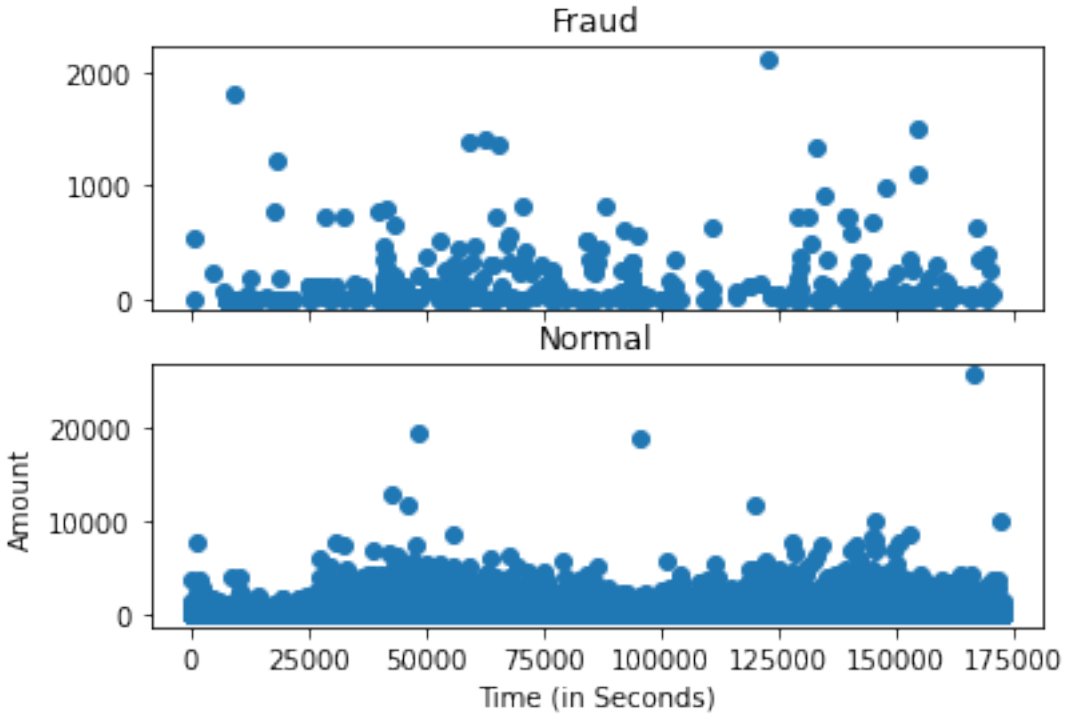
[17]:
```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
f.suptitle('Amount per transaction by class')
bins = 50
ax1.hist(fraud.Amount, bins = bins)
ax1.set_title('Fraud')
ax2.hist(normal.Amount, bins = bins)
ax2.set_title('Normal')
plt.xlabel('Amount ($)')
plt.ylabel('Number of Transactions')
plt.xlim((0, 20000))
plt.yscale('log')
plt.show();

# Through the below graph we can conclude that transaction amount is very small␣
 ↪for fraud transactions.
```



[18]:
```
# We Will check Do fraudulent transactions occur more often during certain time␣
 ↪frame.
```

[19]:
```
f, (ax1, ax2) = plt.subplots(2, 1, sharex=True)
ax1.scatter(fraud.Time, fraud.Amount)
ax1.set_title('Fraud')
```

```
ax2.scatter(normal.Time, normal.Amount)
ax2.set_title('Normal')
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amount')
plt.show()
```



[20]:
```
# We will store Independent variables in x.
```

[21]:
```
x = creditcard_data.drop('Class',axis = 1)
```

[22]:
```
x.head(10)
```

[22]:

