

BANK FRAUD DETECTION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements

for the award of

Bachelor of Engineering degree in Computer Science and Engineering

by

MATTA SUNIL CHANDU (Reg. No. 37110441)

MANNURU VENKATESH (Reg. No. 37110433)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI,**

CHENNAI – 600 119

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **MATTA SUNIL CHANDU (Reg. No. 37110441)** and **MANNURU VENKATESH (Reg. No. 37110433)** who carried out the project entitled "**BANK FRAUD DETECTION USING MACHINE LEARNING ALGORITHM**" under my supervision from **August 2020 to March 2021**.

Internal Guide

Mrs. Ramya.G.Franklin,M.E.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **MANNURU VENKATESH** hereby declare that the Project Report entitled **“BANK FRUAD DETECTION USING MACHINE LEARNING ALGORITHM”** is done by me under the guidance of **Mrs. RAMYA.G.FRANKLIN.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs. Ramya.G.Franklin.M.E., Assistant Professor** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The banking sector is a very important sector in our present day generation where almost every human has to deal with the bank either physically or online. In dealing with the banks, the customers and the banks face the chances of being trapped by fraudsters. Examples of fraud include insurance fraud, credit card fraud, accounting fraud, etc. Detection of fraudulent activity is thus critical to control these costs. This paper hereby addresses bank fraud detection via the use of machine learning techniques; association, clustering, forecasting, and classification to analyze the customer data in order to identify the patterns that can lead to frauds. Upon identification of the patterns, adding a higher level of verification/authentication to banking processes can be added.

TABLE OF CONTENTS

ABSTRACT	v
LIST OF FIGURES	viii

CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
	1.1 OUTLINE OF PROJECT	2
	1.2 AIM AND PURPOSE	4
2.	AIM AND SCOPE	5
	2.1 AIM OF PROJECT	5
	2.1.1 OBJECTIVES	5
	2.2 SCOPES	6
	2.2.1 ADVANTAGES	7
3.	SYSTEM DESIGN & METHODOLOGY	9
	3.1 EXISTING SYSTEM	9
	3.2 SYSTEM CAPABILITIES	9
	3.3 MULTIPLE PROTECTION LAYERS	10
	3.4 SYSTEM FUNCTIONALITIES	11
	3.5 NON FUNCTIONALITIES	12
	3.6 MODULES	13
	3.6.1 ADMIN	13
	3.6.2 DATA COLLECTION	13
	3.6.3 MACHINE LARNING TECHNIQUE	13
4.	SOFTWARE DEVELOPMENT METHODOLOGY	14
	4.1 DESCRIPTION OF DIAGRAM	16

4.3 DATA FLOW DIAGRAM	18
4.4 FRAUD DETECTION	18
5. RESULTS AND DISCUSSION	22
6. CONCLUSION AND FUTURE WORK	23
REFERENCES	24
APPENDIX	
A. PAPER ACCEPTANCE MAIL	26
B. PLAGIARISM REPORT	31
C. JOURNAL PAPER	39
D. SOURCE CODE	56

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
4.1	DESCRIPTION DIAGRAM	15
4.2	ARCHITECTURAL DESIGN	16
4.3	DATA FLOW DIAGRAM	18

CHAPTER 1

INTRODUCTI

ON

1.INTRODUCTION

According to The American Heritage dictionary, second college edition, fraud is defined as a deception deliberately practiced in order to secure unfair unlawful gain. Fraud detection is the recognition of symptoms of fraud where no prior suspicion or tendency to fraud exists. Examples include insurance fraud, credit card fraud and accounting fraud. Data from the Nigeria Inter-Bank Settlement System (NIBSS) has revealed that fraudulent transactions in the banking sector at its peak.

Fraud has evolved from being committed by casual fraudsters to being committed by organized crime and fraud rings that use sophisticated methods to take over control of accounts and commit fraud. Some 6.8 million Americans were victimized by card fraud in 2007, according to Javelin research. Such fraud on existing accounts accounted for more than \$3 billion in losses in 2007.

The Nilson Report estimates the cost to the industry to be \$4.84 billion. Javelin estimates the losses at more than six times that amount – some \$30.6 billion in 2007. Of course, fraud is not a domestic product as it's everywhere. For instance, card fraud losses cost UK economy GBP 423 million in 2006. Credit card fraud accounts for the biggest cut of the \$600 million that airlines lose each year globally.

This case study is aimed to demonstrate how you can obtain a forecast for fraudulent card transactions with a 93.5% (or higher) accuracy rate using Java 8 and Apache Spark. Apache Spark isn't the only Big Data framework you can use to create a robust credit card fraud detection algorithm.

Distributed computing, in this case, will massively speed up the entire process and allow you to conduct simultaneous calculations for different types of problems. Other technologies will not allow to obtain such speed.

Fraud continues to be of major concern to business companies, not-for-profit organizations and governmental agencies. Recent surveys by leading accounting firms document that fraud is costing these organizations billions of dollars per year (BDO 2008; KPMG 2008; Standards Australia 2008). Furthermore, fraud means reduced macroeconomic output. Estimates indicate that fraud costs the Australian economy up to 3 billion dollars each year (Standards Australia 2008). The incidence and financial impact of fraud seems to be steadily increasing and many organizations are ill-prepared to prevent and detect fraud (KPMG 2008). Australian Standard AS 8001-2008 (Standards Australia 2008) proposes that organizations implement a fraud detection program to quickly identify instances of fraud should preventive measures and internal controls fail. It recommends the development of systems for targeted post-transactional review and strategic use of computer systems including effective data mining and real-time transaction assessment to identify suspect fraudulent transactions. In a similar vein, PCAOB Auditing Standard No. 5 stresses the responsibility of external auditors to conduct a fraud risk assessment in planning and performing the audit of internal control over financial reporting, and to consider deficiencies in controls to prevent and detect fraud when assessing the risk of material misstatement in the financial statements (PCAOB 2007). Few information technology innovations have had as much impact on business organizations in recent years as enterprise systems (sometimes known as enterprise resource planning systems or ERP) (Rikhardsson & Kraemmergaard 2006). Enterprise systems are off-the-shelf applications that offer a comprehensive set of functionalities supporting and integrating most business processes, including accounting, sales, purchasing and production in a single system architecture. An enterprise system has several distinctive characteristics (Norris et al. 1998):

- Multi- functional in scope – it tracks financial results (dollars), procurement (material), sales (people and goods) and manufacturing (people and resources);
- Integrated in nature, that is, when a piece of data is entered regarding one of the functions, data relevant to other functions is changed;
- Modular in structure, that is, it can be used in a way that is as expansive or narrow as an organization chooses.

Modern enterprise systems are web enabled, which can mean browser based user interfaces, standardised data exchange and web-based reporting. It has been estimated that organizations worldwide spend approximately \$18.3 billion US every year on enterprise systems

(Shanks et al. 2003). These systems have significant

implications for accounting and auditing in general and fraud control in particular (ITGI 2006; Bae & Ashcroft 2004, Chapman & Chua 2003; Journal of Digital Forensics, Security and Law, Vol. 4(1) 41 Chapman 2005). However, enterprise systems are not typically utilised for fraud detection, certainly not in a systematic manner. These systems increase the complexity of the accounting and auditing environment but also offer new opportunities for improvements in effectiveness and efficiency of these processes (Spathis 2006). Enterprise systems offer functionalities for continuous monitoring of controls and detecting fraudulent transactions. One such functionality is audit trails. This paper illustrates how audit trails in enterprise systems can be used for continuous fraud detection. It discusses continuous assurance and fraud detection and links these processes to enterprise systems. It explains the concept of audit trails and how they can be used for fraud detection within the context of a specific enterprise system solution – i.e. mySAP, which is described below, is a product of the German company SAP. It then proposes a methodology for continuous fraud detection that utilises various audit trails available in enterprise systems, namely security audit logs, changes in master records and accounting audit trails. This methodology is comprised of two stages:

(1) threat monitoring, involving highlevel surveillance of security audit logs for ‘red flags’, and (2) automated extraction and analysis of data from audit trails to document user actions. At that point, forensic investigation is used to determine whether a financial fraud

1.1 OUTLINE OF PROJECT

Considering that this is a case study, the software program was written for one interesting task from Kaggle. We have a dataset of credit card transactions conducted in Europe during two days in September 2014. The initial data is given as a .csv file containing 31 variables within 284.807 transactions. This dataset may seem nice, but for a real-life experiment it isn’t well-balanced. The percent of fraudulent transactions is only 0.172% of the total number. Yet to make a viable forecast you will need to obtain the highest possible True Positive Rate (TPR) for the classified data and the lowest False Positive Rate (FPR) possible.

This dataset may seem nice, but for a real-life experiment it isn’t well-balanced.

The percent of fraudulent transactions is only 0.172% of the total number. Yet to make

a viable forecast you will need to obtain the highest possible True Positive Rate (TPR) for the classified data and the lowest False Positive Rate (FPR) possible.

