

A
Major Project Report
On
**Credit Card Fraud Detection Using State Art Of
Machine Learning and Deep Learning**

Submitted to CMREC, HYDERABAD

In Partial Fulfillment of the requirements for the Award of Degree of
**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)**

Submitted

By

M. Meghana	(218R1A6747)
S. Chiraag Kumar	(218R1A6757)
M. Sreeja	(218R1A6744)
K. Murali Manohar	(218R1A6734)

Under the Esteemed guidance of

Mr. A. SAIKIRAN

Assistant Professor, Department of CSE (Data Science)



Department of Computer Science & Engineering (Data Science)
CMR ENGINEERING COLLEGE
UGC AUTONOMOUS

(Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401.

2024-2025

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)

Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering (Data Science)



CERTIFICATE

This is to certify that the project entitled “**Credit Card Fraud Detection Using State Art of Machine Learning and Deep Learning**” is a Bonafide work carried out by

M. Meghana	(218R1A6747)
S. Chiraag Kumar	(218R1A6757)
M. Sreeja	(218R1A6744)
K. Murali Manohar	(218R1A6734)

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	Major Project Coordinator	Head of the Department	External Examiner
Mr. A. Saikiran	Mrs. G. Shruthi	Dr. M. Laxmaiah	
Assistant Professor CSE (Data Science), CMREC	Assistant Professor CSE (Data Science), CMREC	Professor & HOD CSE (Data Science), CMREC	

DECLARATION

This is to certify that the work reported in the present Major project entitled " **Credit Card Fraud Detection Using State Art of Machine Learning and Deep Learning**" is a record of Bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

M. Meghana	(218R1A6747)
S. Chiraag Kumar	(218R1A6757)
M. Sreeja	(218R1A6744)
K. Murali Manohar	(218R1A6734)

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Laxmaiah**, HOD, **Department of CSE (Data Science), CMR Engineering College** for their constant support.

We are extremely thankful to **Mr. A. Saikiran**, Assistant Professor, Internal Guide, Department of CSE(DS), for his constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if We do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mrs. G. Shruthi**, Assistant Professor, CSE(DS) Department, Major Project Coordinator for her constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided me for every step.

M. Meghana	(218R1A6747)
S. Chiraag Kumar	(218R1A6757)
M. Sreeja	(218R1A6744)
K. Murali Manohar	(218R1A6734)

ABSTRACT

With the increase in usage of credit cards, the capacity of credit card misuse has also enhanced. Credit card frauds cause significant financial losses for both credit card holders and financial companies. In this research study, the main aim is to detect such frauds, including the accessibility of public data, high-class imbalance data, the changes in fraud nature, and high rates of false alarm. The relevant literature presents many machine learning based approaches for credit card detection, such as Extreme Learning Method, Decision Tree, Random Forest, Support Vector Machine, Logistic Regression and XG Boost. However, due to low accuracy, there is still a need to apply state of the art deep learning algorithms to reduce fraud losses. Comparative analysis of both machine learning and deep learning algorithms was performed to find efficient outcomes. A machine learning algorithm was first applied to the dataset, which improved the accuracy of detection of the frauds to some extent. A comprehensive empirical analysis has been carried out by applying variations in the number of hidden layers, epochs and applying the latest models. The evaluation of research work shows the improved results achieved, such as accuracy, f1-score, precision and AUC Curves having optimized values. The proposed model outperforms the state-of-the-art machine learning and deep learning algorithms for credit card detection problems. In addition, we have performed experiments by balancing the data and applying deep learning algorithms to minimize the false negative rate. The proposed approaches can be implemented effectively for the real-world detection of credit card fraud.

CONTENTS

TOPIC	PAGE NO
ABSTRACT	v
LIST OF FIGURES	vii
1. INTRODUCTION	
1.1. Overview	1
1.2. Research Motivation	1
1.3. Problem Statement	2
1.4. Applications	2
2. LITERATURE SURVEY	4
3. EXISTING SYSTEM	6
4. PROPOSED SYSTEM	7
5. METHODOLOGY	
5.1. SDLC - Umbrella Model	9
5.2. Requirements Gathering Stage	9
5.3. Analysis Stage	11
5.4. Designing Stage	12
5.5. Coding Stage	12
5.6. Integration & Testing Stage	13
5.7. Installation & Acceptance Test	14
5.8. Maintenance	15
6. SYSTEM DESIGN AND UML DIAGRAMS	
6.1. System Architecture	16
6.2. Class Diagram	19
6.3. Use Case Diagram	20
6.4. Sequence Diagram	21
6.5. Activity Diagram	22
7. MACHINE LEARNING	
7.1. Categories of Machine Learning	24
7.2. Need for Machine Learning	25
7.3. Challenges in Machine Learning	25
7.4. Applications of Machine Learning	26
7.5. Advantages & Disadvantages of ML	26
7.6. Machine Learning Algorithms	27
8. SOFTWARE ENVIRONMENT	
8.1. What is python and its Advantages and Disadvantages	31
8.2. History of python	34
8.3. Modules used in project	36
8.4. Installation of python	38
9. SYSTEM REQUIREMENTS SPECIFICATIONS	
9.1. Software Requirements	45
9.2. Hardware Requirements	45

10.	FUNCTIONAL REQUIREMENTS	
	10.1. Output Design and Definition	46
	10.2. Input Design, Stages, Types, Media	46
	10.3. User Interface Design	48
	10.4. Performance Requirements	49
11.	SYSTEM TESTING	
	11.1. System Testing	51
	11.2. Module Testing	51
	11.3. Integration Testing	51
	11.4. Acceptance Testing	51
12.	SOURCE CODE	52
13.	RESULTS AND DISCUSSION	59
14.	CONCLUSION AND REFERENCES	62

LIST OF FIGURES

FIG.NO	DESCRIPTION	PAGE NO
5.1	Umbrella model	9
5.2	Requirements Gathering stage	10
5.3	Analysis Stage	11
5.4	Designing Stage	12
5.5	Coding Stage	13
5.6	Integration Stage & Testing Stage	14
5.7	Installation	15
6.1	Architecture Diagram for Credit Card Fraud	17
6.2	Class Diagram for fraud detection	20
6.3	Use case Diagram for fraud detection	21
6.5	Sequence Diagram for fraud detection	22
6.6	Activity Diagram for fraud detection	23
8.4	Python installation Diagrams	39
13.1	Credit Card Fraud Detection Web page	59
13.2	List of Remote Users	59
13.3	Registered Status of Users	60
13.4	Bar Chart of Credit Card Fraud	60
13.5	Prediction of Credit Card Fraud Detection	61
13.6	Line Chart of Credit Card Fraud Detection	61

1. INTRODUCTION

1.1 OVERVIEW

Credit card fraud (CCF) has become a major concern with the rise of digital payments and online transactions. It involves unauthorized use of credit card information to conduct fraudulent activities, often resulting in significant financial losses. The shift toward a cashless economy and the widespread adoption of e-banking and online shopping have further increased the risk of fraud. In response, the development of intelligent and automated fraud detection systems has become essential for financial institutions.

This study explores the use of state-of-the-art machine learning (ML) and deep learning (DL) techniques to detect fraudulent credit card transactions. Traditional ML models, such as Support Vector Machines (SVM), are compared with advanced DL models like Convolutional Neural Networks (CNN). A key focus is placed on addressing the class imbalance issue, where fraudulent transactions are vastly outnumbered by legitimate ones. Feature selection methods are used to identify the most important attributes for classification, and enhancements to the CNN model are proposed to improve performance.

The study conducts a comparative analysis of ML and DL methods on a real-world dataset, evaluating models based on accuracy, precision, and recall. Results demonstrate that the proposed CNN-based model significantly outperforms existing approaches in detecting fraudulent activities.

1.2 RESEARCH MOTIVATION

The rapid expansion of digital payment systems and online financial transactions has significantly increased the vulnerability to credit card fraud (CCF). As businesses and consumers shift toward a cashless society, the volume and complexity of transaction data have grown, making traditional fraud detection methods insufficient. Manual and rule-based systems are no longer effective in identifying sophisticated and evolving fraud patterns. This growing threat poses serious financial and reputational risks to individuals, institutions, and global economies.

The motivation for this research lies in the urgent need for intelligent, automated systems that can accurately detect fraudulent activities in real-time. Machine learning (ML) and deep learning (DL) techniques offer promising solutions by learning hidden patterns from historical transaction data and generalizing them to predict future frauds.

However, challenges such as class imbalance, feature selection, and the need for fast and scalable models remain unresolved in many existing approaches.

This research is driven by the goal of enhancing fraud detection by leveraging both ML and DL models, specifically by improving CNN architectures for better feature extraction and classification. By addressing key issues like data imbalance and model performance, this study aims to contribute toward more reliable, efficient, and accurate credit card fraud detection systems in real-world financial environments.

1.3 PROBLEM STATEMENT

Credit card fraud (CCF) is a significant issue in modern financial systems, with fraudsters constantly devising new methods to exploit digital payment platforms. The rapid growth of e-commerce and digital transactions has expanded the attack surface for credit card fraud, leading to billions of dollars in losses annually. Traditional fraud detection systems, which rely on predefined rules and manual monitoring, are no longer sufficient to address the complexity and scale of current fraud patterns. These systems struggle with the increasing volume of transactions and the ability to detect evolving fraudulent behaviours in real-time.

The challenge in detecting credit card fraud lies in the extreme class imbalance, where fraudulent transactions represent a small fraction of all transactions, making it difficult for models to learn accurate classification patterns. Additionally, the vast amount of data, including complex transaction features, further complicates the detection process. This issue requires advanced machine learning (ML) and deep learning (DL) models capable of efficiently processing large datasets and effectively distinguishing between legitimate and fraudulent transactions.

This research aims to address these challenges by proposing a novel deep learning-based approach using Convolutional Neural Networks (CNNs) to enhance the detection of fraudulent transactions. By incorporating feature selection techniques and handling class imbalance, this study seeks to develop a robust, accurate, and scalable fraud detection system.

1.4 APPLICATIONS

1. **Banking and Financial Institutions:** Fraud detection systems help banks monitor real-time transactions to detect and prevent unauthorized activities, protecting both the institution and customers from financial losses.
2. **E-commerce Platforms:** Online retailers use fraud detection algorithms to identify

suspicious transactions and reduce chargebacks, safeguarding their revenue and maintaining customer trust.

3. **Payment Gateways:** Services like PayPal, Stripe, and Razorpay implement ML/DL-based models to assess transaction legitimacy and block fraudulent payments during checkout.
4. **Credit Card Companies:** Companies like Visa, MasterCard, and American Express integrate fraud detection tools into their systems to offer secure payment environments and alert users of irregular activities.
5. **Insurance Industry:** Detecting fraudulent claims related to stolen credit cards or unauthorized transactions helps insurance companies reduce payouts and protect against false claims.
6. **Cybersecurity Solutions:** Fraud detection is often a core component of broader cybersecurity frameworks, helping organizations detect identity theft and prevent data breaches involving financial data.
7. **Mobile Wallets and Fintech Apps:** Apps like Google Pay, Apple Pay, and Venmo use real-time fraud analytics to flag unusual spending patterns and enhance user security.
8. **Regulatory Compliance and Risk Management:** Accurate fraud detection supports regulatory requirements and minimizes the risk associated with financial crime and money laundering.

2. LITERATURE SURVEY

1. An efficient real time model for credit card fraud detection based on deep learning: Machine Learning has significantly transformed data processing and classification, enabling the development of real-time, interactive, and intelligent fraud detection systems. This study proposes a fraud detection model based on deep neural networks, leveraging an autoencoder architecture to identify fraudulent transactions efficiently. The autoencoder-based approach enhances anomaly detection by learning normal transaction patterns and flagging deviations. The model is trained using a large dataset of real-world credit card transactions, demonstrating its ability to generalize across various fraud scenarios. Evaluation metrics such as accuracy, precision, recall, and F1-score indicate superior performance in distinguishing fraudulent activities from legitimate transactions. The benchmarking results confirm that the proposed model is highly effective in detecting fraud in real-time environments, offering a robust solution for financial institutions.

2. Facilitating user authorization from imbalanced data logs of credit cards using artificial intelligence: The application of machine learning in financial risk assessment has proven to be instrumental in automating fraud detection for commercial organizations and credit agencies. This paper presents an AI-driven predictive framework designed to assist credit bureaus in evaluating credit card delinquency risks. The study addresses the challenge of imbalanced datasets, where fraudulent transactions constitute only a small fraction of the total records. The proposed framework incorporates advanced resampling techniques and cost-sensitive learning strategies to mitigate class imbalance issues. The model is assessed using multiple evaluation metrics, including sensitivity, specificity, precision, F1-score, and the area under the receiver operating characteristic (ROC) and precision-recall curves. Experimental results suggest that AI-driven models can enhance fraud detection and risk assessment accuracy, contributing to better decision-making in financial institutions.