	Time	V1	V2	V3	V4	V5	V6	V7	\
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	

```
         V8        V9  ...       V20       V21       V22       V23       V24  \
0  0.098698  0.363787  ...  0.251412 -0.018307  0.277838 -0.110474  0.066928
1  0.085102 -0.255425  ... -0.069083 -0.225775 -0.638672  0.101288 -0.339846
2  0.247676 -1.514654  ...  0.524980  0.247998  0.771679  0.909412 -0.689281
3  0.377436 -1.387024  ... -0.208038 -0.108300  0.005274 -0.190321 -1.175575
4 -0.270533  0.817739  ...  0.408542 -0.009431  0.798278 -0.137458  0.141267
5  0.260314 -0.568671  ...  0.084968 -0.208254 -0.559825 -0.026398 -0.371427
6  0.081213  0.464960  ... -0.219633 -0.167716 -0.270710 -0.154104 -0.780055
7 -3.807864  0.615375  ... -0.156742  1.943465 -1.015455  0.057504 -0.649709
8  0.851084 -0.392048  ...  0.052736 -0.073425 -0.268092 -0.204233  1.011592
9  0.069539 -0.736727  ...  0.203711 -0.246914 -0.633753 -0.120794 -0.385050

        V25       V26       V27       V28  Amount
0  0.128539 -0.189115  0.133558 -0.021053  149.62
1  0.167170  0.125895 -0.008983  0.014724    2.69
2 -0.327642 -0.139097 -0.055353 -0.059752  378.66
3  0.647376 -0.221929  0.062723  0.061458  123.50
4 -0.206010  0.502292  0.219422  0.215153   69.99
5 -0.232794  0.105915  0.253844  0.081080    3.67
6  0.750137 -0.257237  0.034507  0.005168    4.99
7 -0.415267 -0.051634 -1.206921 -1.085339   40.80
8  0.373205 -0.384157  0.011747  0.142404   93.20
9 -0.069733  0.094199  0.246219  0.083076    3.68

[10 rows x 30 columns]
```

[23]:
```
# We will store Dependent variables in y.
```

[24]:
```
y = creditcard_data.iloc[:, [30]]
```

[25]:
```
y.head(10)
```

[25]:
```
   Class
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
```

[26]:
```
# We will take some 60% of data to train the model.
# Remaining data we will use to test the model.
```

[27]:
```
from sklearn.model_selection import train_test_split
```

[28]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.
↪4,random_state = 1)
```

[29]:
```
# We take 170884 entries(rows) & 30 columns to train the model.
```

[30]:
```
x_train.shape
```

[30]: (170884, 30)

[31]:
```
y_train.shape
```

[31]: (170884, 1)

[32]:
```
x_train.head(10)
```

[32]:

	Time	V1	V2	V3	V4	V5	V6	\
160434	113377.0	2.055797	-0.326668	-2.752041	-0.842316	2.463072	3.173856	
101731	67929.0	-0.540285	-0.490724	2.448389	-1.645927	-0.686132	1.064138	
264482	161450.0	1.876534	-0.557387	-0.255089	0.516167	-0.772171	-0.355414	
136757	81854.0	1.143457	-0.276825	-0.797496	0.161368	0.581342	0.588723	
3213	2790.0	-0.678097	0.774112	1.069828	-2.205852	0.239830	-0.849156	
37830	39128.0	-1.987443	1.751958	0.680044	-1.470307	-0.946481	-0.922410	
134509	80837.0	0.978370	-0.659165	0.846408	0.853976	-1.213071	-0.119725	
250800	155071.0	-1.069186	0.106874	1.487987	-1.202796	-0.480794	-0.031617	
193500	130161.0	1.941252	-0.364879	-0.573095	0.329598	-0.318954	-0.313494	
99110	66966.0	-0.748939	-0.371579	1.532966	-2.301357	-1.673885	-0.851999	

	V7	V8	V9	...	V20	V21	V22	\
160434	-0.432126	0.727706	0.608606	...	-0.180370	0.269765	0.844627	
101731	-0.939464	0.545946	-0.001087	...	0.063702	0.202704	0.791300	
264482	-0.590483	-0.047078	1.113383	...	-0.021504	0.274268	0.909949	
136757	0.088451	0.176529	0.042890	...	-0.020403	-0.060396	-0.434133	
3213	0.966565	-0.140874	0.794798	...	0.112617	0.120434	0.770676	
37830	-0.268145	1.015104	0.033159	...	0.178375	-0.040819	-0.203584	
134509	-0.639207	0.282894	0.967720	...	-0.084760	0.135728	0.241543	
250800	-0.457501	0.598291	-1.304145	...	-0.378777	-0.554149	-1.459540	
193500	-0.312450	-0.133456	1.123576	...	-0.024409	-0.181526	-0.409657	
99110	-0.976504	0.374899	-1.998718	...	-0.526869	0.020154	0.231104	

	V23	V24	V25	V26	V27	V28	Amount
160434	0.020675	0.726212	0.366624	-0.398828	0.027735	-0.060282	1.00
101731	-0.443976	-0.788248	0.227620	-0.142376	0.142450	0.091109	3.70
264482	0.155183	1.237195	-0.253294	0.560668	-0.028017	-0.026609	59.90
136757	-0.311813	-1.724827	0.596453	0.484119	-0.071482	-0.013778	94.52
3213	-0.238002	0.216774	-0.016602	-0.834273	0.354532	0.034160	1.00