If you have similar unbalanced data at hand even the smallest amount of FPR can actually stand for a significant chunk of false positive results within the classification model. Meaning, you will end up verifying a large number of positive cases, which got labelled as fraud instead. Surely, that's not what you aim for. The good news is that the problem isn't new and there are some good solutions to that. In most cases, you can solve it by applying the stratified sampling, so that each stratum is represented by data of the same size.

In this Bank fraud detection you should also pay attention to the fact that the mean values are zero for the main components, but their variables are different. That's why we need to avoid the scale effect and standardize the variables. First we will get the input from the user and the new user should sign in as the user is new to the site and by giving all of the input details such as email, pan card details and other details related to the account now the user will get the login and now the user can access the banking page and now the main task begins where the user starts the payment and while the user is doing the payment the user will be asked to answer the security questions where if the user is a true user he will be able to give the correct answer to the security question.

1.2 AIM AND PURPOSE

In the first approach, we increased the Fraudulent dataset sampling to the size of Honest sampling using the mechanical reproduction of the samples and getting missing observations using the random non-repeat selection process. In the second one, we accounted only for those transactions, which are situated close to the separation zone between the studied classes. In simple words – only those fraudulent transactions, which were closely alike to the honest ones. If you need to classify your data into groups and generate random samplings for anomaly detection with Apache Spark, you can use a couple of approaches. For instance, with Dataset class or with the help of SQL. Here we are dealing with binary classifiers. So our job is to build a model that would estimate the function, which can

evaluate the differences between the classes, and then set a threshold that separates Class 0 from Class 1. One of the cool Spark use cases is that you can generate ROC curves rather efficiently. First, you will need to obtain an instance of Binary Logistic Regression Summary class, and then calculate the coordinate points on the curve. When developing a classifier for identifying the unlikely events, you need to aim for the high value of TRP) and low value of FPR. As the tabs above demonstrate – as the threshold increases, the number of identified fraudulent transactions reduces. Yet, you also reduce the FPR number. frauds are increasing heavily because of fraud financial loss is increasing drastically. Now days Internet or online transaction growing as new technology are coming day by day. In these transaction Credit card holds the maximum share. In 2018 Credit card fraud losses in London estimated US dollar 844.8 million. any models are implemented for fraud detection. In every model different algorithm are used. Detection of credit card fraud for new frauds will be problematic if new data has drastic changes in fraud patterns. Replacing the model is risky as machine learning algorithm take much time for training rather than predicting. The procedure which we followed to predict the result are understanding problem statement and data by performing statistical analysis and visualization then checking whether the data is balance or not, In this data set the data is imbalanced, balanced by using oversampling, then scaling the data using standardization and normalization and testing data with different ML algorithms For any data science project some package are very important such as Numpy that is numeric pandas and for visualization of the data. These details specify the bank number and the account number to be paid through bank transfer. The perpetrator switches these details to their own bank account or that of an accomplice. An invoice (often a duplicate) is entered for payment, and is subsequently paid automatically by the system (possibly without the involvement of the perpetrator). The banking details are then switched back to their original form. This is referred to as ‘flipping bank details’. The respective vendor does not receive the duplicate payment and is therefore not aware of the fraud. Auditors may sample the invoice and payment, but will find them apparently genuine. Tests for duplicate invoices and payments may detect this fraud.

CHAPTER 2

AIM AND SCOPE

2.1 AIM OF PROJECT

The primary aim of project's to show how to detect the fraud in the banking using some security questions given by the user as a inputs. Fraud detection is a challenging task for many banks and online payment companies. In today's life ,hackers are more convenient to get personal information and powerful password decoding tools to do online fraud. Online transaction fraud causes a loss around billions of dollars for customers per year.Hence, many financial and online transation companies devoted many dollars to develop fraud detection algorithms. Data mining, data warehouse, big data and machine learning are need.

2.1.1 OBJECTIVES

The main aim of fraud detection systems is to detect fraud accurately and before fraud is committed. The goal is to detect and accurate false fraud detection. There are several methodologies for detecting credit card fraud, like neural networks, Genetic Algorithms, k-means clustering. fraud has been a major issue in sectors like banking, medical, insurance, and many others. Due to the increase in online transactions through different payment options, such as credit/debit cards, PhonePe, Gpay, Paytm, etc., fraudulent activities have also increased. Moreover, fraudsters or criminals have become very skilled in finding escapes so that they can loot more. Since no system is perfect and there is always a loophole them, it has become a challenging task to make a secure system for authentication and preventing customers from fraud. So, Fraud detection algorithms are very useful for preventing frauds. Large amounts of such tagged data are fed into the supervised learning model in order to train it in such a way that it gives a valid output. Also, the accuracy of the model's output depends on how well-organized your data . The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data

from a written or printed document or it can occur by having people keying the data directly into the system.

2.2 SCOPE

Unsupervised learning models are built to detect unusual behavior in transactions which is not detected previously. Unsupervised learning models involve self-learning that helps in finding hidden patterns in transactions. In this type, the model tries to learn by itself, analyzes the available data, and tries to find the similarities and dissimilarities between the occurrences of transactions. This helps in detecting fraudulent activities.

So, both these models, supervised and unsupervised, can be used independently or in combination for detecting anomalies in transactions. In the rule-based approach, the algorithms are written by fraud analysts. They are based on strict rules. If any changes have to be made for detecting a new fraud, then they are done manually either by making those changes in the already existing algorithms or by creating new algorithms. In this approach, with the increase in the number of customers and the data, human effort also increases. So, the rule-based approach is time-consuming and costly. Another drawback of this approach is that it is more likely to have false positives. This is an error condition where an output of a test specifies the existence of a particular condition that does not even exist. The output of a transaction depends upon the rules and guidelines made for training the algorithm for non-fraudulent transactions. So, for a fixed risk threshold, if a transaction is rejected where it should not be, it will generate a condition of high rates of false positives. This false-positive condition will result in losing genuine customers. In the rule-based approach, the algorithms cannot recognize the hidden patterns. Since they are based on strict rules, they cannot predict fraud by going beyond these rules. But in real world, fraudsters are very skilled and can adopt new techniques every time to commit a crime. Therefore, there is a need for a system that can analyze patterns in data and predict and respond to new situations for which it is not trained or explicitly programmed.

Hence, we use Machine Learning for detecting fraud. Here, a machine tries to learn by itself and becomes better by experience. Also, it is an efficient way of detecting fraud because of its fast computing. It does not even require the guidance of a fraud analyst. It helps in reducing false positives for transactions as the patterns are detected by an automated system for streaming transactions that

are in huge

volume. Fraud detection has been usually seen as a data mining problem where the objective is to correctly classify the transactions as legitimate or fraudulent. For classification problems many performance measures are defined most of which are related with correct number of cases classified correctly. A more appropriate measure is needed due to the inherent structure of credit card transactions. When a card is copied or stolen or lost and captured by fraudsters it is usually used until its available limit is depleted. Thus, rather than the number of correctly classified transactions, a solution which minimizes the total available limit on cards subject to fraud is more prominent. Since the fraud detection problem has mostly been defined as a classification problem, in addition to some statistical approaches many data mining algorithms have been proposed to solve it. Among these, decision trees and artificial neural networks are the most popular ones. The study of Bolton and Hand provides a good summary of literature on fraud detection problems. However, when the problem is approached as a classification problem with variable misclassification costs as discussed above, the classical data mining algorithms are not directly applicable; either some modifications should be made on them or new algorithms developed specifically for this purpose are needed. An alternative approach could be trying to make use of general purpose meta heuristic approaches like genetic algorithms.

2.2.1 ADVANTAGES

In near future, the pervasive plans for new fraud prevention will include stronger caller authentication for customers that phone call centers; enterprise fraud detection that manages fraud across customer channels and accounts; and a case management system for managing fraud. This would lead to better protection against fraud at the point and time of transaction, and more accurate detection of incidents of fraud in completed transactions. Driven by metadata, all records are exhaustively linked based on combinations of attributes within the data. Then, using statistical techniques, common entities are identified and collapsed to produce single views of entities within networks. Discrete, bounded networks within the data are also generated, representing statistically relevant groups of activities and relationships. New generation banking fraud system goes beyond the typical customer view to provide a holistic view of fraudulent activity – including related perpetrators and unrelated channels – and enables a much deeper understanding of customer behavior. Each transaction – account opening, ATM access, an online banking transaction, call center encounter, etc. – is passed through a set of rules and predictive models. To eliminate real

time fraud to the lowest level. To increase the confidence of customers in the banking system especially for online transactions. To discourage fraudsters (both present and intending ones). In case of bank fraud detection the existing system is detect the fraud after fraud has been happen. Existing system maintain the large amount of data when customer comes to know about inconsistency in transaction he/she made complaint and then fraud detection system start it working. It first tries to detect that fraud has actually occur after that it transactions that was used to fraud detection mechanism developed by master and visa cardsA machine learning paradigm classification, with Bank Fraud Detection being the base. Intrusion detections to track fraud location and so on. In case of existing system there is no confirmation of recovery of fraud and Customer satisfaction. Secure electronic system used to analyze the behaviour of legitimate users. Data Mining mechanisms to classify and preprocess the users data. Genetic algorithms. In many models are implemented for fraud detection. In every model different algorithm are used. Detection of credit card fraud for new frauds will be problematic if new data has

drastic changes in fraud patterns. Replacing the model is risky as machine learning algorithm take much time for training rather than predicting. In Logistic Regression algorithm (LR) is implemented to sort the classification problem. Using Gaussian Mixture Models fraudulent cases are discretized. To balance data synthetic minority oversampling is used. Sensitivity analysis is used calculate economic value.