3. Performance analysis of feature selection methods in software defect prediction: High-dimensional data poses a significant challenge in software defect prediction (SDP), impacting the accuracy and efficiency of predictive models. This study explores the role of feature selection (FS) techniques in improving the performance of SDP models by reducing data dimensionality. The research evaluates four filter feature ranking (FFR)

methods and fourteen filter feature subset selection (FSS) methods across five publicly available software defect datasets obtained from the NASA repository. The experimental results reveal that feature selection enhances the predictive accuracy of machine learning classifiers by eliminating irrelevant or redundant features. Furthermore, the study finds that the effectiveness of FS methods varies depending on the dataset and classifier used. However, FFR methods exhibit greater stability in terms of predictive performance, making them a reliable choice for SDP applications. The findings provide valuable insights into selecting optimal FS techniques for software quality assurance and defect prediction.

4. Hybrid Machine Learning Approach for Credit Card Fraud Detection: The increasing sophistication of fraudulent transactions necessitates the development of robust fraud detection systems that can adapt to evolving patterns. This research presents a hybrid machine learning model that combines supervised and unsupervised learning techniques to improve fraud detection accuracy. The proposed model integrates Random Forest (RF) for feature selection and Gradient Boosting (GB) for classification, while also incorporating an anomaly detection mechanism using Isolation Forest. The study evaluates the model on a publicly available credit card dataset and compares its performance against traditional machine learning classifiers. Key performance indicators such as accuracy, recall, precision, and F1-score demonstrate that the hybrid approach outperforms standalone models in detecting fraudulent transactions while minimizing false positives. The research highlights the importance of combining multiple techniques to enhance fraud detection capabilities in financial systems.

3. EXISTING SYSTEM

ML has many branches, and each branch can deal with different learning tasks. However, ML learning has different framework types. The ML approach provides a solution for CCF, such as random forest (RF). The ensemble of the decision tree is the random forest [3]. Most researchers use the RF approach. To combine the model, we can use (RF) along with network analysis. This method is called APATE [1]. Researchers can use different ML techniques, such as supervised learning and unsupervised techniques. ML algorithms, such as LR, ANN, DT, SVM and NB, are commonly used for CCF detection.

The researcher can combine these techniques with ensemble techniques to construct solid detection classifiers [3]. The linking of multiple neurons and nodes is known as an artificial neural network. A feed-forward perceptron multilayer is built up of numerous layers: an input layer, an output layer and one or more hidden layers. For the representation of the exploratory variables, the first layer contains the input nodes. With a precise weight, these input layers are multiplied, and each of the hidden layer nodes is transferred with a certain bias, and they are added together.

An activation function is then applied to create the output of each neuron for this summation, which is then transferred to the next layer. Finally, the algorithm's reply is provided by the output layer. The first set randomly used weights and formerly used the training set to minimise the error. All these weights were adjusted by detailed algorithms such as backpropagation [2], [6]. The graphic model for contingency relationships between a set of variables is called the Bayesian belief network. The independence assumption in naïve Bayes is that it was developed to relax and allow for dependencies among variables.

Disadvantages

1. The system is not implemented Classification on Imbalanced Data.
2. The system is not implemented CONVOLUTIONAL NEURAL NETWORK (CNN) for test and train the datasets.

4. PROPOSED SYSTEM

Credit card fraud detection is a crucial aspect of financial security, necessitating advanced machine learning (ML) and deep learning (DL) techniques to identify fraudulent transactions accurately. One of the key challenges in fraud detection is identifying the most relevant features from vast transactional datasets. To address this, feature selection algorithms are employed to rank the top features from the CCF transaction dataset, significantly aiding in class label predictions. By focusing on the most relevant attributes, feature selection enhances the model's efficiency and reduces computational overhead, allowing for improved detection performance.

To further strengthen fraud detection, a deep learning model is proposed, incorporating additional layers that improve feature extraction and classification. Unlike traditional ML models, which rely on predefined statistical rules, deep learning models can automatically learn complex patterns in data. The proposed model leverages convolutional neural networks (CNNs), a powerful deep learning architecture, to capture intricate transactional behaviours. By introducing multiple CNN layers, the model enhances its ability to detect fraudulent activities with greater precision. Different CNN layer architectures are explored to optimize the model's performance, ensuring that it can accurately distinguish between legitimate and fraudulent transactions.

A comprehensive comparative analysis is conducted to evaluate the effectiveness of the proposed model. This analysis compares the performance of conventional ML algorithms, such as logistic regression, decision trees, and random forests, against deep learning approaches, particularly the CNN model. Additionally, the proposed CNN model is benchmarked against a baseline model to determine its superiority. The results of these comparisons demonstrate that the proposed deep learning approach consistently outperforms existing methods in terms of accuracy, precision, and recall.

Performance evaluation is a critical aspect of fraud detection models, as it determines their reliability in real-world applications. To assess classifier accuracy, key performance metrics such as accuracy, precision, and recall are utilized. Accuracy

measures the overall correctness of the model's predictions, while precision evaluates the proportion of correctly identified fraudulent transactions. Recall, on the other hand, assesses the model's ability to detect fraudulent cases among all actual fraud cases. By incorporating these evaluation metrics, the effectiveness of the proposed model is rigorously validated.

Experiments are conducted on the latest credit card dataset to ensure the model's robustness and applicability to real-world scenarios. The dataset comprises a vast number of transactions, including both legitimate and fraudulent ones, making it an ideal benchmark for testing fraud detection models. The proposed model is trained and tested using this dataset, and the results confirm its superior performance in identifying fraudulent transactions with high accuracy.

In conclusion, the integration of feature selection techniques with deep learning models significantly enhances credit card fraud detection. By ranking the most relevant features, the model optimizes its predictive capabilities while reducing unnecessary complexity. The use of CNN architectures further improves classification performance, surpassing traditional ML models and baseline methods. Through rigorous performance evaluation and experimentation on real-world datasets, the proposed approach proves to be a highly effective solution for fraud detection. As financial fraud continues to evolve, such advanced machine learning techniques will play a pivotal role in safeguarding financial transactions and preventing fraudulent activities.

Advantages

1. The proposed system uses supervised machine learning approaches which are effective for testing and training datasets.
2. The proposed system implemented CNN is to minimize processing without losing key features by reducing the image to make predictions

5. METHODOLOGY

5.1 SDLC (SOFTWARE DEVELOPMENT LIFE CYCLE) – UMBRELLA MODEL

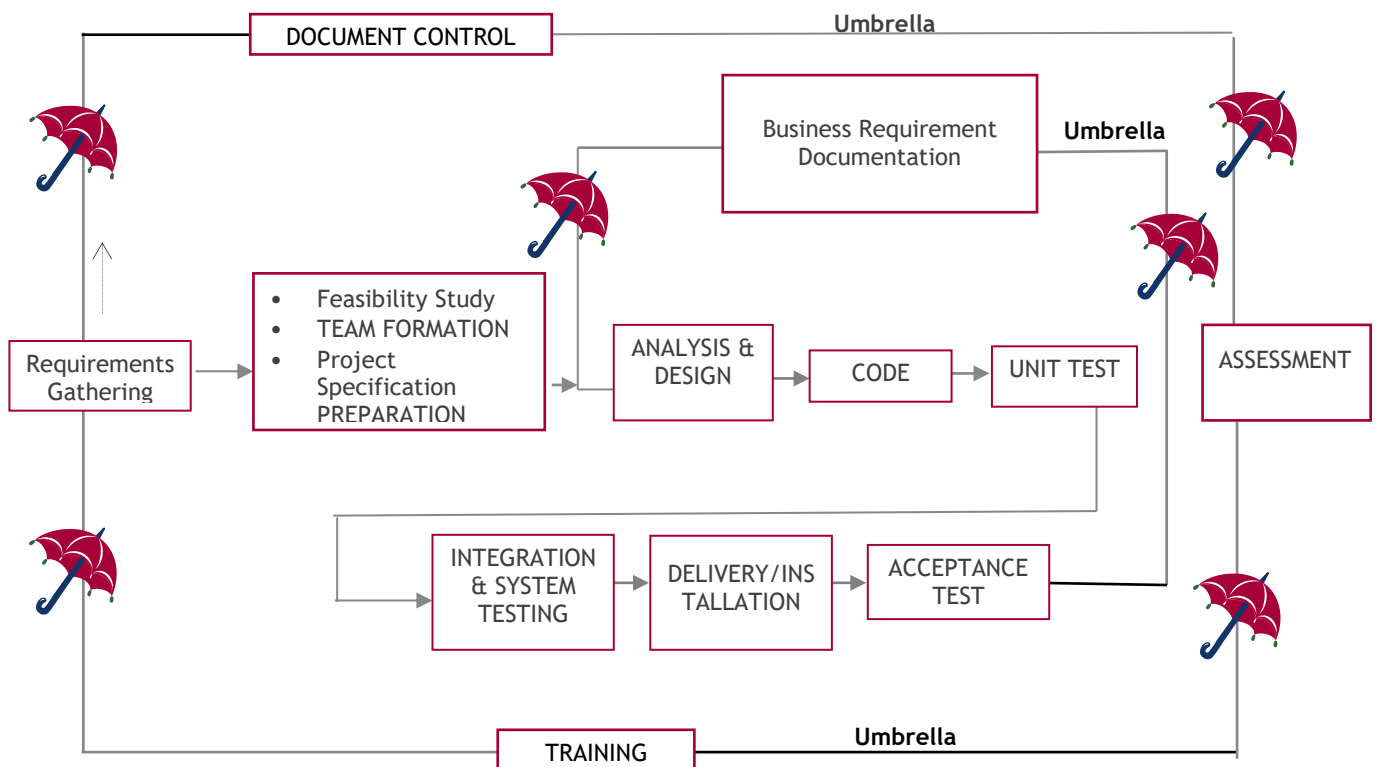


Fig 5.1 Umbrella model

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

5.2 REQUIREMENTS GATHERING STAGE

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these

definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

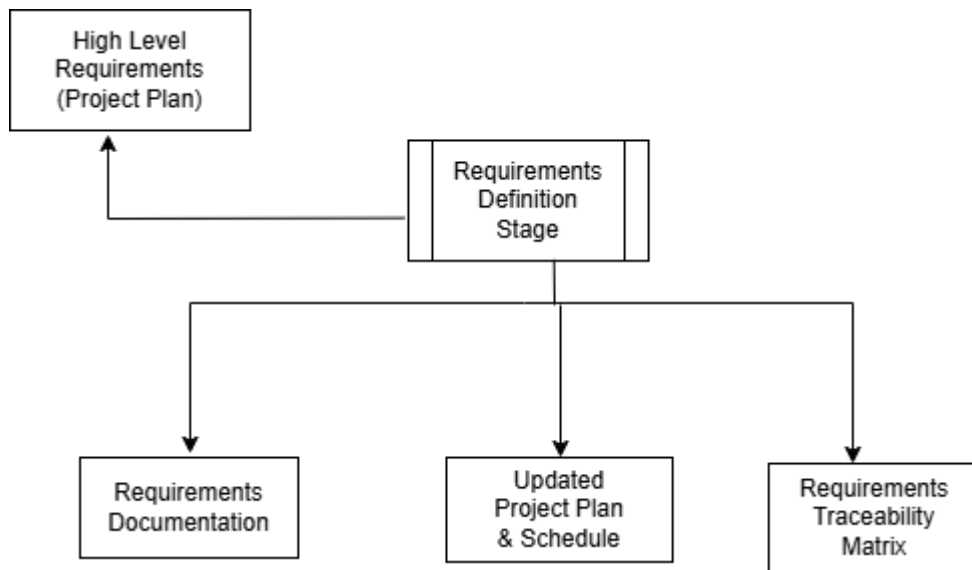


Fig 5.2 Requirements Gathering stage

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are not included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

Feasibility study is all about identification of problems in a project, number of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project. Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

5.3 ANALYSIS STAGE

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.

