

**A Project Report**  
**on**  
**A Machine Learning Method With Hybrid Feature**  
**Selection For Improved Credit Card Fraud Detection**

**Submitted for partial fulfillment of the award of**

**BACHELOR OF TECHNOLOGY**

**In**

**Computer Science and Engineering**

**By**

**Dhulipalla Vidhayini – 20BQ1A0547**

**Gaddam Sharmila – 20BQ1A0554**

**Bathula Munna – 20BQ1A0518**

**Divyakolu Vamsi – 20BQ1A0548**

**Under the guidance of**

**Mr. China Pentu Saheb**

**Assistant Professor**



**(Autonomous)**

**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**

**Approved by AICTE, Permanently Affiliated to JNTU, Kakinada**

**Accredited by NAAC with 'A' Grade - ISO 9001:2008 Certified**

**Nambur (V), Peda Kakani (M), Guntur Dt. - 522508**

**April, 2024.**



**VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY**  
**(Autonomous)**

Permanently Affiliated to JNTU, Kakinada, Approved by AICTE  
Accredited by NAAC with 'A' Grade, ISO 9001:2015 Certified  
Nambur, Pedakakani (M), Guntur (Dt) - 522508

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**  
**B.Tech Programme Accredited by NBA**

**CERTIFICATE**

This is to certify that the project report titled “A MACHINE LEARNING METHOD WITH HYBRID FEATURE SELECTION FOR IMPROVED CREDIT CARD FRAUD DETECTION” is being submitted by DHULIPALLA VIDHAYINI, GADDAM SHARMILA, BATHULA MUNNA, DIVYAKOLU VAMSI bearing 20BQ1A0547, 20BQ1A0554, 20BQ1A0518, 20BQ1A0548 in IV B.Tech II semester *Computer Science & Engineering* is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

**Mr. China Pentu Saheb**  
**Project Guide**

**Dr. V Rama Chandran**  
**Head of the Department**

---

Signature of Internal Examiner with Date

Signature of External Examiner with Date

## **DECLARATION**

We, DHULIPALLA VIDHAYINI (20BQ1A0547), GADDAM SHARMILA (20BQ1A0554), BATHULA MUNNA (20BQ1A0518), DIVYAKOLU VAMSI (20BQ1A0548 hereby declare that the Project Report entitled “ A MACHINE LEARNING METHOD WITH HYBRID FEATURE SELECTION FOR IMPROVED CREDIT CARD FRAUD DETECTION ” done by us under the guidance of Mr. China Pentu Saheb , Assistant Professor, CSE at Vasireddy Venkatadri Institute of Technology is submitted in partial fulfillment of the requirements for the award of degree in Bachelor of Technology in Computer Science & Engineering.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE

- 1.
- 2.
- 3.
- 4.

## ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand our ideas and helped us towards the successful completion of this project work.

First and foremost, we express our deep gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the B.Tech programme.

We express our sincere thanks to **Dr. V. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the B.Tech programme.

We express our sincere gratitude to **Dr. V. Ramachandran**, Professor & HOD, Computer Science & Engineering, Vasireddy Venkatadri Institute of Technology for his constant encouragement, motivation and faith by offering different places to look to expand my ideas.

We would like to express our sincere gratefulness to our guide **Mr. China Pentu Saheb**, Assistant Professor, CSE for her insightful advice, motivating suggestions, valuable guidance, help and support in successful completion of this project.

We would like to express our sincere heartfelt thanks to our Project Coordinator **Dr. N. Lakshmi Prasanna**, Professor, CSE for her valuable advices, motivating suggestions, moral support, help and coordination among us in successful completion of this project.

I would like to take this opportunity to express my thanks to the **teaching and non-teaching staff** in Department of Computer Science & Engineering, VVIT for their valuable help and support.

# INDEX

Topics	Page No
List of Figures	iv
List of Screens	v
List of Tables	vi
Nomenclature	vii
<b>Abstract</b>	viii
<b>1. Introduction</b>	
1.1 Basic Concepts	1
1.1.1 Credit Card Fraud	1
1.1.2 Machine Learning Classifiers	2
1.1.3 Hybrid Feature-Selection Technique	3
1.1.4 Extreme Learning Machine	3
1.1.5 Ensemble Method	3
1.1.6 Voting Classifier	3
1.2 Problem Statement	4
1.3 Objectives	4
1.4 Advantages	4
<b>2. Aim and Scope</b>	
2.1 Aim	5
2.2 Scope	5
2.3 Existing System	5
2.4 Proposed System	6
2.5 Literature Survey	6
2.6 Algorithms	8
<b>3. Design</b>	
3.1 Functional Requirements	10