Dataset contains the transaction from Europe card owners during September 2013. In this 492 out of 2,84,807 are fraud transactions. Data is not balanced because less amount of fraud cases as compared to huge transaction data. Dataset is converted PCA transformation and contains only numeric values. Due to privacy and confidentiality many background information is not provided only PCA transformed data is given. Only time and amount are not transformed to PCA all other given values v1, v2, v3 v28 are PCA transformed numeric values. Feature class contains 1 for fraud and 0 for normal transaction.

CHAPTER 3

SYSTEM DESIGN AND METHODOLOGY

3.1 EXISTING SYSTEM

In case of bank fraud detection the existing system is detect the fraud after fraud has been happen. Existing system maintain the large amount of data when customer comes to know about inconsistency in transaction he/she made complaint and then fraud detection system start it working. It first tries to detect that fraud has actually occur after that it transactions that was used to fraud detection mechanism developed by master and visa cards.A machine learning paradigm classification, with Bank Fraud Detection being the base. Intrusion detections to track fraud location and so on. In case of existing system there is no confirmation of recovery of fraud and Customer satisfaction.Secure electronic system used to analyze the behaviour of legitimate users.Data Mining mechanisms to classify and preprocess the users data.Genetic algorithms.

3.2 SYSTEM CAPABILITIES

Fraud detection and prevention system is the core of any fraud risk management strategy. Teams choose software with functionality that works best for their workflow and business needs in general.In our whitepaper, we compared machine learning- based systems with rule-based ones and described how ML-based solutions help prevent and identify fraudulent activity across several industries.For this article, we contacted specialists from NoFraud and SAS to discuss the purposes and capabilities of anti-fraud software and get their advice on the solution choice. The final section of the article contains descriptions of several solutions available on the market.

3.3 MULTIPLE PROTECTION LAYERS

Fraud analyst Avivah Litan from Gartner Group has suggested a five-layer approach to fraud detection and prevention. Each of the levels represents a specific type of customer activity and behaviour.

- Level 1 is endpoint-centric and includes user authentication, the device they are using for the transaction, as well as geolocation.
- particular session is analyzed for anomalies.
- Level 3, or channel-centric, considers analyzing account activity for anomalies.
- Level 4 is cross-product, cross-channel, and entails monitoring entity behavior across channels and bank products.
- Level 5 – entity link analysis – is about evaluating connections between various users or transactions.

3.4 SYSTEM FUNCTIONALITIES

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that it has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information. These predictions create political appetite for the rapid development of the AI industry,³ and healthcare is a priority area where this technology has yet to be exploited.^{2 3} The digital health revolution described by Duggal et al ⁴ is already in full swing with the potential to 'disrupt' healthcare.

3.5 NON FUNCTIONAL

REQUIREMENTS Portability

It is the usability of the same software in different environments. The project can be run in any operating system. Regardless of the medium used to learn probability, be it books, videos, or course material, machine learning practitioners study probability the wrong way.

Performance

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system. In Machine Learning it is key to be able to correctly evaluate the model being produced to guarantee that the predictions are accurately describing the intended phenomenon (disease prediction, future cost estimation, etc.).

Accuracy

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective. Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Maintainability

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability. Algorithms are left to their own devices to discover and present the interesting structure in the data.

3.6 MODULES

3.6.1 ADMIN

- Admin module allows system administrator to set up back-end of the system and perform basic system configuration, mainly definition of predefined drop-down fields, definition of classes time schedule, etc. All the new packages and promo bundles as well as new prices and price types for classes, new subjects offered, etc. are defined here. All data is labeled and the algorithms learn to predict the output from the input data. Some data is labeled but most of it is unlabeled and a mixture of supervised and unsupervised techniques can be used. All data is labeled and the algorithms learn to predict the output from the input data.

3.6.2 DATA COLLECTION

Benchmarking. Benchmarking is a technique by which an organization compares its actual or planned practices, to those of comparable organizations.

Benchmarking can be used to identify best practices, generate ideas for improvement and provide a basis for measuring performance.

3.6.3 MACHINE LEARNING TECHNIQUE

The process of an algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers, the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance. Unsupervised learning is where you only have input data (X) and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data. These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devices to discover and present the interesting structure in the data.

CHAPTER 4

SOFTWARE DEVELOPMENT METHODOLOGY

4.1 DESCRIPTION OF DIAGRAM

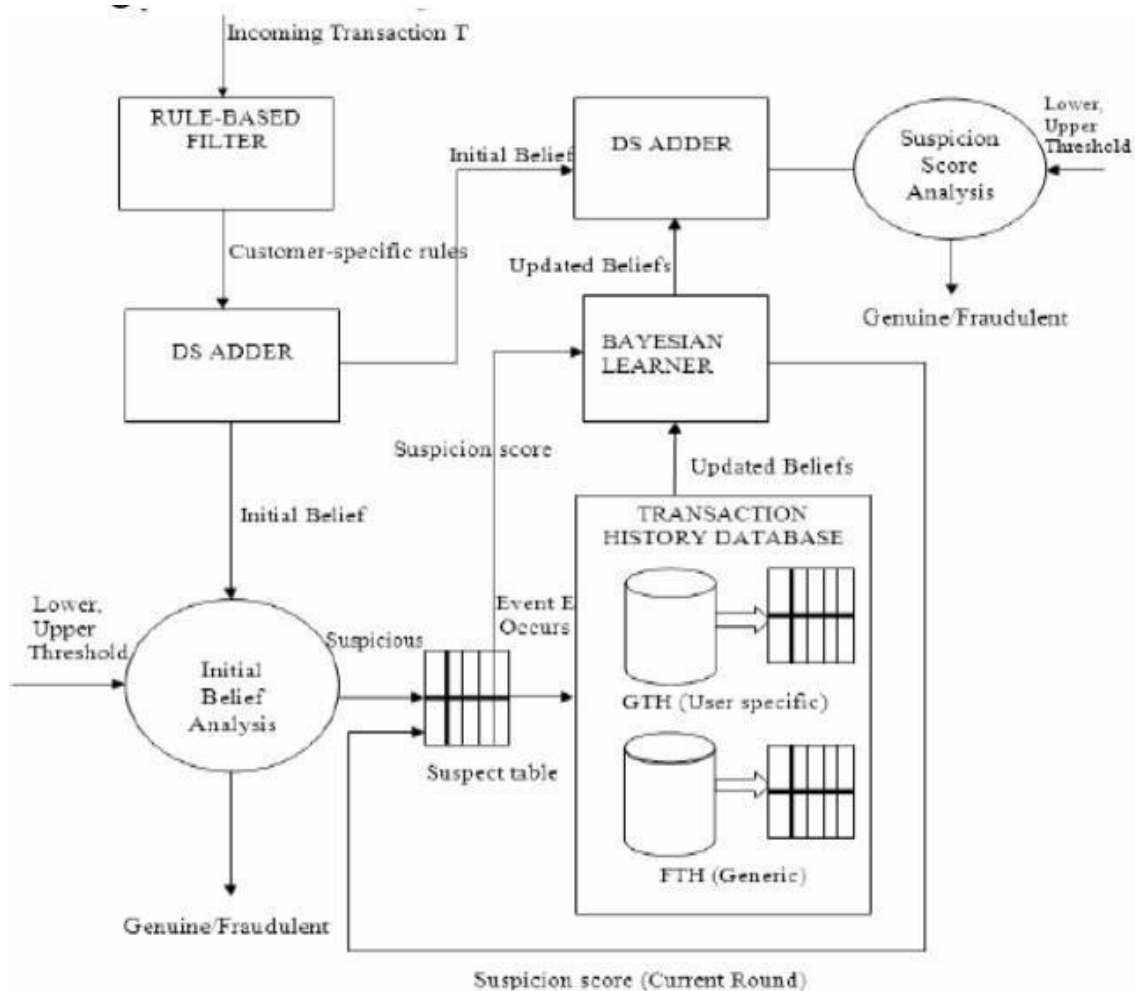


Fig 4.1(Description Diagram)

Another popular area where data mining can be used in the banking industry is in fraud detection. Being able to detect fraudulent actions is an increasing concern for many businesses; and with the help of data mining more fraudulent actions are being detected and reported. Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and

traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design. It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality. It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