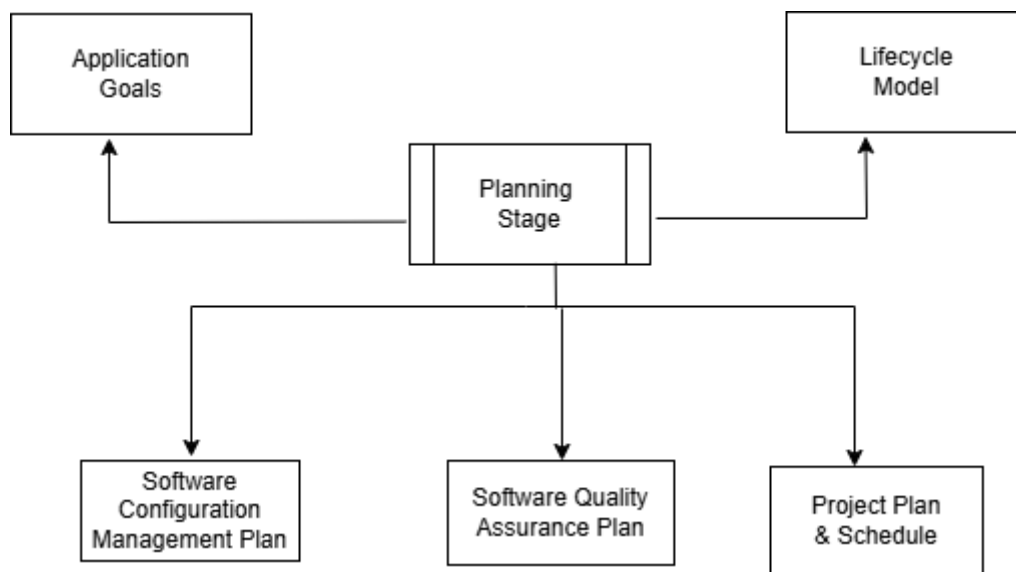


Fig 5.3 Analysis stage

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

5.4 DESIGNING STAGE

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

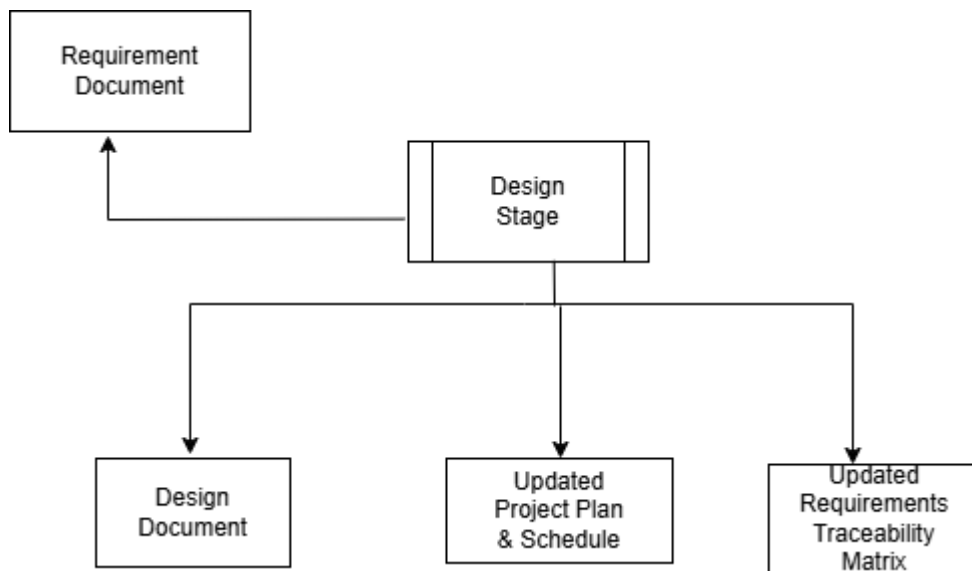


Fig 5.4 Designing stage

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

5.5 DEVELOPMENT (CODING) STAGE

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures

and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

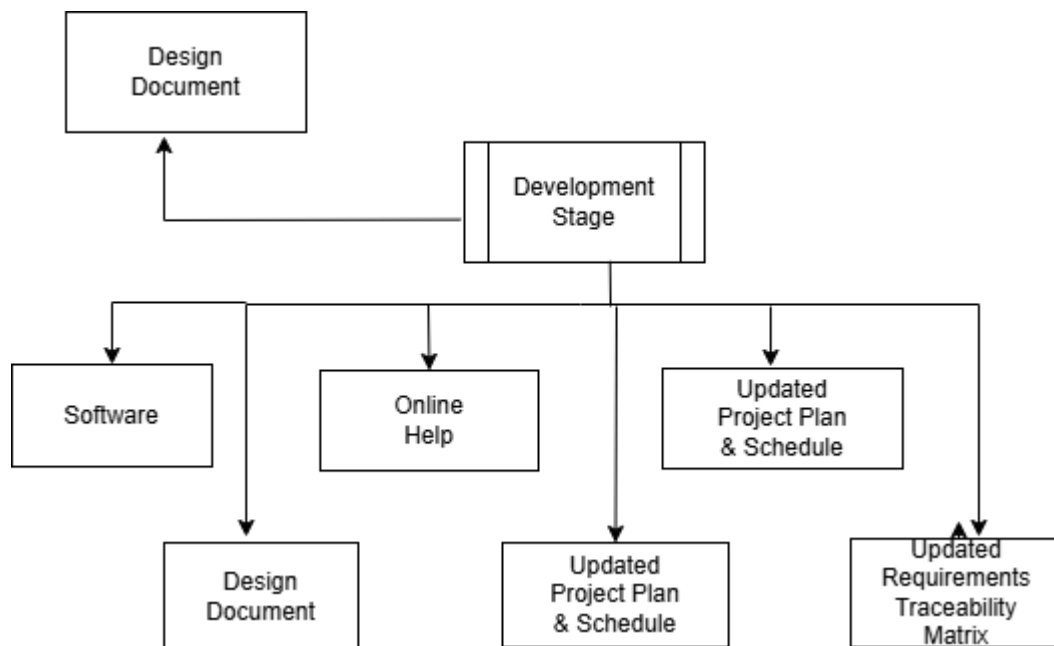


Fig 5.5 Coding stage

5.6 INTEGRATION & TEST STAGE

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

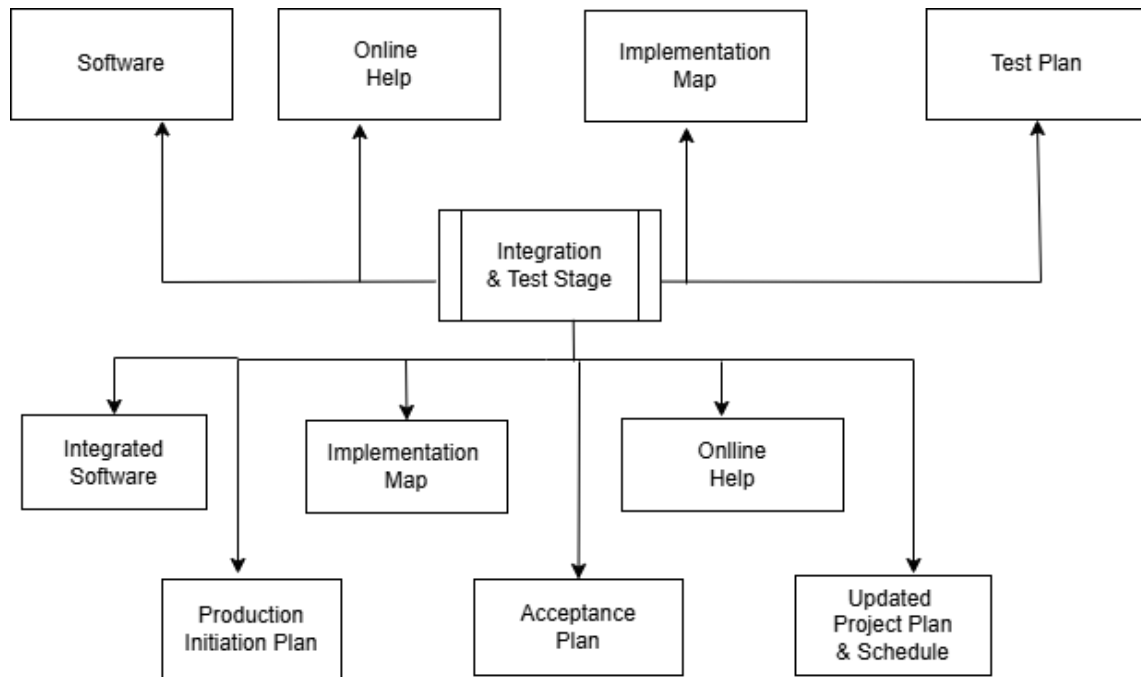


Fig 5.6 Integration and Testing Stage

5.7 INSTALLATION & ACCEPTANCE TEST

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.

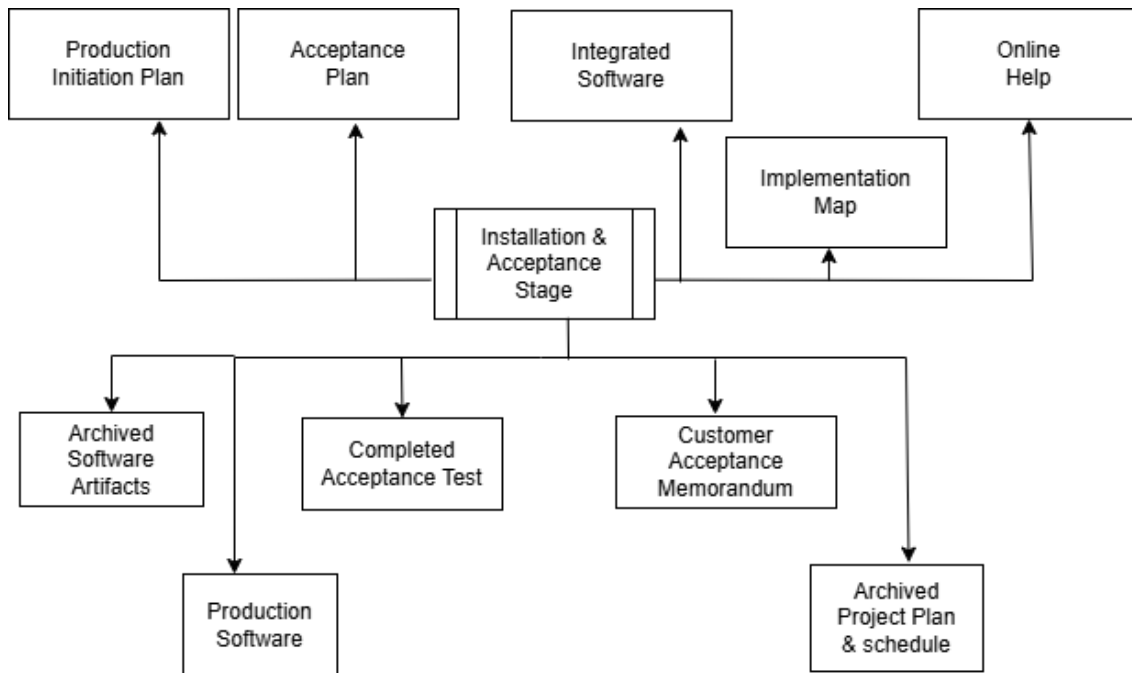


Fig 5.7 Installation

5.8 MAINTENANCE

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category.

6. SYSTEM DESIGN AND UML DAIGRAMS

6.1 SYSTEM ARCHITECTURE

The purpose of the design phase is to arrange an answer of the matter such as by the necessity document. This part is that the opening moves in moving the matter domain to the answer domain. The design phase satisfies the requirements of the system. The design of a system is probably the foremost crucial issue warm heartedness the standard of the software package. It's a serious impact on the later part, notably testing and maintenance.

The output of this part is that the style of the document. This document is analogous to a blueprint of answer and is employed later throughout implementation, testing and maintenance. The design activity is commonly divided into 2 separate phases System Design and Detailed Design.

System Design conjointly referred to as top-ranking style aims to spot the modules that ought to be within the system, the specifications of those modules, and the way them move with one another to supply the specified results.

At the top of the system style all the main knowledge structures, file formats, output formats, and also the major modules within the system and their specifications square measure set. System design is that the method or art of process the design, components, modules, interfaces, and knowledge for a system to satisfy such as needs. Users will read it because the application of systems theory to development.

Detailed Design, the inner logic of every of the modules laid out in system design is determined. Throughout this part, the small print of the info of a module square measure sometimes laid out in a high-level style description language that is freelance of the target language within which the software package can eventually be enforced.

In system design the main target is on distinguishing the modules, whereas throughout careful style the main target is on planning the logic for every of the modules.