3.2 Non-Functional Requirements	10
3.3 Software Requirements	11
3.4 Hardware Requirements	12
3.5 Software Development Life Cycle	13
3.6 System Architecture	14
3.7 Data Flow Diagram	15
3.8 UML Diagrams	17
3.8.1 Usecase Diagram	17
3.8.2 Sequence Diagram	19
3.8.3 Class Diagram	21
3.8.4 Activity Diagram	22
3.8.5 Collaboration Diagram	23
3.8.6 Component Diagram	23
3.8.7 Deployment Diagram	25
<b>4. Implementation</b>	
4.1 Importing Libraries	26
4.2 Loading Data	26
4.3 Data Exploration	26
4.4 Fraud Detection Setup	27
4.5 Data Preparation	27
4.6 Model Definition	28
4.7 Model Fitting And Evaluation	28
<b>5. System Testing</b>	
5.1 Software Testing Strategies	30
5.2 Test Cases	33
<b>6. Results</b>	
6.1 Experimental Results	34
6.2 Graphical Representation	35

6.3 Output Screens	38
6.4 Comparison	42
<b>7. Conclusion and Summary</b>	
7.1 Conclusion	44
7.2 Future Enhancements	44
<b>References</b>	45
<b>Appendix</b>	46
1.Paper Publication	
2.Published Article	

## List of Figures

Figure No	Figure Name	Page No
3.1	Software Development Life Cycle	14
3.2	System Architecture	15
3.3	Data Flow Diagram	16
3.4	Usecase Diagram	18
3.5	Sequence Diagram	20
3.6	Class Diagram	21
3.7	Activity Diagram	22
3.8	Collaboration Diagram	23
3.9	Component Diagram	24
3.10	Deployment Diagram	25
5.1	Types of Static Testing	31
5.2	Types of Structural Testing	32
5.3	Black Box Testing	32
6.1	Tabular Representation Of Evaluation Metrics	34
6.2	Accuracy Comparison Graph	35
6.3	Precision Comparison Graph	35
6.4	Recall Comparison Graph	36
6.5	F1 Score Comparison Graph	36
6.6	G-Mean Comparison Graph	37
6.7	Specificity Comparison Graph	37



## **List of Screens**

<b>Screen No</b>	<b>Screen Name</b>	<b>Page No</b>
6.1	Home Page	38
6.2	Login Page	38
6.3	Registration Page	39
6.4	Main Page	39
6.5	About Page	40
6.6	Upload Input Values	40
6.7	Predict Result As Fraudulant Transaction	41
6.8	Predict Result as Non-Fraudulant Transaction	41

## List of Tables

Table No	Table Name	Page No
5.1	Test Cases	33
6.1	Comparison Table	42

## NOMENCLATURE

Sl.No	Acronym	Abbreviation
1	ELM	Extreme Learning Machine
2	GA	Genetic Algorithm
3	GAW	Genetic Algorithm Wrapper
4	IG	Information Gain
5	LR	Logistic Regression
6	ML	Machine Learning
7	RF	Random Forest
8	SVM	Support Vector Machine

## ABSTRACT

The burgeoning challenge of credit card fraud detection in the realm of electronic commerce and digital payments. Recognizing the surge in credit card transactions, the research focuses on optimizing machine learning (ML) classifiers by proposing a hybrid feature-selection technique. This method comprises filter and wrapper feature-selection steps, ensuring the utilization of only the most pertinent features for analysis. Various feature extraction approaches, including Full Feature, Information Gain (IG), Genetic Algorithm Wrapper (GAW), and an ensemble of IG-GAW, are employed. The study evaluates the performance of Extreme Learning Machine (ELM) and diverse ML models, such as AdaBoost, Logistic Regression, Random Forest, SVM, Decision Tree, and Voting Classifier. Notably, the extension of the research incorporates an ensemble method, demonstrating superior accuracy by combining predictions from multiple models. The Voting Classifier, in particular, attains a remarkable 100% accuracy. This research contributes significantly to the ongoing efforts in enhancing credit card fraud detection, providing valuable insights into feature selection and ensemble techniques for bolstering the robustness and accuracy of ML models in real-world scenarios.

**Keywords:** credit card; feature selection; fraud detection; genetic algorithm; machine learning; information gain; extreme learning machine;

# **CHAPTER 1**

## **INTRODUCTION**

In recent years, electronic payments (e-payments) have become ubiquitous, driven by technological advancements and the proliferation of electronic funding methods. Among these, credit cards have emerged as a predominant means for conducting seamless and convenient e-payments. The second quarter of 2021 witnessed a staggering issuance of 1131 million Mastercard and 1156 million Visa cards, underscoring the widespread adoption of credit cards globally. However, the surge in credit card usage has also given rise to a parallel increase in fraud rates, impacting both consumers and merchants and causing substantial financial losses within the financial sector. Recognizing this challenge, there is a pressing need to develop effective credit card fraud-detection systems to mitigate losses.