4.2 ARCHITETURAL DESIGN

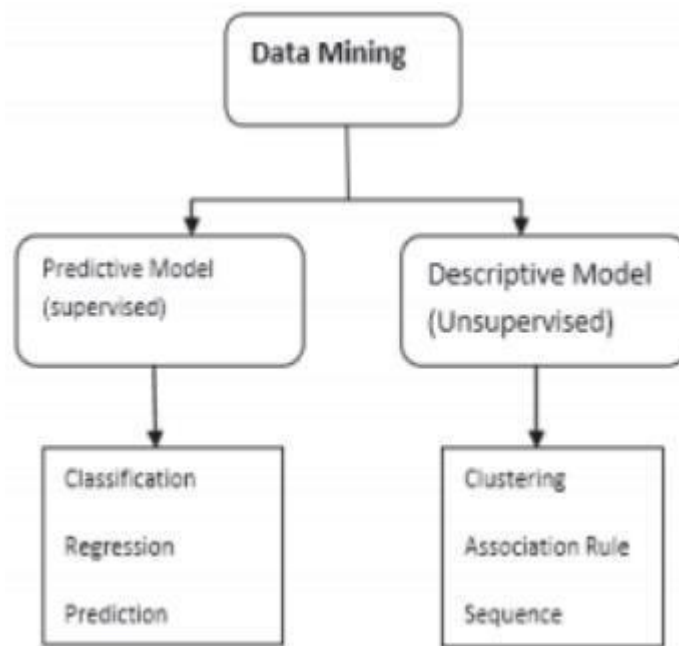


Fig 4.2(Architctural Design)

The number of Non-Performing Loans has increased in recent years, paralleling the current financial crisis, thus increasing the importance of credit scoring models. This study proposes a three stage hybrid Adaptive Neuro Fuzzy Inference System credit scoring model, which is based on statistical techniques and Neuro Fuzzy. The proposed model's performance was compared with conventional and commonly utilized models. The credit scoring models are tested using a 10-fold cross- validation process with the credit card data of an international bank operating in Turkey. Results demonstrate that the proposed model consistently

performs better

than the Linear Discriminant Analysis, Logistic Regression Analysis, and Artificial Neural Network (ANN) approaches, in terms of average correct classification rate and estimated misclassification cost. As with ANN, the proposed model has learning ability; unlike ANN, the model does not stay in a black box. In the proposed model, the interpretation of independent variables may provide valuable information for bankers and consumers, especially in the explanation of why credit applications are rejected as banking markets in developing countries are maturing, banks face competition not only from other domestic banks but also from sophisticated foreign banks. Given the substantial growth of consumer credit and increased regulatory attention to risk management, the development of a well-functioning credit assessment framework is essential. As part of such a framework, we propose a credit scoring model for Vietnamese retail loans. First, we show how to identify those borrower characteristics that should be part of a credit scoring model. Second, we illustrate how such a model can be calibrated to achieve the strategic objectives of the bank. Finally, we assess the use of credit scoring models in the context of transactional versus relationship lending. Credit scoring is the term used to describe formal statistical methods used for classifying applicants for credit into 'good' and 'bad' risk classes. Such methods have become increasingly important with the dramatic growth in consumer credit in recent years. A wide range of statistical methods has been applied, though the literature available to the public is limited for reasons of commercial confidentiality. Particular problems arising in the credit scoring context are examined and the statistical methods which have been applied are reviewed.

4.3 DATAFLOW DIAGRAM

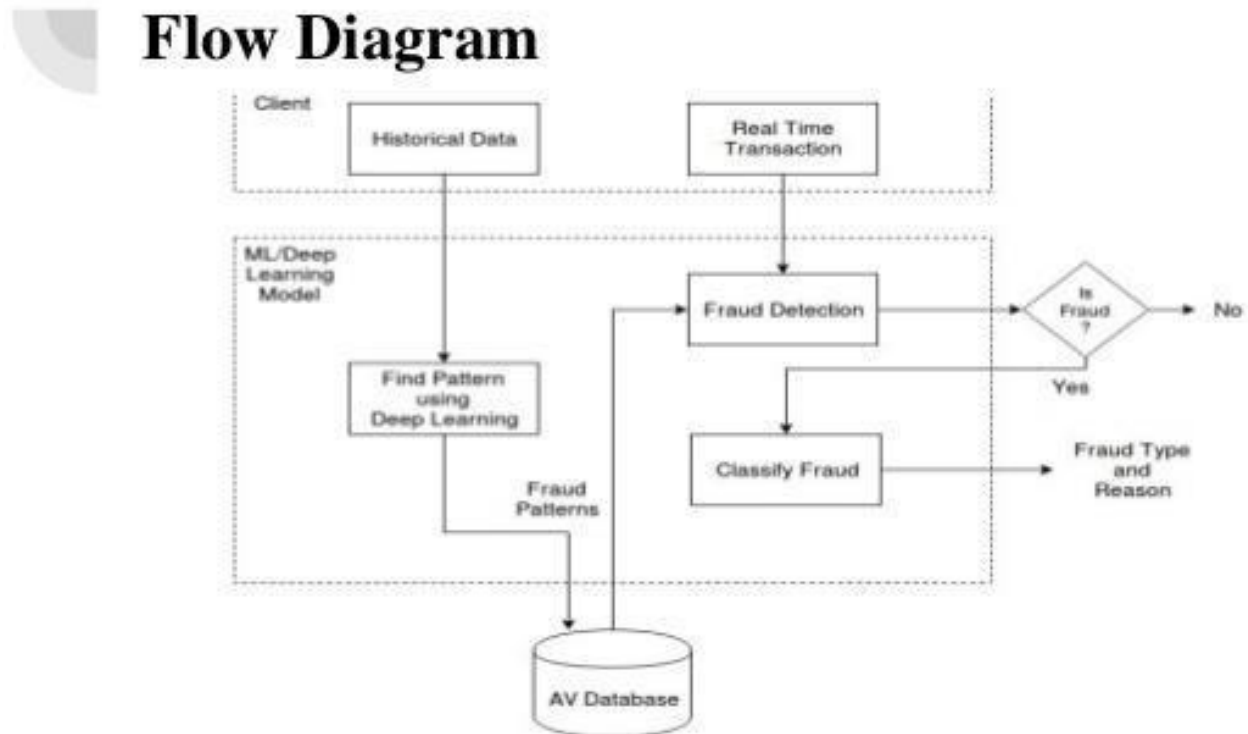


Fig 4.3(Flow Diagram)

The purpose of the present paper is to explore the ability of neural networks such as multilayer perceptron's and modular neural networks, and traditional techniques such as linear discriminant analysis and logistic regression, in building credit scoring models in the credit union environment. Also, since funding and small sample size often preclude the use of customized credit scoring models at small credit unions, we investigate the performance of generic models and compare them with customized models. Our results indicate that customized neural networks offer a very promising avenue if the measure of performance is percentage of bad loans correctly classified. However, if the measure of performance is percentage of good and bad loans correctly classified, logistic regression models are comparable to the neural networks approach. The performance of generic models was not as good as the customized models, particularly when it came to correctly classifying bad loans. Although we found significant differences in the results for the three credit unions, our modular neural network could not accommodate these differences, indicating

that more innovative architectures might be necessary for building effective generic models.

4.4 FRAUD DETECTION

Some of the biggest threats to data quality are bots and cheaters. Often bots will complete surveys in masse, or a person will take surveys on behalf of someone else multiple times. Thankfully, Expert Review tracks these patterns, so you can ensure your surveys are collecting only the highest quality of data.

With ExpertReview Fraud Detection, you can do the following with responses identified as fraudulent:

1. Discard them, preventing them from being counted against auditable responses or quotas.
2. Redirect these responses for analysis separately.
3. Flag these responses so they can be filtered, reported on, etc.
4. Analyze the number of fraudulent responses you've received, breaking them down by duplicates and bots.