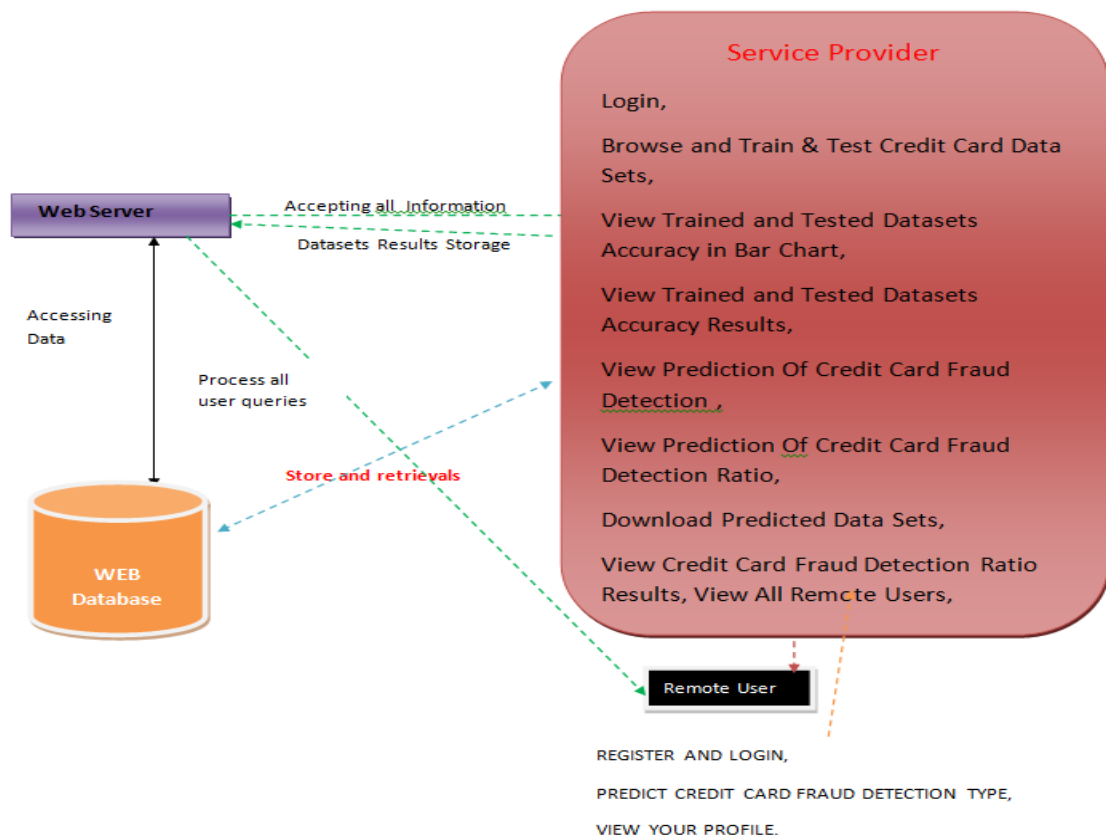


Fig 6.1.1 Architecture diagram for Credit Card Fraud

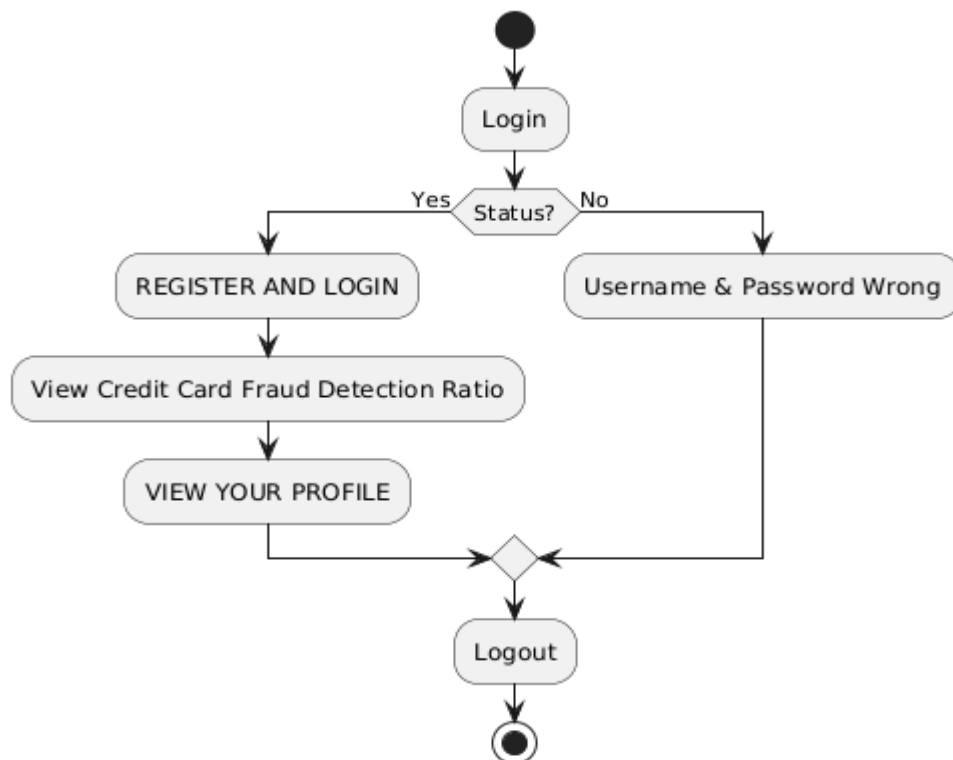


Fig 6.1.2 Flow Chart: Remote User for Credit Card Fraud

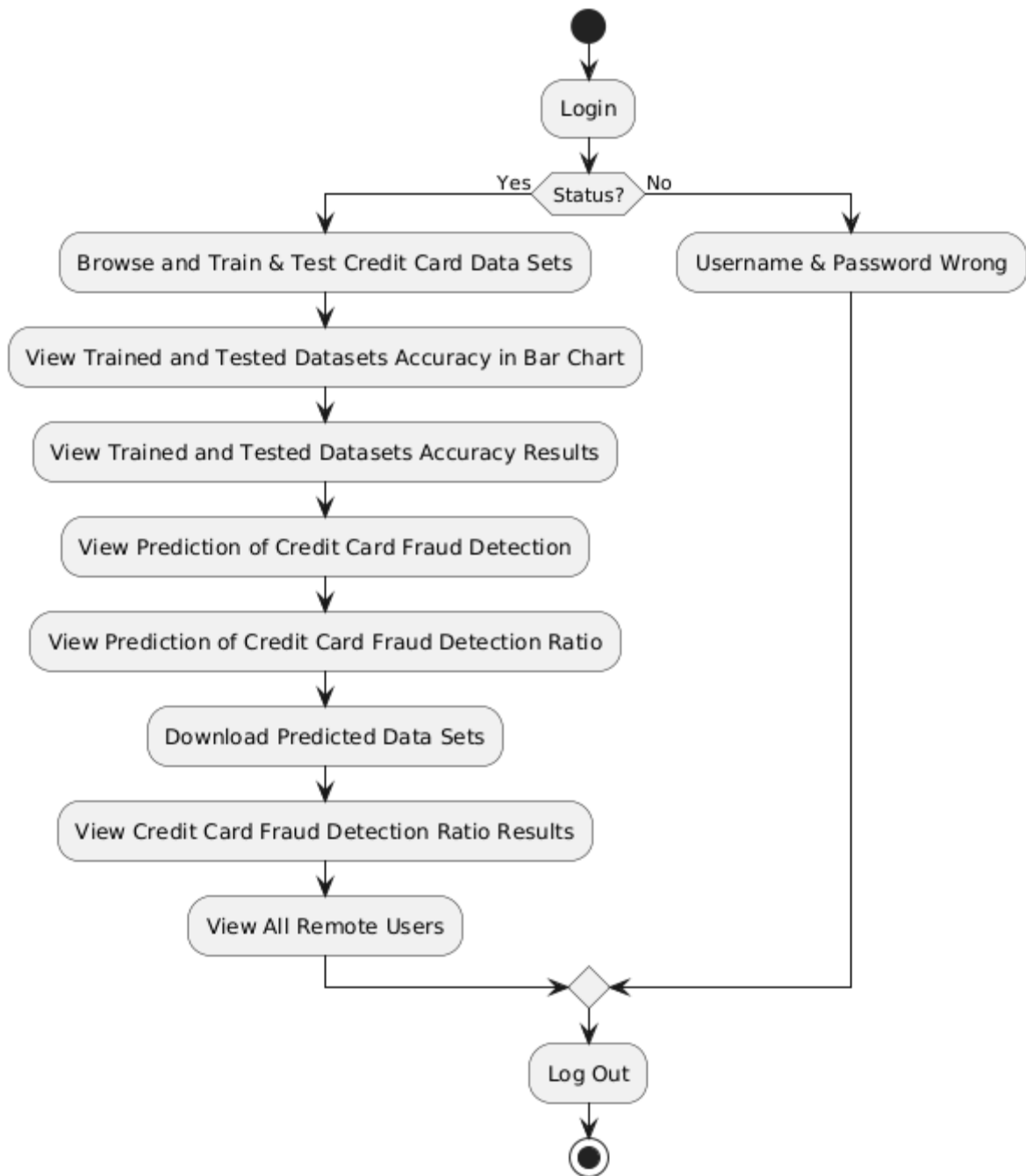


Fig 6.1.3 Flow Chart: Service Provider for Credit Card Fraud

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Goals: The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

6.2 CLASS DIAGRAM

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

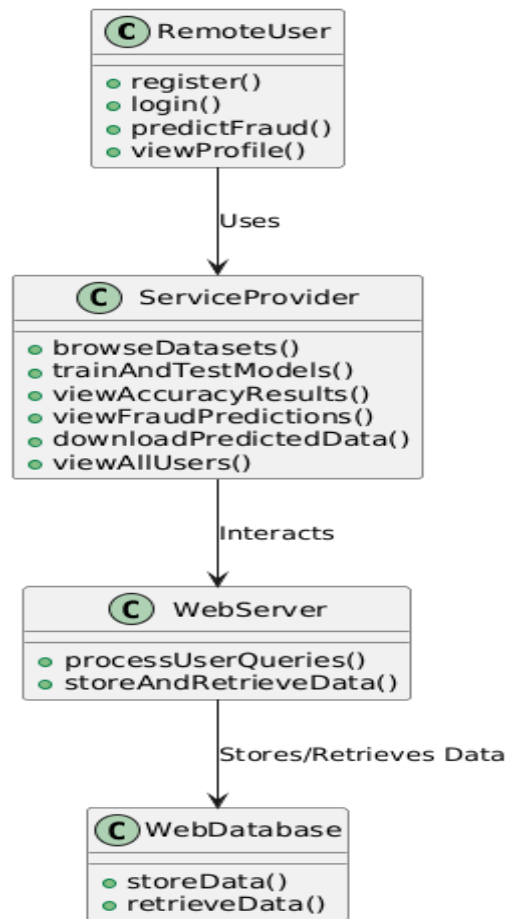


Fig 6.2 class Diagram for fraud detection

6.3 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

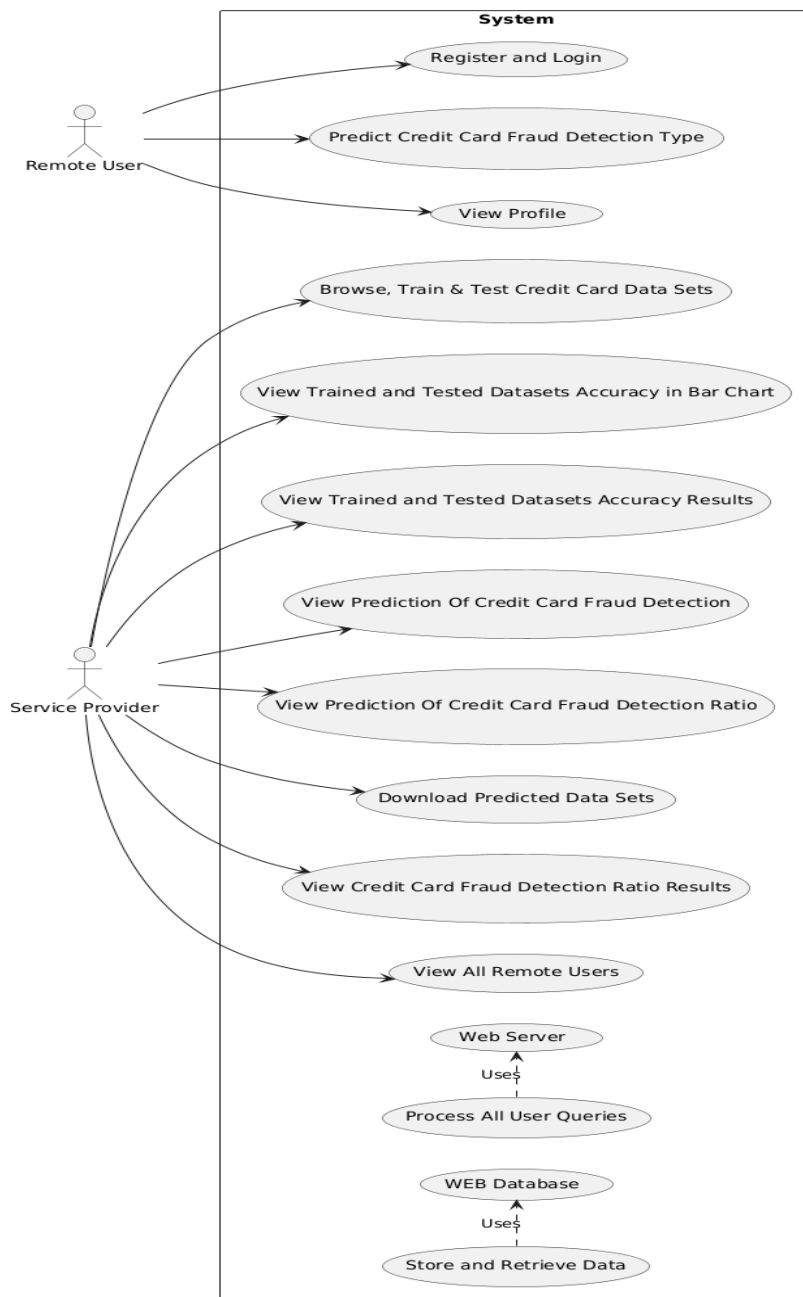


Fig 6.3 Use Case Diagram for fraud detection

6.4 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines ("lifelines"), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