```
37830   0.056342  0.335927 -0.188718  0.721492  0.313551  0.222156    0.77
134509 -0.164153  0.339991  0.335235  0.491845 -0.034497  0.018901   99.99
250800  0.023364  0.585929 -0.004110  0.414684 -0.103067 -0.040118   23.75
193500  0.229508 -0.661213 -0.309630 -0.590878  0.033805 -0.026512   54.99
99110   0.204030  0.344760 -0.706139 -0.490673 -0.041364  0.097102   25.00

[10 rows x 30 columns]
```

[33]:
```
y_train.head(10)
```

[33]:
```
        Class
160434      0
101731      0
264482      0
136757      0
3213        0
37830       0
134509      0
250800      0
193500      0
99110       0
```

[34]:
```
# Determine the number of fraud and normal transactions in the train dataset.
```

[35]:
```
fraud_train = y_train[y_train['Class']==1]
normal_train = y_train[y_train['Class']==0]
```

[36]:
```
fraud_train.shape
```

[36]:
```
(306, 1)
```

[37]:
```
normal_train.shape
```

[37]:
```
(170578, 1)
```

[38]:
```
outlier_fraction = len(fraud_train)/float(len(normal_train))
```

[39]:
```
print(outlier_fraction)
```

```
0.0017939007374925253
```

[40]:
```
# Logitic Regression
```

[41]:
```
from sklearn.linear_model import LogisticRegression
```

[42]:
```
from sklearn.metrics import confusion_matrix
```

[43]:
```
model = LogisticRegression()
```

[44]:
```
model.fit(x_train, y_train.values.ravel())
```

```
d:\users\gopal\appdata\local\programs\python\python38-32\lib\site-
packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[44]:
```
LogisticRegression()
```

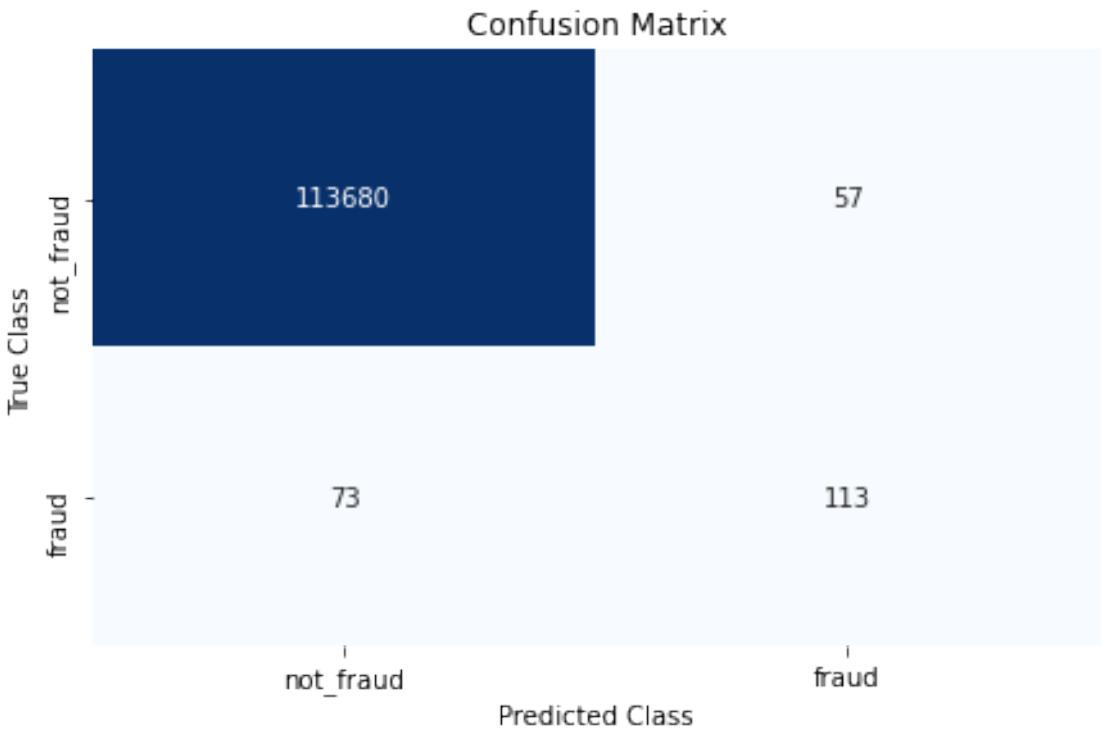
[45]:
```
pred = model.predict(x_test)
```