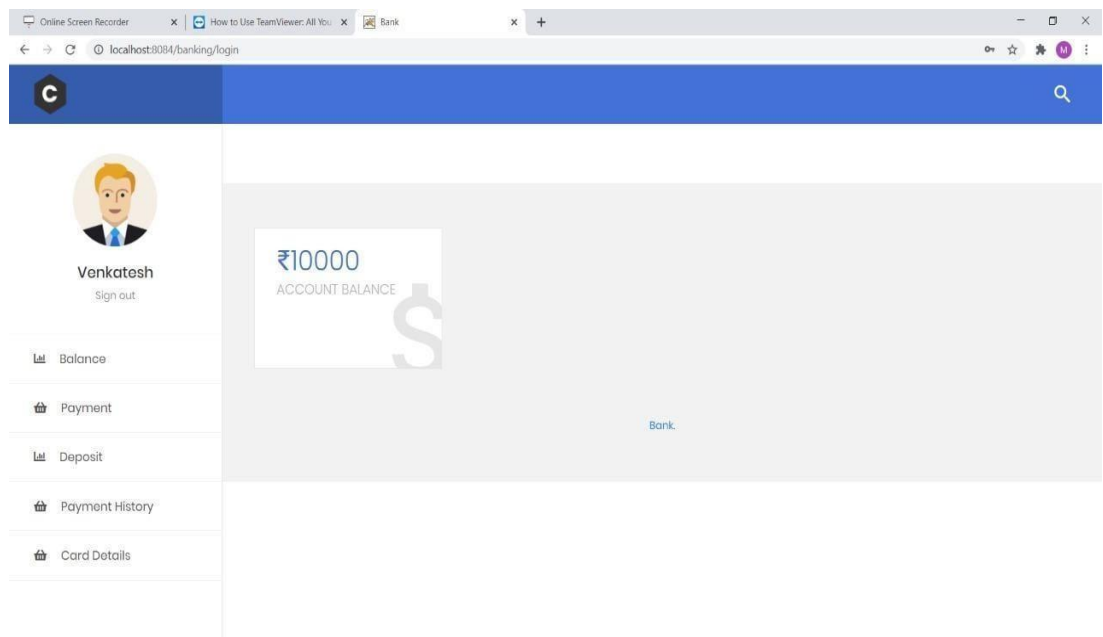
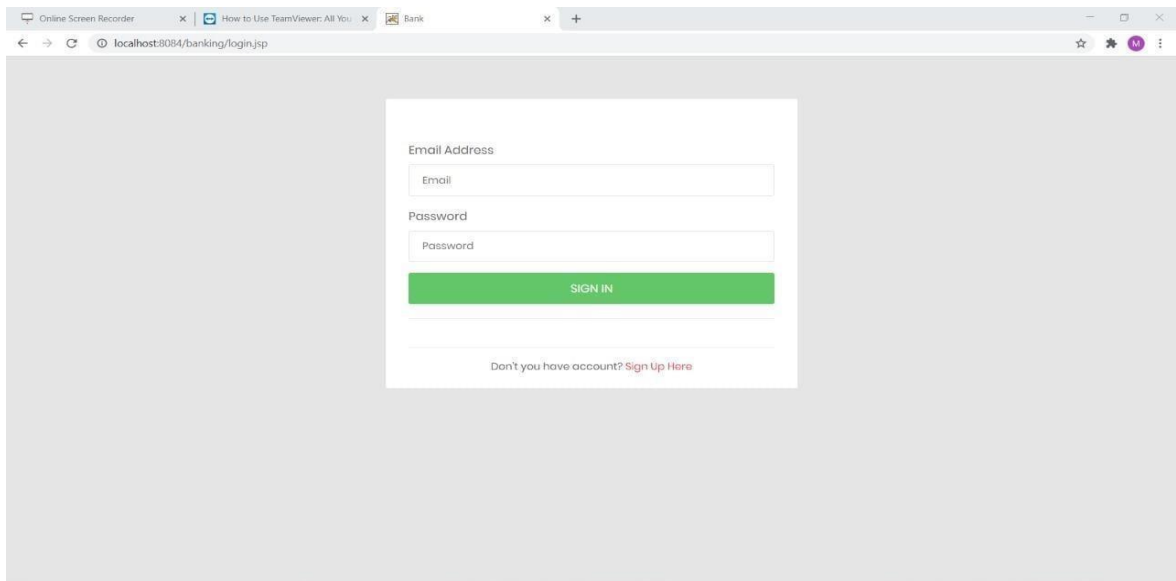
You can also prevent email scanning software from inadvertently submitting a survey response to begin with. You can enable Prevent Ballot Box Stuffing, Bot Detection, Email Scan Roadblock, and RelevantID all in the same survey.

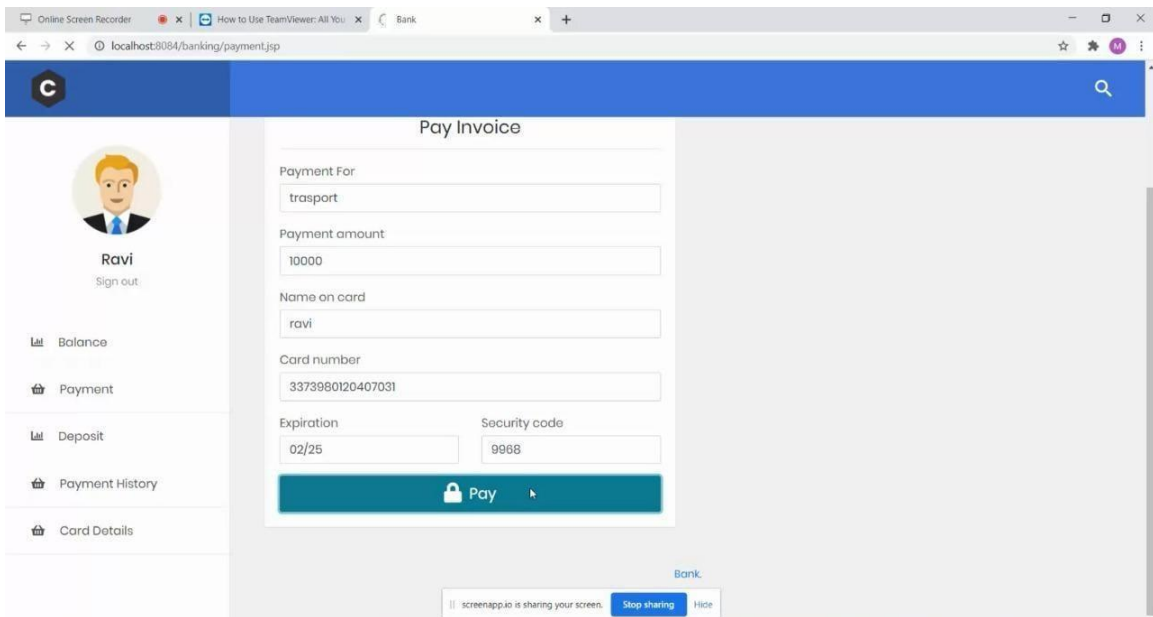
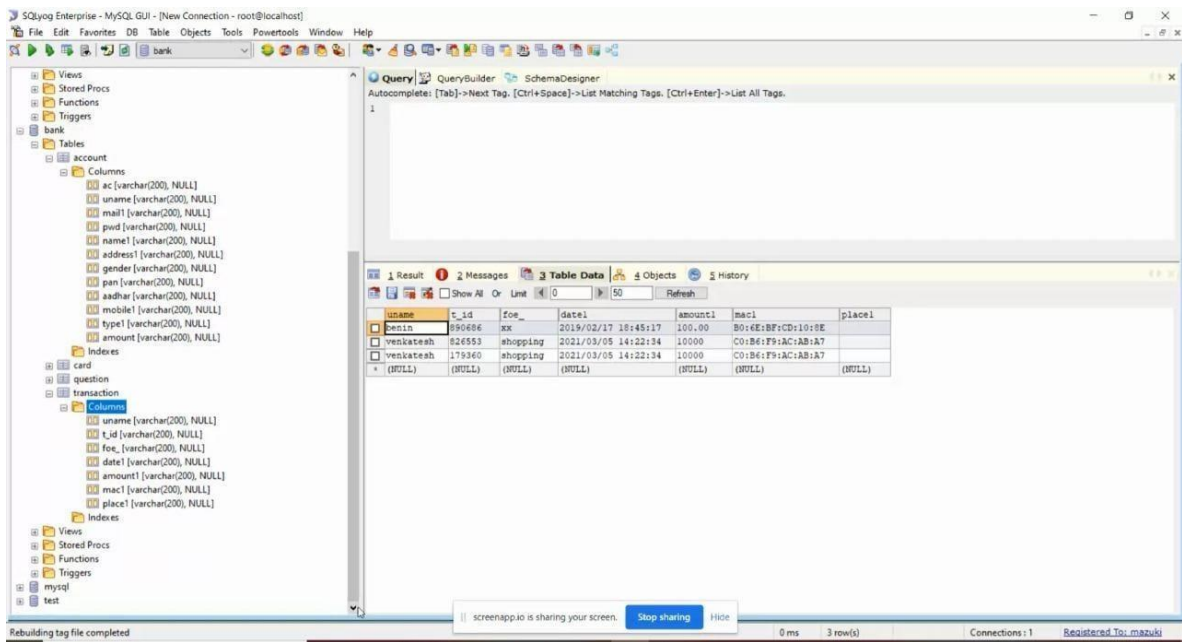
CHAPTER 5