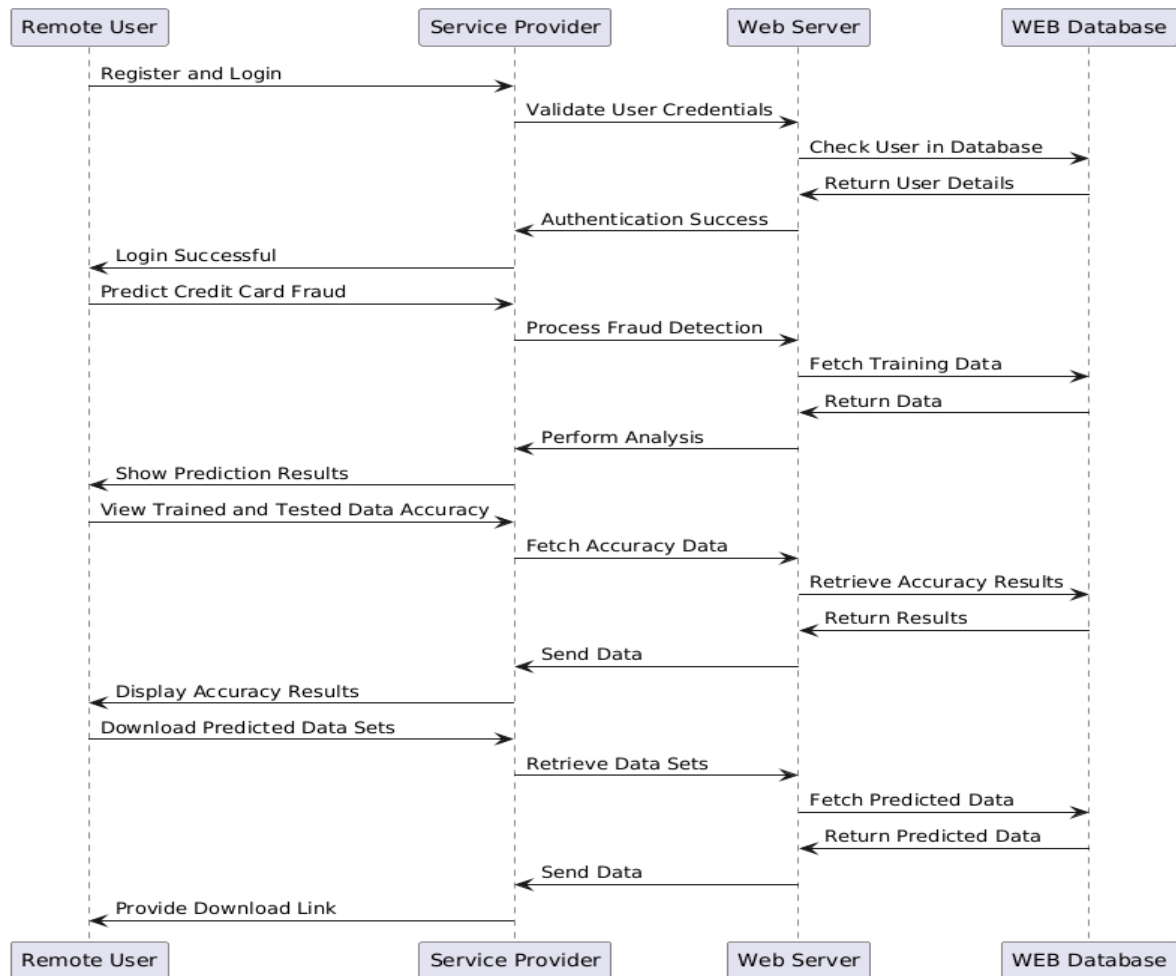


Fig 6.4 Sequence Diagram for fraud detection

6.5 ACTIVITY DIAGRAM

In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

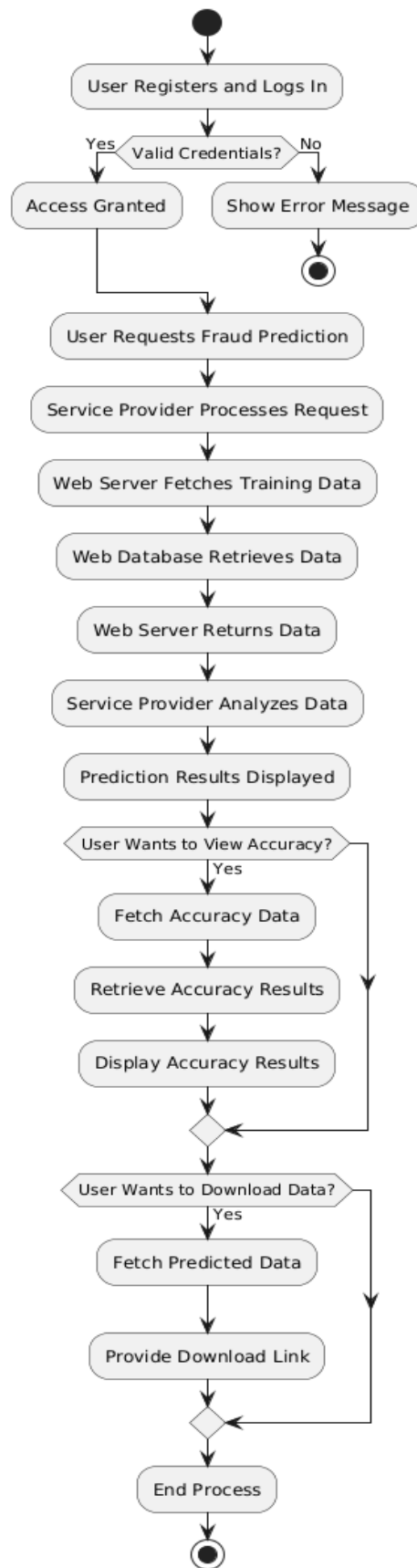


Fig 6.5 Activity Diagram for fraud detection

7. MACHINE LEARNING

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

7.1 CATEGORIES OF MACHINE LEARNING:

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data.

7.2 NEED FOR MACHINE LEARNING

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can't do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

7.3 CHALLENGES IN MACHINE LEARNING

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are

1. Quality of data – Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
2. Time-Consuming task – Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
3. Lack of specialist persons – As ML technology is still in its infancy stage, availability of expert resources is a tough job.
4. No clear objective for formulating business problems – Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
5. Issue of overfitting & underfitting – If the model is overfitting or underfitting, it cannot be represented well for the problem.

7.4 APPLICATIONS OF MACHINE LEARNING

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping

7.5 ADVANTAGES & DISADVANTAGES OF ML

Advantages:

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own.

3. Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

Disadvantages:

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

7.6 MACHINE LEARNING ALGORITHMS

Decision Tree Classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision-making knowledge from the supplied data. Decision tree can be generated from training sets.

The procedure for such generation based on the set of objects (S), each belonging to one of the classes C_1, C_2, \dots, C_k is as follows:

Step 1. If all the objects in S belong to the same class, for example C_i , the decision tree for S consists of a leaf labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O_1, O_2, \dots, O_n . Each object in S has one outcome for T so the test partitions S into subsets S_1, S_2, \dots, S_n where each object in S_i has outcome O_i for T . T becomes the root of the decision tree and for each outcome O_i we build a subsidiary decision tree by invoking the same procedure recursively on the set S_i .

Gradient Boosting

It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision tree. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differential loss function.

K-Nearest Neighbors (KNN)

- Simple, but a very powerful classification algorithm
- Classifies based on a similarity measure
- Non-parametric
- Lazy learning
- Does not “learn” until the test example is given
- Whenever we have a new data to classify, we find its K-nearest neighbors from the training data

Example

- Training dataset consists of k-closest examples in feature space
- Feature space means, space with categorization variables (non-metric variables)
- Learning based on instances, and thus also works lazily because instance close to the input vector for test or prediction may take time to occur in the training dataset.

Logistic Regression Classifiers

Logistic regression analysis examines the relationship between a categorical dependent variable and a set of independent variables. The term "logistic regression" is used when the dependent variable has only two possible values, such as 0 and 1 or

Yes and No. Logistic regression is often preferred over discriminant analysis for analyzing categorical-response variables, as it does not assume a normal distribution of independent variables. This makes it a more versatile and widely applicable method. The logistic regression model computes both binary and multinomial logistic regression on numerical and categorical independent variables. It provides insights into the regression equation, goodness of fit, odds ratios, confidence limits, likelihood, and deviance. Additionally, it includes comprehensive residual analysis with diagnostic reports and plots. The model can perform an independent variable subset selection search to identify the best regression model with the fewest independent variables. It also offers confidence intervals for predicted values and generates ROC curves to determine the optimal cutoff point for classification. Furthermore, it supports validation by automatically classifying rows that were not included in the analysis, ensuring robust and reliable results.

Naive Bayes

It is a supervised learning method based on a simplistic yet effective assumption: it considers that the presence or absence of a particular feature in a class is independent of other features. Despite this strong assumption, the Naïve Bayes classifier remains robust and efficient, often performing competitively with other supervised learning techniques. One explanation for its success is its representation bias, as it functions as a linear classifier similar to logistic regression, linear discriminant analysis, and support vector machines (SVM). However, it differs in its method of estimating classifier parameters, known as learning bias. Although widely used in research due to its simplicity, fast learning speed, and reasonable accuracy. To address this limitation, a new way of presenting the learning process results is introduced, making the classifier more understandable and easier to deploy. The theoretical aspects of Naïve Bayes are first explored before implementing the approach using the Tanagra dataset. The results are then compared with those from other linear classifiers such as logistic regression, linear discriminant analysis, and SVM, demonstrating consistency in performance.

Random Forest

It is, also known as random decision forests, are ensemble learning methods used for classification, regression, and other tasks. They operate by constructing multiple decision trees during training, where for classification tasks, the class selected by the majority of trees is chosen, and for regression tasks, the average prediction of the trees

is returned. While they generally outperform individual decision trees, their accuracy is often lower than gradient-boosted trees, although this can vary depending on data characteristics. The first random forest algorithm was introduced by Tin Kam Ho in 1995, utilizing the random subspace method, which was a way to implement the stochastic discrimination approach proposed by Eugene Kleinberg. Further developments were made by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006. Their extension combined Breiman's "bagging" technique with random feature selection, first introduced by Ho and later independently by Amit and Geman. Due to their ability to provide reliable predictions with minimal configuration, random forests are widely used as "black-box" models in businesses across various industries.

Support Vector Machine (SVM)

Classification tasks, discriminant machine learning techniques aim to find a function that accurately predicts labels for new instances based on an independent and identically distributed (iid) training dataset. Unlike generative machine learning approaches, which compute conditional probability distributions, discriminant classifiers assign data points to predefined classes without estimating underlying distributions. Although generative approaches are more powerful for outlier detection, discriminant methods require fewer computational resources and less training data, especially when dealing with high-dimensional feature spaces. SVM is a discriminant classification technique that solves convex optimization problems analytically, ensuring it always returns the same optimal hyperplane parameters. This characteristic differentiates SVM from genetic algorithms (GAs) and perceptrons, both of which are commonly used for classification but depend heavily on initialization and termination criteria. Unlike perceptrons and GA-based classifiers, which yield different models with each training session, SVM produces uniquely defined model parameters for a given training dataset when using a specific kernel transformation from input space to feature space.