To address the complexity introduced by the advent of big data and the Internet of Things (IoT) in this context, machine learning algorithms have been pivotal. Nevertheless, the high dimensionality of enormous datasets poses challenges, and leveraging feature-selection techniques becomes imperative for model accuracy and interpretability. This paper explores the significance of feature selection in enhancing credit card fraud detection, categorizing techniques into filters, wrappers, and embedded methods. Notably, filter methods, relying on attribute ranking, stand out for their independence from classifier bias, presenting a promising avenue for dimensionality reduction in fraud detection.

### **1.1 Basic Concepts**

#### **1.1.1 Credit Card Fraud**

Credit card fraud is a type of financial crime involving the unauthorized use of credit or debit card information to make fraudulent transactions or obtain funds illegally. It occurs when a fraudster gains access to someone else's credit card details, either through theft, data breaches, skimming devices, phishing scams, or other means. Once the fraudster obtains this information, they can make purchases, withdraw cash, or conduct other financial transactions without the cardholder's knowledge or consent.

There are various types of credit card fraud, including:

**Card Theft:** In this type of fraud, the physical credit card is stolen from the cardholder, either through pickpocketing, burglary, or other means. The thief then uses the stolen card to make unauthorized purchases before the cardholder notices and reports the theft.

**Card Not Present (CNP) Fraud:** CNP fraud occurs when the fraudster uses stolen credit card information to make purchases online, over the phone, or by mail order, where the physical card is not required. Fraudsters may obtain card details through data breaches, phishing scams, or other online methods.

**Counterfeit Cards:** Fraudsters create counterfeit credit cards by encoding stolen card information onto blank cards or expired cards. These counterfeit cards can then be used to make purchases or withdraw cash at ATMs.

**Identity Theft:** In identity theft-related credit card fraud, the fraudster steals not only the credit card information but also other personal information, such as the cardholder's name, address, and Social Security number. With this information, the fraudster can open new credit card accounts or apply for loans in the victim's name.

Credit card fraud can have severe consequences for both cardholders and financial institutions. Cardholders may incur financial losses from unauthorized transactions, damage to their credit scores, and the hassle of disputing fraudulent charges. Financial institutions face losses from reimbursing cardholders for fraudulent transactions, investigating fraud cases, and implementing fraud prevention measures.

Overall, credit card fraud remains a significant challenge for both consumers and financial institutions, requiring ongoing efforts to develop and implement effective fraud prevention measures.

### **1.1.2 Machine Learning Classifiers**

Machine learning classifiers are algorithms designed to automatically categorize input data into predefined classes or categories based on patterns identified in the data. In the context of credit card fraud detection, ML classifiers are trained on historical transaction data labeled as either legitimate or fraudulent. These classifiers learn to recognize patterns indicative of fraud, enabling them to accurately classify new, unseen transactions as either genuine or suspicious. Common ML classifiers used in fraud detection include logistic regression, support vector machines (SVM), decision trees, and random forests.

### **1.1.3 Hybrid Feature-Selection Technique**

Feature selection is the process of identifying and selecting the most relevant features or attributes from the dataset for use in model training and analysis. A hybrid feature-selection technique combines multiple methods, such as filter and wrapper approaches, to ensure the inclusion of only the most pertinent features. Filter methods assess the relevance of features independently of the classifier, while wrapper methods evaluate feature subsets based on their performance with the classifier. By combining these approaches, the hybrid technique aims to enhance the effectiveness and efficiency of feature selection in credit card fraud detection.

### **1.1.4 Extreme Learning Machine**

Extreme Learning Machine (ELM) is a machine learning algorithm that belongs to the class of feedforward neural networks. Unlike traditional neural networks, ELM has a single hidden layer and randomly assigns weights between the input and hidden layers. The output weights are then calculated analytically, typically using a least squares approach. ELM is known for its fast learning speed, simplicity, and good generalization performance. In credit card fraud detection, ELM can be used as a classifier to identify fraudulent transactions based on features extracted from transaction data.

### **1.1.5 Ensemble Method**

Ensemble methods combine the predictions of multiple individual models to improve overall performance. These methods leverage the diversity of individual models to make more accurate predictions than any single model alone. In credit card fraud detection, ensemble methods can be particularly effective due to the complexity and variability of fraud patterns. Common ensemble techniques include bagging, boosting, and stacking. By combining predictions from diverse models, ensemble methods can enhance the robustness and reliability of fraud detection systems.