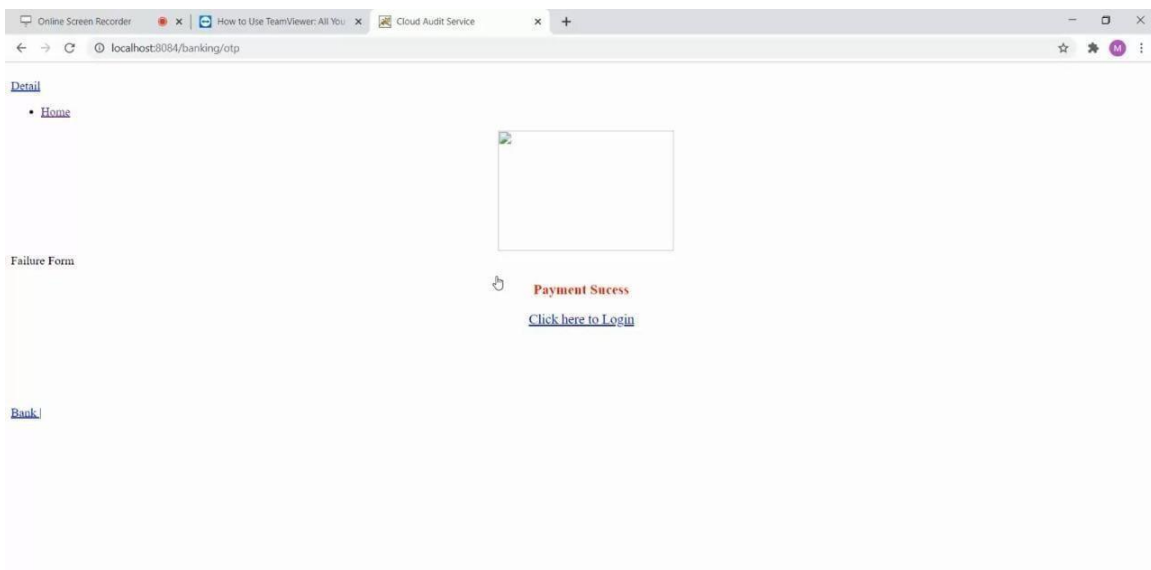
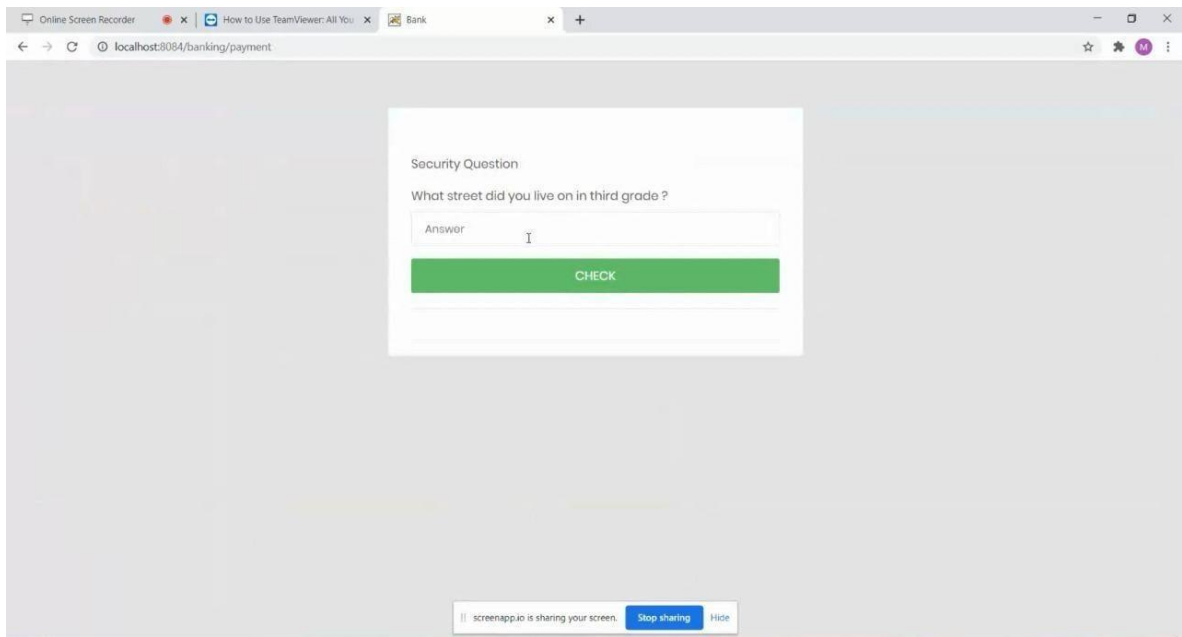
RESULT AND DISCUSSION

First we will get the input from the user and the new user should sign in as the user is new to the site and by giving all of the input details such as email ,pan card details and other details related to the account now the user will get the login and now the user can access the banking page and now the main task begins where the user starts the payment and while the user doing the payment the user will be asked to answer the security questions where if the user is a true user he will be able to give the correct answer to the security question. So incase the user is a fraud user he will not be able to give the answer to the security question so by this we can find the fraud user and Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement. Train and Test buttons are given to the user where using train the algorithms are trained and then o predict the fraud by clicking predict button it will take to another window where the input is given and output is seen as fraud or nonfraud. machine learning algorithms and in the predict it will give output as fraud or nonfraud. comma separated 30 values are given including amount and time. Predicted result is displayed as fraud after providing the data. These results along with the classication report for each algorithm is given in the output as follows, where class 0 means the transaction was determined to be valid and 1 means it was determined as a fraud transaction. detection of fraud or nonfraud transaction. when predict button is clicked it will take to another window where it asks for data which is input to thedetection of fraud or nonfraud transaction. when predict button is clicked it will take to another window where it asks for data which is input to the machine learning algorithms and in the predict it will give output as fraud or nonfraud. comma separated 30 values are given including amount and time. Predicted

result is displayed as fraud after providing the data.







CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Machine Learning is a technique used to extract vital information from existing huge amount of data and enable better decision-making for the banking and retail industries. They use data warehousing to combine various data from databases into an acceptable format so that the data can be mined. The data is then analyzed and the information that is captured is used throughout the organization to support decision-making. Data Mining techniques are very useful to the banking sector for better targeting and acquiring new customers, most valuable customer retention, automatic credit approval which is used for fraud prevention, fraud detection in real time, providing segment based products, analysis of the customers, transaction patterns over time for better retention and relationship, risk management and marketing.

6.2 FUTURE WORK

At Bolt, our difference is fundamental. As the first full-stack market solution to handle checkout, payments, and fraud, we have unique visibility into the full suite of checkout payment information to use for fraud detection purposes. Bolt's competitive access to data produces ample benefits to using Bolt in terms of time, labor-hours, and saved revenue. Fraud modeling, like any other statistical modeling or machine learning, relies on having clear, consistent, normalized access to data. Very simply, without access to a particular feature (variable) that might be a strong signal, a machine-learning model is at a disadvantage. Bolt, as a payments processor and checkout flow, sees everything a merchant sees and more. Consider the types of insights Bolt might pick up that a typical fraud tool could not.

REFERENCES

- [1] S. M. Darwish, "An intelligent credit card fraud detection approach based on semantic fusion of two classifiers," (in English), *Soft Computing*, Article vol. 24, no. 2, pp. 1243-1253, Jan 2020.
- [2] A. Eshghi and M. Kargari, "Introducing a new method for the fusion of fraud evidence in banking transactions with regards to uncertainty," (in English), *Expert Systems with Applications*, Article vol. 121, pp. 382-392, May 2019.
- [3] S. Hossain, A. Abtahee, I. Kashem, M. M. Hoque, and I. H. Sarker, "Crime Prediction Using Spatio-Temporal Data," in *Computing Science, Communication and Security*, Singapore, 2020, pp. 277-289: Springer Singapore.
- [4] M. Zamini and S. M. H. Hasheminejad, "A comprehensive survey of anomaly detection in banking, wireless sensor networks, social networks, and healthcare," (in English), *Intelligent Decision Technologies-Netherlands*, Article vol. 13, no. 2, pp. 229-270, 2019.
- [5] I. Gonzalez-Carrasco, J. L. Jimenez-Marquez, J. L. Lopez-Cuadrado, and B. Ruiz-Mezcua, "Automatic detection of relationships between banking operations using machine learning," (in English), *Information Sciences*, Article vol. 485, pp. 319-346, Jun 2019.
- [6] M. Pohoretskyi, D. Serhieieva, and Z. Toporetska, "The proof of the event of a financial resources fraud in the banking sector: problematic issues," (in English), *Financial and Credit Activity-Problems of Theory and Practice*, Article vol. 1, no. 28, pp. 36-45, 2019.
- [7] K. Noor et al., "Performance analysis of a surveillance system to detect and track vehicles using Haar cascaded classifiers and optical flow method," 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA), Siem Reap, 2017, pp. 258-263.