8. SOFTWARE ENVIRONMENT

8.1 WHAT IS PYTHON?

Python is currently the most widely used multi-purpose, high-level programming language. It supports both Object-Oriented and Procedural programming paradigms, making it versatile and easy to use. Python programs are generally smaller compared to those written in other languages like Java, as it requires less typing and enforces indentation, which enhances code readability. Due to its simplicity and efficiency, Python is widely adopted by major tech giants such as Google, Amazon, Facebook, Instagram, Dropbox, and Uber. It is also extensively used in fields like data science, artificial intelligence, and web development.

The biggest strength of Python is huge collection of standard libraries which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contains code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory.

8. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

9. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

10. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution.

In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client- side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

3. Design Restrictions

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

8.2 HISTORY OF PYTHON

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners¹, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm

indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it. Some changes in Python 7.3:

- Print is now a function.
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behavior.

Purpose

We demonstrated that our approach enables successful segmentation of intra-retinal layers— even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

Python

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

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

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

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

8.3 MODULES USED IN PROJECT

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

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

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

Pandas

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

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object-oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

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

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

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

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background.

8.4 INSTALL PYTHON STEP-BY-STEP IN WINDOWS

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high- level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www.python.org>



Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.



Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPC
Cropped source tarball	Source release		68111671a5b2db4aef7b9ab01b7079be	23017663	360
XZ compressed source tarball	Source release		d33e4aa66097051c3eca45ee3604803	17133432	360
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.5 and later	6428b4fa7583daf1a4c2ba8ce00e6	34898416	360
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5dd605c38217a45773b9ea936b2a1f	28882845	360
Windows help file	Windows		063999573a2e98b2ac3d4ade0b4f7d12	8131761	360
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64/x64	9609c3b7d2ee0b6a0e02184a6728a2	7504391	360
Windows x86-64 executable installer	Windows	for AMD64/EM64/x64	a702b4b0aaf76d4b0b35c3a5d3e5d3400	26882948	360
Windows x86-64 web-based installer	Windows	for AMD64/EM64/x64	28c31c000b6f73a0e651a3b4351b4bd2	1362904	360
Windows x86 embeddable zip file	Windows		9fab38d19841d79fda9412574139d0	6741626	360
Windows x86 executable installer	Windows		33c2802942a54446a3d8451476394789	25663848	360
Windows x86 web-based installer	Windows		1b670cfa5d117d83c30983ea371d87c	1324608	360

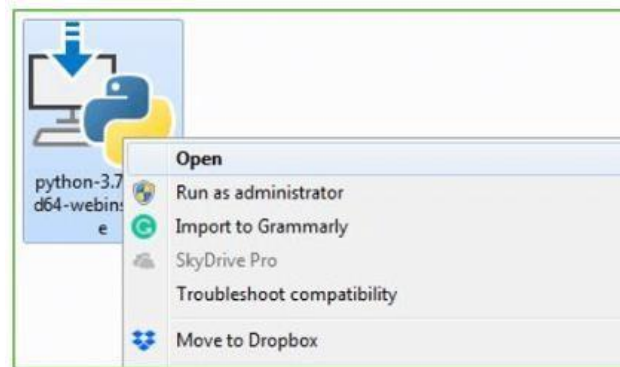
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e., Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, make sure to put a tick on Add Python 3.7 to PATH.



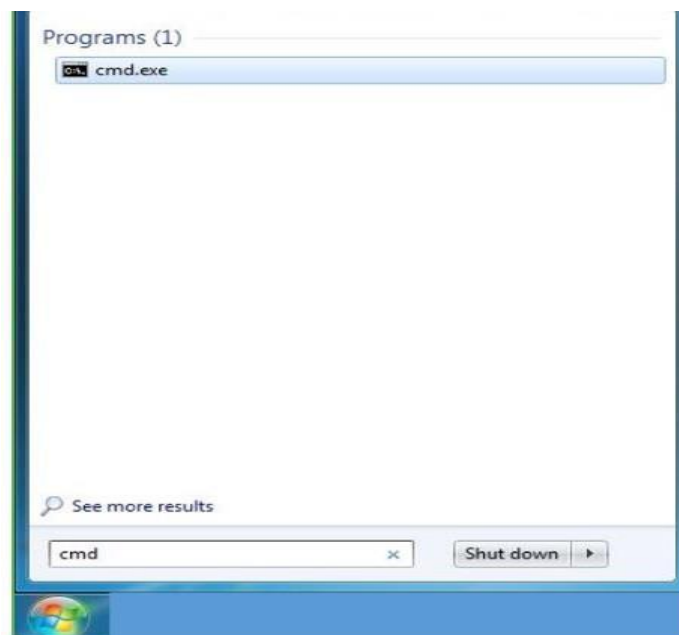
Step 3: Click on Install NOW After the installation is successful. Click on Close.



With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

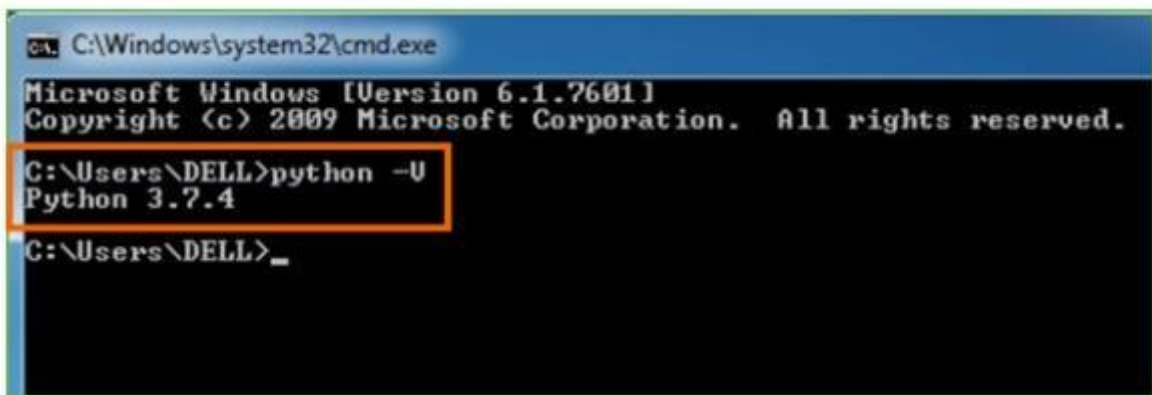
Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python -V and press Enter.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\DELL>python -U
Python 3.7.4

C:\Users\DELL>_
```

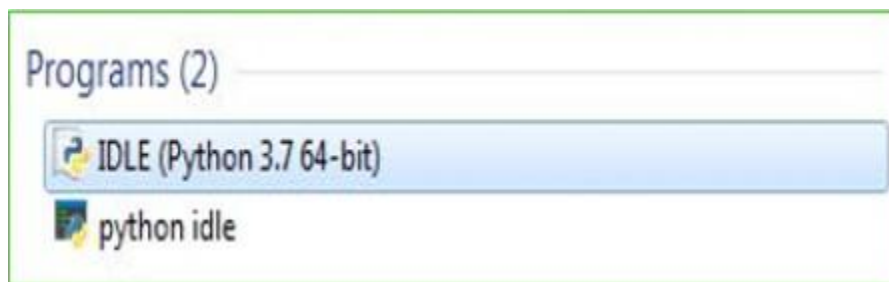
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE

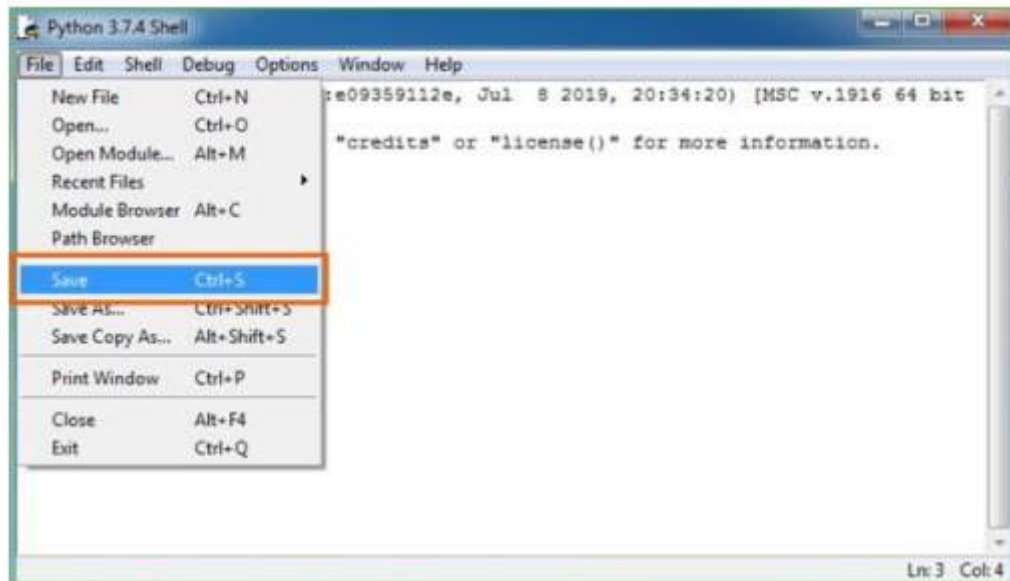
works Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.



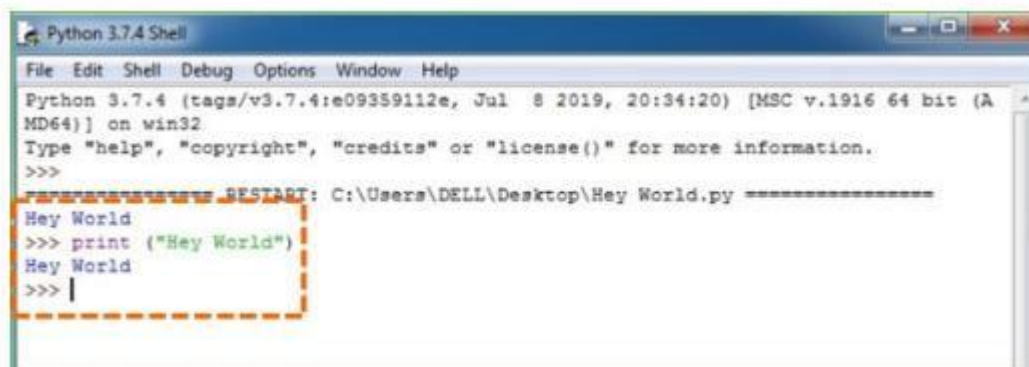
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g., enter print (“Hey World”) and Press Enter.



You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

9. SYSTEM REQUIREMENTS SPECIFICATIONS

9.1 SOFTWARE REQUIREMENTS

The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

- Operating system : Windows 10/11
- Coding Language : Python 3.7

9.2 HARDWARE REQUIREMENTS

The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

- System : Intel Core i5/i7 (11th Gen or later) / AMD Ryzen 5/7
- Hard Disk : 512 GB SSD (or higher)
- Monitor : 24-inch FHD/IPS Display
- Mouse : Any optical mouse (e.g., Logitech, Razer)
- RAM : 8 GB or more

10. FUNCTIONAL REQUIREMENTS

10.1 OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

10.2 INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input Stages

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

Input Types

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As Input data is to be directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

Error Avoidance

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

Data Validation

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary. The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

10.3 USER INTERFACE DESIGN

It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Classified As:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Interfaces

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

10.4 PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

11. SYSTEM TESTING

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system properly function as a unit. The test data should be chosen such that it passed through all possible condition.

11.1 SYSTEM TESTING

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. The program was tested logically and pattern of execution of the program for a set of data are repeated.

11.2 MODULE TESTING

To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

11.3 INTEGRATION TESTING

After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system

11.4 ACCEPTANCE TESTING

When that user find no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management.

12. SOURCE CODE

```
from django.db.models import Count, Avg
from django.shortcuts import render, redirect
from django.db.models import Count
from django.db.models import Q
import datetime
import xlwt
from django.http import HttpResponse

import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier

# Create your views here.
from Remote_User.models import
ClientRegister_Model, cc_fraud_detection_type, detection_ratio, detection_accuracy
def serviceproviderlogin(request):
    if request.method == "POST":
        admin = request.POST.get('username')
        password = request.POST.get('password')
        if admin == "Admin" and password == "Admin":
            detection_accuracy.objects.all().delete()
            return redirect('View_Remote_Users')
        return render(request, 'SProvider/serviceproviderlogin.html')

def View_CC_Fraud_Detection_Ratio(request):
    detection_ratio.objects.all().delete()
    ratio = ""
    kword = 'No Credit Card Fraud'
    print(kword)
    obj = cc_fraud_detection_type.objects.all().filter(Q(Prediction=kword))
```