[ ]:
```
# Check Accuracy
```

[53]:
```
confusion_matrix(y_test,pred)
```

[53]:
```
array([[113680,     57],
       [    73,    113]], dtype=int64)
```

[54]:
```
class_names = ['not_fraud', 'fraud']
matrix = confusion_matrix(y_test, pred)
# Create pandas dataframe
dataframe = pd.DataFrame(matrix, index=class_names, columns=class_names)
# Create heatmap
sns.heatmap(dataframe, annot=True, cbar=None, cmap="Blues", fmt = 'g')
plt.title("Confusion Matrix"), plt.tight_layout()
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
plt.show()
```



[56]:
```
from sklearn.metrics import accuracy_score
```

[57]:
```
accuracy_score(y_test,pred)
```

[57]:
```
0.9988588783652116
```

[47]:
```
from sklearn.metrics import f1_score, recall_score
f1_score = round(f1_score(y_test, pred), 2)
recall_score = round(recall_score(y_test, pred), 2)
print("Sensitivity/Recall for Logistic Regression Model 1 : {recall_score}".
 ↪format(recall_score = recall_score))
print("F1 Score for Logistic Regression Model 1 : {f1_score}".format(f1_score =␣
 ↪f1_score))
```

```
Sensitivity/Recall for Logistic Regression Model 1 : 0.61
F1 Score for Logistic Regression Model 1 : 0.63
```

[ ]:
```
# Check efficiency
```

[58]:
```
from sklearn.metrics import classification_report
```

[59]:
```
classification_report(y_test,pred)
```

[59]:
```
'              precision    recall  f1-score   support\n\n           0
1.00      1.00      1.00    113737\n           1       0.66      0.61      0.63
186\n\n    accuracy                           1.00    113923\n   macro avg
0.83      0.80      0.82    113923\nweighted avg       1.00      1.00      1.00
113923\n'
```

[60]:
```
from sklearn.metrics import f1_score, recall_score
f1_score = round(f1_score(y_test, pred), 2)
recall_score = round(recall_score(y_test, pred), 2)
print("Sensitivity/Recall for Logistic Regression Model 1 : {recall_score}".
 ↪format(recall_score = recall_score))
print("F1 Score for Logistic Regression Model 1 : {f1_score}".format(f1_score =␣
 ↪f1_score))
```

```
Sensitivity/Recall for Logistic Regression Model 1 : 0.61
F1 Score for Logistic Regression Model 1 : 0.63
```

5.4 Summary

In this chapter, Detailed Coding of project is described. In the next chapter,Testing is explained.

Chapter 6

Testing

Testing is a set activity that can be planned and conducted systematically.Testing begins at the module level and work towards the integration of entire computers based system. For testing our software we test each and every path that user can go at any point in the lifetime of the system. Nothing is complete without testing, as it is vital success of the system. Errors and bugs do occur because of various factor like error in design of system, implementing the design of system, human error and much more, due to which testing becomes indispensable as application has to be error free, for customer satisfaction and develop the good will in market.

The organization of this Chapter is as follows. Section 6.1 Black Box Testing. Section 6.2 represents the Manual Testing. Section 6.3 describes the Test Cases Identification And Execution of the project. Finally, The Summary is described in the last Section 6.4.