APPENDIX

```
import java.io.IOException;

import java.io.PrintWriter;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import sun.misc.BASE64Encoder;

@WebServlet(urlPatterns = {"/reg"})

public class reg extends HttpServlet {

    Connection con=null;

    Statement st=null;
```

```

    ResultSet rs=null;

    Statement st1=null;

    RequestDispatcher rd=null;

protected    void    processRequest(HttpServletRequest request,
    HttpServletResponse response)

        throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");

    try (PrintWriter out = response.getWriter()) {

out.println("<!DOCTYPE html>");

        out.println("<html>");

        out.println("<head>");

        out.println("<title>Servlet reg</title>");

        out.println("</head>");

        out.println("<body>");

        out.println("<h1>Servlet reg at " + request.getContextPath() +
"</h1>");

        out.println("</body>");

        out.println("</html>");

    }

}

@Override

protected    void    doGet(HttpServletRequest request,
    HttpServletResponse response)

```

```

        throws ServletException, IOException {

    processRequest(request, response);

}

@Override

    protected void doPost(HttpServletRequest req, HttpServletResponse
res)

        throws ServletException, IOException {

        String username= req.getParameter("username");

        String password= req.getParameter("password");

        String name= req.getParameter("name1");


        String emailid1= req.getParameter("email");

        String address= req.getParameter("address");

        String gender= req.getParameter("gender");

        RequestDispatcher rd;

        if(gender.equals("0"))

        {

            rd=req.getRequestDispatcher("register.jsp");

```

```
        rd.forward(req,res);

    }

    else

    {

try {

        HttpSession sn = req.getSession(true);

        sn.setAttribute("username",username);

        sn.setAttribute("email",emailid1);

        sn.setAttribute("pwd",password);

        sn.setAttribute("name",name);

        sn.setAttribute("address",address);

        sn.setAttribute("gender",gender

);

        rd=req.getRequestDispatcher("nextpage.jsp");

        rd.forward(req,res);

    } catch(Exception e2) {

        // rd=req.getRequestDispatcher("failure.jsp");

        rd=req.getRequestDispatcher("register.jsp");

        rd.forward(req,res);

    }
```

```

        }

    }

    @Override

    public String getServletInfo() {

        return "Short description";

    }

}

import java.io.IOException;

import java.io.PrintWriter;

import static java.lang.Math.random;

import java.math.BigInteger;

import java.security.MessageDigest;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.util.Random;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

```

```
import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import sun.misc.BASE64Encoder;

@WebServlet(urlPatterns = {"/reg1"})

public class reg1 extends HttpServlet {

    Connection con=null;

    Statement st=null;

    ResultSet rs=null;

    Statement st1=null;

    RequestDispatcher rd=null;

    protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)

        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        try (PrintWriter out = response.getWriter()) {

            out.println("<!DOCTYPE html>");

            out.println("<html>");

            out.println("<head>");

            out.println("<title>Servlet reg</title>");

            out.println("</head>");

            out.println("<body>");
```

```
        out.println("<h1>Servlet reg at " + request.getContextPath() +  
"</h1>");
```

```
        out.println("</body>");
```

```
        out.println("</html>");
```

```
    }
```

```
}
```

@Override

```
    protected void doGet(HttpServletRequest request,  
HttpServletRequest response)
```

```
        throws ServletException, IOException {
```

```
        processRequest(request, response);
```

```
    }
```

@Override

```
    protected void doPost(HttpServletRequest req, HttpServletResponse  
res)
```

```
        throws ServletException, IOException {
```

```
String pan= req.getParameter("pan");
```

```
String aadhar= req.getParameter("aadhar");
```

```
String mobile= req.getParameter("mobile");
```

```
String type= req.getParameter("ac_type");
```

RequestDispatcher rd;

```

    HttpSession sn = req.getSession(true);

    String username1= sn.getAttribute("username").toString();

    String email=  sn.getAttribute("email").toString();

    String pwd= sn.getAttribute("pwd").toString();

    String name= sn.getAttribute("name").toString();

    String address= sn.getAttribute("address").toString();

    String gender= sn.getAttribute("gender").toString();

    sn.setAttribute("balance","0");

    try
    {

        Random rnd = new Random();

        int n = 100000 + rnd.nextInt(900000);

        int n1 = 100000 + rnd.nextInt(900000);

        String n3=String.valueOf(n)+String.valueOf(n1);

        String                                card=String.format("%04d",
rnd.nextInt(10000))+String.format("%04d",
rnd.nextInt(10000))+String.format("%04d",
rnd.nextInt(10000))+String.format("%04d", rnd.nextInt(10000)));

        String pin = String.format("%04d", rnd.nextInt(10000));

        Class.forName("com.mysql.jdbc.Driver");

```

```

        con
        =
DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root","root");

        st = con.createStatement();

        int add=st.executeUpdate("insert into account
values('"+n3+"','"+username1+"','"+email+"','"+pwd+"','"+name+"','"+
+address+"','"+gender+"','"+pan+"','"+aadhar+"','"+mobile+"','"+type+"','0')");

        add=st.executeUpdate("insert into card
values('"+username1+"','"+card+"','02/25','"+pin+"','"+getSaltString()
+"')");

// st1 = con.createStatement();

// int add1=st1.executeUpdate("insert into usersec
values('"+username+"','"+sb.toString()+"','"+dec1.mod(bi)+"')");

        rd=req.getRequestDispatcher("balance.jsp");

        rd.forward(req,res);

    } catch(Exception e2) {

        rd=req.getRequestDispatcher("failure.jsp");

    }

}

@Override

```

```

public String getServletInfo() {

    return "Short description";

} // </editor-fold>

public String getSaltString() {

    String SALTCHARS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";

    StringBuilder salt = new StringBuilder();

    Random rnd = new Random();

    while (salt.length() < 5) { // length of the random string.

        int index = (int) (rnd.nextFloat() * SALTCHARS.length());

        salt.append(SALTCHARS.charAt(index));

    }

    String saltStr = salt.toString();

    return saltStr;

}

}

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

```

```
import java.sql.Statement;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

@WebServlet(urlPatterns = {"/login"})

public class login extends HttpServlet {

    String username="";

    String password="";

    String type1="";

    Connection con=null;

    Statement st=null;

    ResultSet rs=null;

    protected void processRequest(HttpServletRequest request,
    HttpServletResponse response)

        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        try (PrintWriter out = response.getWriter()) {

            out.println("<!DOCTYPE html>");
```

```

        out.println("<html>");

        out.println("<head>");

        out.println("<title>Servlet login</title>");

        out.println("</head>");

        out.println("<body>");

        out.println("<h1>Servlet login at " + request.getContextPath() +
"</h1>");

        out.println("</body>");

        out.println("</html>");

    }

}

```

@Override

```

protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {

    processRequest(request, response);

}

```

@Override

```

protected void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    username = req.getParameter("uname");

    password = req.getParameter("pwd");
}

```

```

//type1 = req.getParameter("type1");

HttpSession sn = req.getSession(true);

sn.setAttribute("username",username);


System.out.println("username : " +username);

System.out.println("otp : " +password);

System.out.println("type1 : " +type1);

RequestDispatcher rd = null;

try {

    Class.forName("com.mysql.jdbc.Driver");

    con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",
"root");

    st = con.createStatement();

    rs = st.executeQuery("select * from account
where uname='"+username+"' && pwd='"+password+'");

    if(rs.next()) {

        sn.setAttribute("balance",rs.getString(12));

        sn.setAttribute("type",rs.getString(11));

        rd=req.getRequestDispatcher("balance.jsp");

        //sn.setAttribute("dpm",department);

```

```
    } else {

        rd=req.getRequestDispatcher("failure.jsp");

        }

        rd.forward(req,res);

    }

    catch(Exception e2)

    {

        System.out.println("Exception : "+e2.toString());

    }

}

@Override

public String getServletInfo() {

    return "Short description";

}}

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;
```

```

import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = {"/card"})

public class card extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)

        throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");

    try (PrintWriter out = response.getWriter()) {

        out.println("<!DOCTYPE html>");

        out.println("<html>");

        out.println("<head>");

        out.println("<title>Servlet card</title>");

        out.println("</head>");

        out.println("<body>");

        out.println("<h1>Servlet card at " + request.getContextPath() +
"</h1>");

        out.println("</body>");

        out.println("</html>");

    }

}

@Override

    protected void doGet(HttpServletRequest request,

```

HttpServletResponse response)

```

        throws ServletException, IOException {

        processRequest(request, response);

    }

    @Override

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response)

        throws ServletException, IOException {

        processRequest(request, response);

    }


    @Override

    public String getServletInfo() {

        return "Short description";

    }

}

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

```

```

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

@WebServlet(urlPatterns = {"/deposit"})

public class deposit extends HttpServlet {

    String username="";

    String password="";

    String type1="";

    Connection con=null;

    Statement st=null,st1=null;

    ResultSet rs=null;

    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)

        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        try (PrintWriter out = response.getWriter()) {

out.println("<!DOCTYPE html>");

        out.println("<html>");

        out.println("<head>");

        out.println("<title>Servlet