```

obj1 = cc_fraud_detection_type.objects.all()
count = obj.count();
count1 = obj1.count();
ratio = (count / count1) * 100

if ratio != 0:
    detection_ratio.objects.create(names=kword, ratio=ratio)
ratio12 = ""
kword12 = 'Credit Card Fraud'
print(kword12)
obj12 = cc_fraud_detection_type.objects.all().filter(Q(Prediction=kword12))
obj112 = cc_fraud_detection_type.objects.all()
count12 = obj12.count();
count112 = obj112.count();
ratio12 = (count12 / count112) * 100

if ratio12 != 0:
    detection_ratio.objects.create(names=kword12, ratio=ratio12)
obj = detection_ratio.objects.all()
return render(request, 'SProvider/View_CC_Fraud_Detection_Ratio.html', {'objs': obj})

def View_Remote_Users(request):
    obj=ClientRegister_Model.objects.all()
    return render(request,'SProvider/View_Remote_Users.html',{'objects':obj})

def charts(request,chart_type):
    chart1 = detection_ratio.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts.html", {'form':chart1, 'chart_type':chart_type})

def charts1(request,chart_type):
    chart1 = detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/charts1.html", {'form':chart1, 'chart_type':chart_type})

def View_Prediction_Of_CC_Fraud_Detection(request):
    obj =cc_fraud_detection_type.objects.all()

```

```

    return render(request, 'SProvider/View_Prediction_Of_CC_Fraud_Detection.html',
{'list_objects': obj})

def likeschart(request,like_chart):
    charts =detection_accuracy.objects.values('names').annotate(dcount=Avg('ratio'))
    return render(request,"SProvider/likeschart.html", {'form':charts, 'like_chart':like_chart})

def Download_Predicted_DataSets(request):
    response = HttpResponse(content_type='application/ms-excel')
    # decide file name
    response['Content-Disposition'] = 'attachment; filename="Predicted_Datasets.xls"'
    # creating workbook
    wb = xlwt.Workbook(encoding='utf-8')
    # adding sheet
    ws = wb.add_sheet("sheet1")
    # Sheet header, first row
    row_num = 0
    font_style = xlwt.XFStyle()
    # headers are bold
    font_style.font.bold = True
    # writer = csv.writer(response)
    obj = cc_fraud_detection_type.objects.all()
    data = obj # dummy method to fetch data.

    for my_row in data:
        row_num = row_num + 1
        ws.write(row_num, 0, my_row.trans_date, font_style)
        ws.write(row_num, 1, my_row.cc_num, font_style)
        ws.write(row_num, 2, my_row.category, font_style)
        ws.write(row_num, 3, my_row.AMT_TRANS, font_style)
        ws.write(row_num, 4, my_row.first, font_style)
        ws.write(row_num, 5, my_row.last, font_style)
        ws.write(row_num, 6, my_row.gender, font_style)
        ws.write(row_num, 7, my_row.street, font_style)
        ws.write(row_num, 8, my_row.city, font_style)

```

```

ws.write(row_num, 9, my_row.state, font_style)
ws.write(row_num, 10, my_row.zip, font_style)
ws.write(row_num, 11, my_row.User_Lat, font_style)
ws.write(row_num, 12, my_row.User_Long, font_style)
ws.write(row_num, 13, my_row.city_pop, font_style)
ws.write(row_num, 14, my_row.Job, font_style)
ws.write(row_num, 15, my_row.Dob, font_style)
ws.write(row_num, 16, my_row.trans_num, font_style)
ws.write(row_num, 17, my_row.merch_lat, font_style)
ws.write(row_num, 18, my_row.merch_long, font_style)
ws.write(row_num, 19, my_row.Prediction, font_style)
wb.save(response)
return response

```

```

def train_model(request):
    detection_accuracy.objects.all().delete()
    df = pd.read_csv('CC_Datasets.csv')
    def apply_response(label):
        if (label== 0):
            return 0 # No Fraud
        elif (label==1):
            return 1 # Fraud
    df['results'] = df['is_fraud'].apply(apply_response)
    cv = CountVectorizer()
    X = df['street']
    y = df['results']
    print("Transaction Number")
    print(X)
    print("Results")
    print(y)
    X = cv.fit_transform(X)
    models = []

    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

```

```

X_train.shape, X_test.shape, y_train.shape
print(X_test)
print("Naive Bayes")
from sklearn.naive_bayes import MultinomialNB
NB = MultinomialNB()
NB.fit(X_train, y_train)
predict_nb = NB.predict(X_test)
naivebayes = accuracy_score(y_test, predict_nb) * 100
print(naivebayes)
print(confusion_matrix(y_test, predict_nb))
print(classification_report(y_test, predict_nb))
models.append(('naive_bayes', NB))
detection_accuracy.objects.create(names="Naive Bayes", ratio=naivebayes)

# SVM Model
print("SVM")
from sklearn import svm
lin_clf = svm.LinearSVC()
lin_clf.fit(X_train, y_train)
predict_svm = lin_clf.predict(X_test)
svm_acc = accuracy_score(y_test, predict_svm) * 100
print(svm_acc)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, predict_svm))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, predict_svm))
models.append(('svm', lin_clf))
detection_accuracy.objects.create(names="SVM", ratio=svm_acc)
print("Logistic Regression")
from sklearn.linear_model import LogisticRegression
reg = LogisticRegression(random_state=0, solver='lbfgs').fit(X_train, y_train)
y_pred = reg.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, y_pred) * 100)
print("CLASSIFICATION REPORT")

```



```

print(classification_report(y_test, y_pred))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, y_pred))
models.append(('logistic', reg))
detection_accuracy.objects.create(names="Logistic Regression",
ratio=accuracy_score(y_test, y_pred) * 100)

print("Decision Tree Classifier")
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
dtcpredict = dtc.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, dtcpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, dtcpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, dtcpredict))
models.append(('DecisionTreeClassifier', dtc))
detection_accuracy.objects.create(names="Decision Tree Classifier",
ratio=accuracy_score(y_test, dtcpredict) * 100)
print("Gradient Boosting Classifier")

from sklearn.ensemble import GradientBoostingClassifier
clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1,
random_state=0).fit(
    X_train,
    y_train)
clfpredict = clf.predict(X_test)
print("ACCURACY")
print(accuracy_score(y_test, clfpredict) * 100)
print("CLASSIFICATION REPORT")
print(classification_report(y_test, clfpredict))
print("CONFUSION MATRIX")
print(confusion_matrix(y_test, clfpredict))

```

```
models.append(('GradientBoostingClassifier', clf))
detection_accuracy.objects.create(names="Gradient Boosting Classifier",
                                   ratio=accuracy_score(y_test, clfpredict) * 100)