6.1 Black Box Testing

The technique of testing in which the tester doesn't have access to the source code of the software and is conducted at the software interface without concerning with the internal logical structure of the software is known as black box testing. Black Box Testing is software testing method in which the internal structure/design /implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. The goal of the technique is to find errors in the following categories:-

1. Incorrect or missing functions.
2. Interface errors.
3. Errors in data structures or external database access.
4. Behavior or performance errors.

6.2 Manual Testing

Manual Testing is a type of Software Testing. Where Testers manually execute test cases without using any automation tools. Manual Testing is the most primitive of all testing types and helps find bugs in the software system. Any new application must be manually tested before its testing can be automated. Manual Testing requires more effort but is necessary to check automation feasibility. It does not require knowledge of any testing tool.

The steps in manual testing is as follows:

6.2.1 Understanding Documentation

The main reason behind creating test documentation is on either reduce or remove any uncertainties about the testing activities.it is also provides good marketing and sales strategy. Test documentation helps to improve transparency with the user. It helps to over a proper information regarding various scholarships to the user within specific time limits.

6.2.2 Draft Test Cases

Draft Test cases that cover all the requirement mention in the documentation. It required the explanation of how the system will be tested. Test cases that are as simple as possible and transparent. This makes the understanding the steps easy and test execution faster.

6.2.3 Review and Baseline

Review is the systematic examination of the documents. One or more people with the main aim of finding and removing errors easily. Baseline is a formal document which act as a base document for future work.

6.2.4 Report Bugs

Bug report should be clear and easily understandable. Once bugs are fixed, again execute the failing test cases to verify it.

6.3 Summary

In this chapter, Detailed Testing of project is described. In the next chapter, conclusion andfuture work of project is explained.

Chapter 7

Result & Discussion

1. The code prints out the number of false positives it detected and compares it with the actual values.
2. This is used to calculate the accuracy score and precision of the algorithms.
3. The fraction of data we used for faster testing is 60% of the entire dataset.
4. These results along with the detailed report of the algorithm is given in the output, where class 0 means the transaction was determined to be valid and 1 means it was determined as a fraud transaction.
5. This result matched against the class values to check for false positives.

Chapter 8

Conclusion & Future Work

Credit card fraud is without a doubt an act of criminal dishonesty. This project has listed out the most common methods of fraud along with their detection methods and reviewed recent findings in this field. This project 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 80% accuracy, its precision remains only at 28%. This high percentage of accuracy is to be expected due to the huge imbalance between the number of valid and number of genuine transactions.

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.

Bibliography

[1] Anderson M. (2008). From Subprime Mortgages to Subprime Credit Cards‘. Communities and Banking, Federal Reserve Bank of Boston, pp. 21-23.

[2] Answer ET AL. (2009-2010). Online Credit Card Fraud Prevention System for Developing Countries‘, International Journal of Reviews in Computing, ISSN: 2076-3328, Vol. 2, pp. 62-70.

[3] Arias, J.C. & Miller R. (2009). Market Analysis of Student about Credit Cards‘. Business Intelligence Journal, Vol. 3, No. 1, pp. 23-36.

[4] Bhusari V. & Patil S. (2011). Study of Hidden Markov Model in Credit Card Fraudulent Detection‘. International Journal of Computer Applications, Vol. 20, No. 5, pp. 33-36.

[5] Delamaire et al. (2009) ‘Credit Card Fraud Detection Techniques: A Review’, Banks and banks Systems, Vol. 4, Issue 2, pp. 57-68.

[6] Dharwa J.N. & Patel A.R. (2011). A Data mining with Hybrid Approach Based Transaction Risk Score Generation Model (TRSGM) for Fraud Detection of Online Financial Transaction‘. International Journal of Computer Applications, Vol. 6, No. 1, pp. 18-25.

[7] Erdem Cumhur, (2008). Factors Affecting the Portability of Credit Card Default and the Intension of Card use in Turkey‘. International Research Journal of Finance and Economics. Issue 18, ISSN: 1450-2887, pp. 159-171.

[8] Ghosh S. & Reilly D. L. (1994) Credit card fraud detection with a neural-network‘. System Sciences, ISBN: 0-8186-5090-7, pp. 621 - 630.

[9] Rani J.K. et al (2011). Credit Card Fraud Detection Analysis‘, International Journal of Computer Trends and Technology, Vol. 2, Issue 1, pp. 24-27.