```

deposit</title>");

```

        out.println("</head>");

        out.println("<body>");

        out.println("<h1>Servlet deposit at " + request.getContextPath()
+ "</h1>");

        out.println("</body>");

        out.println("</html>");

    }

}

```

@Override

```

protected void doGet(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {

    processRequest(request, response);

}

```

@Override

```

protected void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

    username = req.getParameter("cc-payment");

    String username1= sn.getAttribute("username").toString();

    System.out.println("username : " +username);

    System.out.println("otp : " +password);

    System.out.println("type1 : " +type1);
}

```

```
RequestDispatcher rd = null;
```

```
try {
```

```
    Class.forName("com.mysql.jdbc.Driver");
```

```
    con
```

```
=
```

```
    DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root","root");
```

```
    st1 = con.createStatement();
```

```
    rs = st1.executeQuery("select amount from account  
where uname='"+username1+"'");
```

```
    if(rs.next()) {
```

```
        String aa=rs.getString(1);
```

```
        String
```

```
        bal=String.valueOf(Integer.parseInt(aa)+Integer.parseInt(username));
```

```
        int add=st1.executeUpdate("update account set  
amount='"+bal+"' where uname='"+username1+"'");
```

```
        sn.setAttribute("balance",bal);
```

```
        rd=req.getRequestDispatcher("balance.jsp");
```

```
        //sn.setAttribute("dpm",department);
```

```
    } else {
```

```
        rd=req.getRequestDispatcher("failure.jsp");
```

```

        }

        rd.forward(req,res);

    }

    catch(Exception e2)

    {

        System.out.println("Exception : "+e2.toString());

    }

}

```

@Override

```

    public String getServletInfo() {

        return "Short description";

    }
}

```

```

import java.net.InetAddress;

```

```

import java.net.NetworkInterface;

```

```

import java.net.SocketException;

```

```

import java.net.UnknownHostException;

```

```

public class HardwareAddress

```

```

{

```

```

    public static String getMacAddress() throws UnknownHostException,

        SocketException

```

```
{  
  
    InetAddress ipAddress = InetAddress.getLocalHost();  
  
    NetworkInterface networkInterface = NetworkInterface  
        .getByInetAddress(ipAddress);  
  
    byte[] macAddressBytes = networkInterface.getHardwareAddress();  
  
    StringBuilder macAddressBuilder = new StringBuilder();  
  
    for (int macAddressByteIndex = 0; macAddressByteIndex  
< macAddressBytes.length; macAddressByteIndex++)  
    {  
        String macAddressHexByte = String.format("%02X",  
            macAddressBytes[macAddressByteIndex]);  
  
        macAddressBuilder.append(macAddressHexByte);  
  
        if (macAddressByteIndex != macAddressBytes.length - 1)  
        {  
            macAddressBuilder.append(":");  
        }  
    }  
  
    return macAddressBuilder.toString();  
}
```

```
public static void main(String[] args) throws UnknownHostException,  
SocketException
```

```
{
```

```
    System.out.println(getMacAddress() );
```

```
}
```

```
}
```

```
import java.io.IOException;
```

```
import java.io.PrintWriter;
```

```
import java.net.InetAddress;
```

```
import java.net.NetworkInterface;
```

```
import java.net.SocketException;
```

```
import java.net.UnknownHostException;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.Statement;
```

```
import javax.servlet.RequestDispatcher;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;
```

```
protected void processRequest(HttpServletRequest request,  
HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
response.setContentType("text/html;charset=UTF-8");
```

```
try (PrintWriter out = response.getWriter()) {
```

```
    out.println("<!DOCTYPE html>");
```

```
    out.println("<html>");
```

```
    out.println("<head>");
```

```
    out.println("<title>Servlet payment</title>");
```

```
    out.println("</head>");
```

```
    out.println("<body>");
```

```
    out.println("<h1>Servlet payment at " + request.getContextPath()  
+ "</h1>");
```

```
    out.println("</body>");
```

```
    out.println("</html>");
```

```
}
```

```
}
```

```
@Override
```

```
protected void doGet(HttpServletRequest request,  
HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
processRequest(request, response);
```

```
}
```

```
@Override
```

```
protected void doPost(HttpServletRequest req, HttpServletResponse  
res)
```

```
throws ServletException, IOException {
```

```
int a=0;
```

```
String cfor=req.getParameter("ccpayment");
```

```
String cam=req.getParameter("ccamount");
```

```
String cname=req.getParameter("ccname");
```

```
String cnum=req.getParameter("ccnumber");
```

```
String cexp=req.getParameter("ccexp");
```

```
String code=req.getParameter("code");
```

```
HttpSession sn = req.getSession(true);
```

```
System.out.println("type1 : " +getMacAddress());
```

```
RequestDispatcher rd = null;
```

```
try {
```

```
Class.forName("com.mysql.jdbc.Driver");
```

```
con
```

```
=
```

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/bank","root",  
"root");
```

```

    st = con.createStatement();

    rs = st.executeQuery("select count(*) from transaction where
uname='"+username1+"'");

    if(rs.next()) {

        a=Integer.parseInt(rs.getString(1));

        // sn.setAttribute("balance",rs.getString(12));

        // sn.setAttribute("type",rs.getString(11));

        // rd=req.getRequestDispatcher("balance.jsp");


        //sn.setAttribute("dpm",department);

    }

    if(a<20)

    {

        rs = st.executeQuery("select * from card where
uname='"+cname+"' and card_='"+cnum+"' and valid='"+cexp+"' and
pin='"+code+"' ");

        if(rs.next()) {

            sn.setAttribute("cfor",cfor);

            sn.setAttribute("cam",cam);

            sn.setAttribute("cname",cname);

```

```
sn.setAttribute("cnum",cnum);
```

```
sn.setAttribute("cexp",cexp);

sn.setAttribute("code",code);

sn.setAttribute("mac",getMacAddress());

rd=req.getRequestDispatcher("p_otp.jsp");

}

else

{

    rs = st.executeQuery("select * from card where
uname='"+cname+"' and card_='"+cnum+"' and valid='"+cexp+"' and
pin='"+code+"' ");

    if(rs.next()) {

        sn.setAttribute("cfor",cfor);

        sn.setAttribute("cam",cam);

        sn.setAttribute("cname",cname);

        sn.setAttribute("cnum",cnum);

        sn.setAttribute("cexp",cexp);

        sn.setAttribute("code",code);

        sn.setAttribute("mac",getMacAddress());

        rd=req.getRequestDispatcher("p_otp.jsp");
```

```

    }

    }

}

else

{

    rs = st.executeQuery("select * from card where
uname='"+cname+"' and card_='"+cnum+"' and valid='"+cexp+"' and
pin='"+code+"' ");

    if(rs.next()) {

        sn.setAttribute("cfor",cfor);

        sn.setAttribute("cam",cam);

        sn.setAttribute("cname",cname);

        sn.setAttribute("cnum",cnum);

        sn.setAttribute("cexp",cexp);

        sn.setAttribute("code",code);

        sn.setAttribute("mac",getMacAddress());

        rd=req.getRequestDispatcher("p_otp.jsp");

    }

}

```

```

        rd.forward(req,res);
    }

    catch(Exception e2)

    {

        e2.printStackTrace();

    }

}

@Override

public String getServletInfo() {

    return "Short description";

}}

SocketException

{

    InetAddress ipAddress = InetAddress.getLocalHost();

    NetworkInterface networkInterface = NetworkInterface

        .getByInetAddress(ipAddress);

    byte[] macAddressBytes = networkInterface.getHardwareAddress();

    StringBuilder macAddressBuilder = new StringBuilder();

```

```
        for (int macAddressByteIndex = 0; macAddressByteIndex
< macAddressBytes.length; macAddressByteIndex++)

        {

            String macAddressHexByte = String.format("%02X",

                macAddressBytes[macAddressByteIndex]);

            macAddressBuilder.append(macAddressHexByte);

            if (macAddressByteIndex != macAddressBytes.length - 1)

            {

                macAddressBuilder.append(":");

            }

        }

        return macAddressBuilder.toString();

    }

}

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;
```

```

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(urlPatterns = {"/payment_"})

public class payment_ extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        try (PrintWriter out = response.getWriter()) {

            out.println("<!DOCTYPE html>");

            out.println("<html>");

            out.println("<head>");

            out.println("<title>Servlet payment_</title>");

            out.println("</head>");

            out.println("<body>");

            out.println("<h1>Servlet payment_ at " +
                request.getContextPath() + "</h1>");

            out.println("</body>");

            out.println("</html>");

        }

    }

    @Override

```

```
protected void doGet(HttpServletRequest request,  
HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    processRequest(request, response);
```

```
}
```

```
@Override
```

```
protected void doPost(HttpServletRequest request,  
HttpServletResponse response)
```

```
throws ServletException, IOException {
```

```
    processRequest(request, response);
```

```
}
```

```
@Override
```

```
public String getServletInfo() {
```

```
    return "Short description";
```

```
}}
```