csv_format = 'Results.csv'
df.to_csv(csv_format, index=False)
df.to_markdown

obj = detection_accuracy.objects.all()
return render(request, 'SProvider/train_model.html', {'objs': obj})
```

13. RESULTS

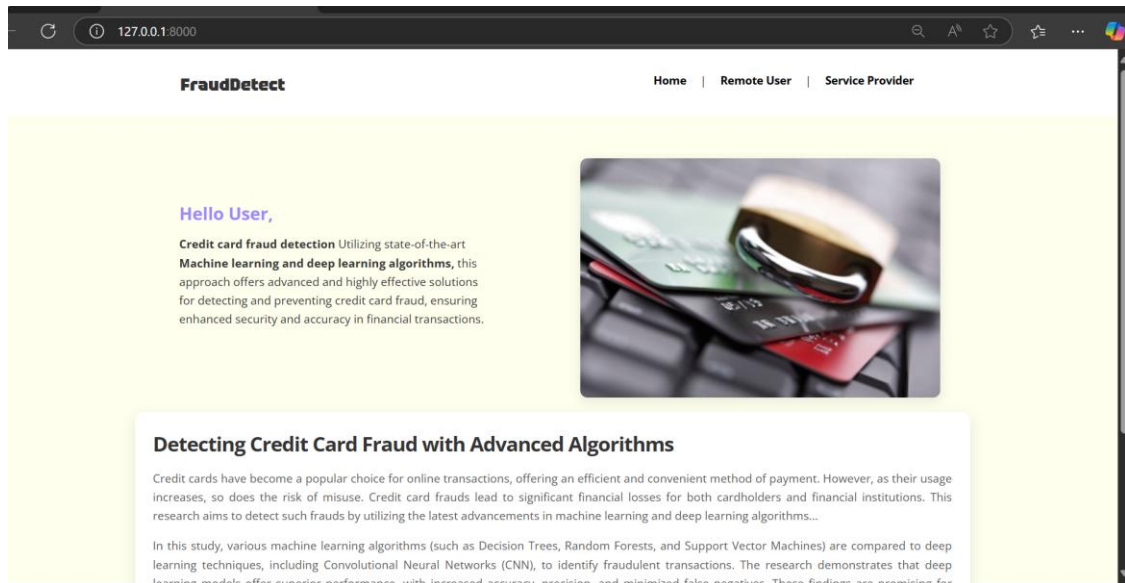


Fig 13.1 Credit Card Fraud Detection Web page

The screenshot shows the 'VIEW ALL REMOTE USERS' page in the FraudDetect application. The page has a purple header with the application name and several navigation links: 'Train & Test Data', 'Bar Chart Accuracy', 'Accuracy Results', 'Fraud Prediction', 'Fraud Detection Ratio', 'Download Data', 'Fraud Ratio Results', 'Remote Users' (which is highlighted), and 'Logout'. Below the header, there is a table listing remote users with columns for USER NAME, EMAIL, GENDER, ADDRESS, MOBILE NO, COUNTRY, STATE, and CITY. The table contains five rows of user data.

USER NAME	EMAIL	GENDER	ADDRESS	MOBILE NO	COUNTRY	STATE	CITY
Karthik	Karthik123@gmail.com	Male	#7822,11th Cross,Rajajinagar	9535866270	India	Karnataka	Bangalore
Manjunath	tmksmanju13@gmail.com	Male	#892,4th Cross,Viajyanagar	9535866270	India	Karnataka	Bangalore
Meghana	meghaspeakss@gmail.com	Female	Aswapuram,Bhadradi Kothagudem	8519966983	India	Telangana	Kothagudem
Shruthi	g.shruthi@cmrec.ac.in	Female	kompally	8998996697	India	Telangana	Hyderabad
VRK	525252@gmail.com	Male	Bhadrachalam	8519966983	India	Telangana	kothagudem

Fig 13.2 List of Remote Users

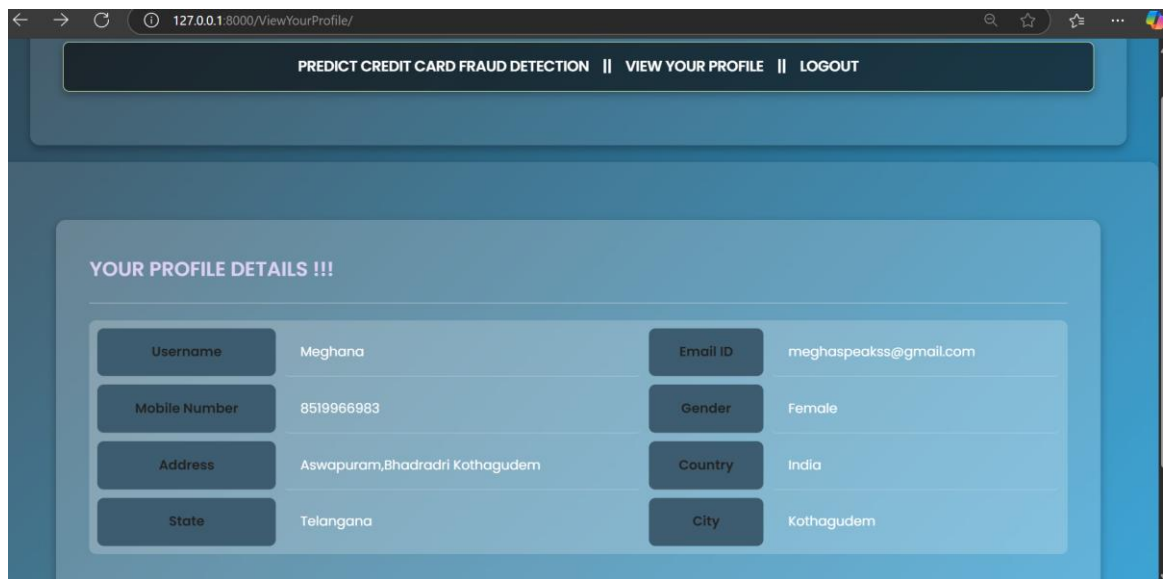


Fig 13.3 Registered Status of Users

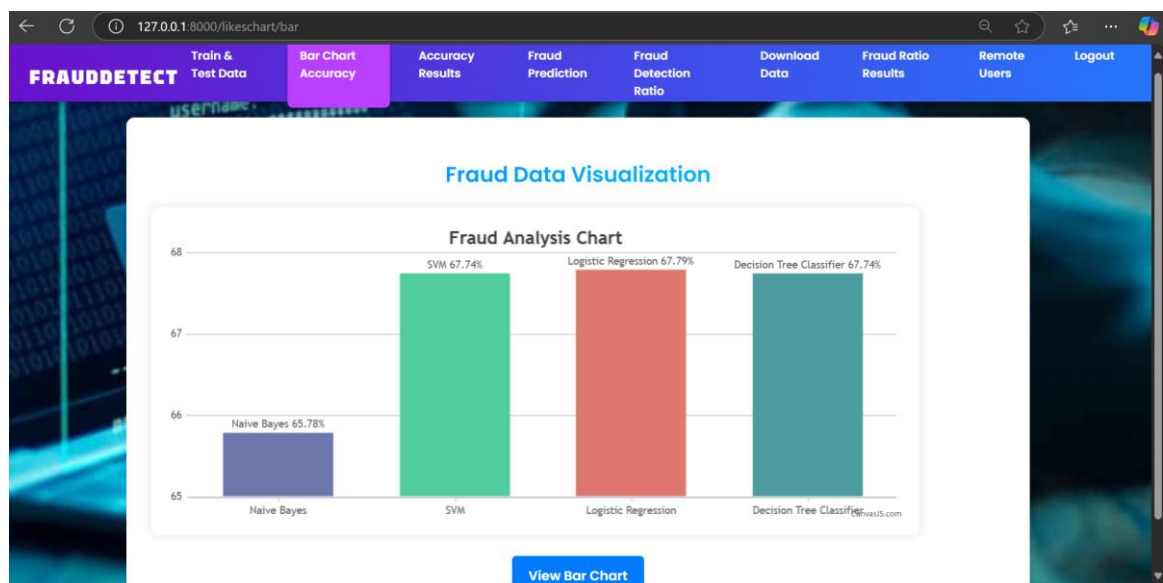


Fig 13.4 Bar Chart of Credit card Fraud

FRAUDETECT Train & Test Data Bar Chart Accuracy Accuracy Results **Fraud Prediction** Fraud Detection Ratio Download Data Fraud Ratio Results Remote Users Logout

View Credit Card Fraud Prediction Details

Transaction Date	CC Number	Category	Amount	First Name	Last Name	Gender	Street	
21-06-20 12:19	4910000000000000	kids_pets	270000	Lauren	Torres	F	03030 White Lakes	Gra
21-06-20 12:21	2130000000000000	travel	239850	Rebecca	Conley	F	181 Moreno Light Apt. 215	Ton
21-06-20 12:23	6010000000000000	health_fitness	979992	William	Johnson	M	50843 Vincent Mission	S Lond
21-06-20 13:21	3590000000000000	health_fitness	364896	Crystal	Fuller	F	000 Jennifer Mills	Iss
21-06-2020	2290000000000000	personal_care	406597.5	Jeff	Elliott	M	351 Darlene Green	Co

Fig 13.5 Prediction of Credit Card Fraud Detection

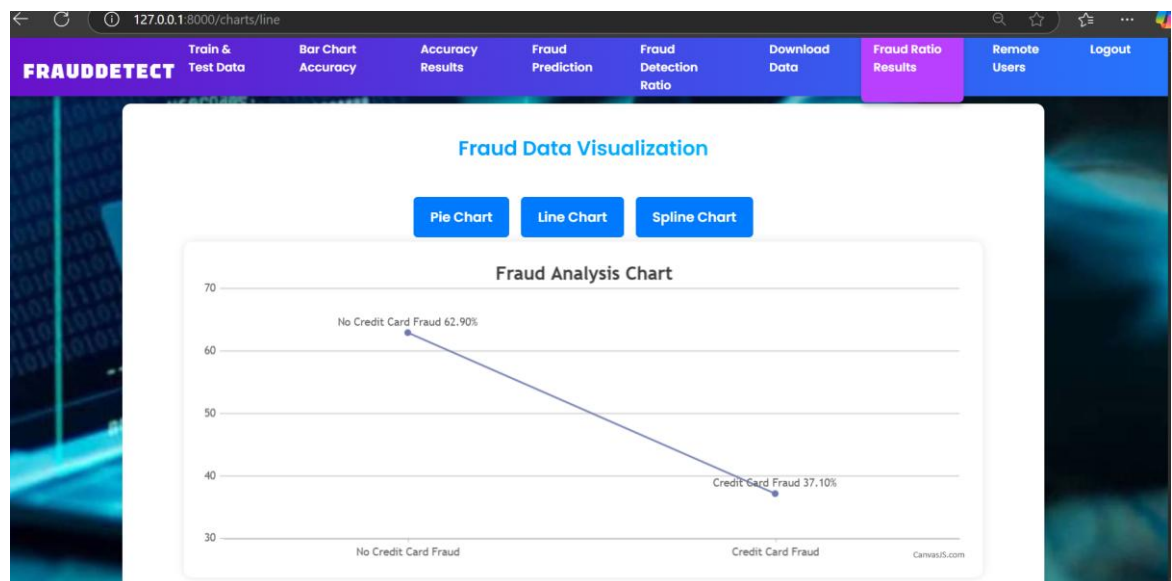


Fig 13.6 Line Chart of Credit Card Fraud Detection

14. CONCLUSION AND FUTURE SCOPE

CONCLUSION

In conclusion, CCF is an increasing threat to financial institutions. Fraudsters tend to constantly come up with new fraud methods. A robust classifier can handle the changing nature of fraud. Accurately predicting fraud cases and reducing false-positive cases is the foremost priority of a fraud detection system. The performance of ML methods varies for each individual business case. The type of input data is a dominant factor that drives different ML methods. For detecting CCF, the number of features, number of transactions, and correlation between the features are essential factors in determining the model's performance. DL methods, such as CNNs and their layers, are associated with the processing of text and the baseline model. Using these methods for the detection of credit cards yields better performance than traditional algorithms. Numerous sampling techniques are used to increase the performance of existing examples, but they significantly decrease on the unseen data. The performance on unseen data increased as the class imbalance increased. Future work associated may explore the use of more state of art deep learning methods to improve the performance of the model proposed in this study.

FUTURE ENHANCEMENTS

It is not possible to develop a system that meets all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As technology emerges, it is possible to upgrade the system and make it adaptable to the desired environments.
- Based on future security issues, security can be improved using emerging technologies like single sign-on.

REFERENCES

- [1] Y. Abakarim, M. Lahby, and A. Attioui, "An efficient real time model for credit card fraud detection based on deep learning," in *Proc. 12th Int. Conf. Intell. Systems: Theories Appl.*, Oct. 2018, pp. 1_7, doi: 10.1145/3289402.3289530.
- [2] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Inter- discipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433_459, Jul. 2010, doi: 10.1002/wics.101.
- [3] V. Arora, R. S. Leekha, K. Lee, and A. Kataria, "Facilitating user authorization from imbalanced data logs of credit cards using arti_cial intelligence," *Mobile Inf. Syst.*, vol. 2020, pp. 1_13, Oct. 2020, doi: 10.1155/2020/8885269.
- [4] A. O. Balogun, S. Basri, S. J. Abdulkadir, and A. S. Hashim, "Performance analysis of feature selection methods in software defect prediction: A search method approach," *Appl. Sci.*, vol. 9, no. 13, p. 2764, Jul. 2019, doi: 10.3390/app9132764.
- [5] B. Bandaranayake, "Fraud and corruption control at education system level: A case study of the Victorian department of education and early childhood development in Australia," *J. Cases Educ. Leadership*, vol. 17, no. 4, pp. 34_53, Dec. 2014, doi: 10.1177/1555458914549669.
- [6] J. B^aaszczy«ski, A. T. de Almeida Filho, A. Matuszyk, M. Szelg_, and R. S^aowi«ski, "Auto loan fraud detection using dominance-based rough set approach versus machine learning methods," *Expert Syst. Appl.*, vol. 163, Jan. 2021, Art. no. 113740, doi: 10.1016/j.eswa.2020.113740.
- [7] B. Branco, P. Abreu, A. S. Gomes, M. S. C. Almeida, J. T. Ascens«o, and P. Bizarro, "Interleaved sequence RNNs for fraud detection," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 3101_3109, doi: 10.1145/3394486.3403361.
- [8] F. Cartella, O. Anunciacao, Y. Funabiki, D. Yamaguchi, T. Akishita, and O. Elshocht, "Adversarial attacks for tabular data: Application to fraud detection and imbalanced data," 2021, arXiv:2101.08030.
- [9] S. S. Lad, I. Dept. of CSERajarambapu Institute of TechnologyRajaramnagarSangliMaharashtra, and A. C. Adamuthe, "Malware classi_cation with improved convolutional neural network model," *Int. J. Comput. Netw. Inf. Secur.*, vol. 12, no. 6, pp. 30_43, Dec. 2021, doi: 10.5815/ijcnis.2020.06.03.
- [10] V. N. Dornadula and S. Geetha, "Credit card fraud detection using machine learning algorithms," *Proc. Comput. Sci.*, vol. 165, pp. 631_641, Jan. 2019, doi: 10.1016/j.procs.2020.01.057.

- [11] I. Benchaji, S. Douzi, and B. E. Ouahidi, "Credit card fraud detection model based on LSTM recurrent neural networks," *J. Adv. Inf. Technol.*, vol. 12, no. 2, pp. 113_118, 2021, doi: 10.12720/jait.12.2.113-118.
- [12] Y. Fang, Y. Zhang, and C. Huang, "Credit card fraud detection based on machine learning," *Comput., Mater. Continua*, vol. 61, no. 1, pp. 185_195, 2019, doi: 10.32604/cmc.2019.06144.
- [13] J. Forough and S. Momtazi, "Ensemble of deep sequential models for credit card fraud detection," *Appl. Soft Comput.*, vol. 99, Feb. 2021, Art. no. 106883, doi: 10.1016/j.asoc.2020.106883.
- [14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, arXiv:1512.03385.
- [15] X. Hu, H. Chen, and R. Zhang, "Short paper: Credit card fraud detection using LightGBM with asymmetric error control," in *Proc. 2nd Int. Conf. Artif. Intell. for Industries (AII)*, Sep. 2019, pp. 91_94, doi: 10.1109/AI4I46381.2019.00030.
- [16] J. Kim, H.-J. Kim, and H. Kim, "Fraud detection for job placement using hierarchical clusters-based deep neural networks," *Int. J. Speech Technol.*, vol. 49, no. 8, pp. 2842_2861, Aug. 2019, doi: 10.1007/s10489-019-01419-2.
- [17] M.-J. Kim and T.-S. Kim, "A neural classifier with fraud density map for effective credit card fraud detection," in *Intelligent Data Engineering and Automated Learning*, vol. 2412, H. Yin, N. Allinson, R. Freeman, J. Keane, and S. Hubbard, Eds. Berlin, Germany: Springer, 2002, pp. 378_383, doi: 10.1007/3-540-45675-9_56.
- [18] N. Kousika, G. Vishali, S. Sunandhana, and M. A. Vijay, "Machine learning based fraud analysis and detection system," *J. Phys., Conf.*, vol. 1916, no. 1, May 2021, Art. no. 012115, doi: 10.1088/1742-6596/1916/1/012115.
- [19] R. F. Lima and A. Pereira, "Feature selection approaches to fraud detection in e-payment systems," in *E-Commerce and Web Technologies*, vol. 278, D. Bridge and H. Stuckenschmidt, Eds. Springer, 2017, pp. 111_126, doi: 10.1007/978-3-319-53676-7_9.
- [20] Y. Lucas and J. Jurgovsky, "Credit card fraud detection using machine learning: A survey," 2020, arXiv:2010.06479.
- [21] H. Zhou, H.-F. Chai, and M.-L. Qiu, "Fraud detection within bankcard enrollment on mobile device based payment using machine learning," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 12, pp. 1537_1545, Dec. 2018, doi: 10.1631/FITEE.1800580.