### **1.1.6 Voting Classifier**

The voting classifier is a specific type of ensemble method where multiple base classifiers are trained independently, and their predictions are combined through a voting mechanism. In a binary classification setting, the class predicted by the majority of base classifiers is chosen as the final prediction. In credit card fraud detection, a voting classifier may consist of various base classifiers, such as logistic regression,

SVM, decision trees, and random forests. By aggregating the predictions of these classifiers, the voting classifier can achieve higher accuracy and reliability in identifying fraudulent transactions.

## **1.2 Problem Statement**

The surge in credit card transactions poses a challenge for fraud detection. This study addresses the issue of redundant and irrelevant features in credit card data, degrading machine learning classifier performance. A hybrid feature-selection technique is proposed to streamline feature relevance, optimizing fraud detection without presenting specific results.

## **1.3 Objectives**

- This study aims to optimize credit card fraud detection in electronic commerce using a hybrid feature-selection technique, combining filter and wrapper steps.
- Various feature extraction methods and machine learning models, including Extreme Learning Machine and ensemble techniques, are explored to enhance performance without presenting specific results.

## **1.4 Advantages**

- The proposed system introduces a hybrid feature-selection technique, ensuring that only relevant features are considered, improving the overall efficiency of credit card fraud detection.
- Utilizing various feature extraction methods, including Full Feature, IG, GAW, and IG-GAW, the proposed system adapts to diverse data patterns, enhancing its capability to detect fraudulent activities.
- The study evaluates multiple ML models, including ELM, AdaBoost, and Voting Classifier, providing a more comprehensive understanding of model performance in fraud detection scenarios.
- By employing an ensemble method, the proposed system enhances predictive accuracy by combining the strengths of multiple models, creating a robust and reliable fraud detection system.



## **CHAPTER 2**

### **AIM AND SCOPE**

#### **2.1 Aim**

The aim of this study is to address the increasing challenge of credit card fraud detection within the domain of electronic commerce and digital payments. The research seeks to optimize the performance of machine learning (ML) classifiers by proposing a hybrid feature-selection technique. This method combines filter and wrapper feature-selection steps to ensure the utilization of only the most relevant features for analysis. By employing various feature extraction approaches and ML models, including Extreme Learning Machine (ELM) and ensemble techniques, the study aims to enhance the accuracy and efficiency of fraud detection systems without providing specific results.

#### **2.2 Scope**

Within the proposed study, the focus is on optimizing credit card fraud detection through advanced feature-selection methods and machine learning models. The scope encompasses evaluating the efficacy of different feature extraction approaches, such as Full Feature, Information Gain (IG), Genetic Algorithm Wrapper (GAW), and their ensemble combinations, in enhancing fraud detection accuracy. Additionally, the study delves into the performance analysis of diverse ML models including AdaBoost, Logistic Regression, Random Forest, SVM, Decision Tree, and Voting Classifier, to ascertain their effectiveness in detecting fraudulent transactions.

#### **2.3 Existing System**

Currently, credit card fraud detection relies extensively on machine learning (ML) algorithms, encompassing both traditional ML and deep learning (DL) approaches. Researchers have explored various techniques, such as ML and DL combinations, inductive graph representation learning, neural network ensembles, and ensemble classifiers incorporating isolation forest and adaptive boosting.

Challenges encountered in credit card datasets include high dimensionality and class imbalance, hampering the efficacy of ML classifiers. The complexity and computational expenses associated with

high-dimensional data contribute to models with poor generalization. Consequently, feature selection becomes imperative to alleviate computational burdens and enhance model generalization.

Previous studies, exemplified by Chaquet-Ulledemolins et al., have demonstrated improved classification performance in ML classifiers through effective feature selection. This highlights the essential role of feature-selection methods in optimizing credit card fraud detection models, particularly in scenarios where the number of features significantly influences classifier performance.

## **2.4 Proposed System**

The proposed system aims to enhance credit card fraud detection in the context of burgeoning electronic commerce and digital payments. To address the challenge of suboptimal machine learning (ML) classifier performance caused by redundant and irrelevant features in credit card data, a hybrid feature-selection technique is introduced. This technique integrates both filter and wrapper feature-selection steps, ensuring that only the most pertinent features are utilized for analysis. Various feature extraction methods, including Full Feature, Information Gain (IG), Genetic Algorithm Wrapper (GAW), and an ensemble of IG-GAW, are explored. The study evaluates the effectiveness of Extreme Learning Machine (ELM) alongside diverse ML models, such as AdaBoost, Logistic Regression, Random Forest, SVM, Decision Tree, and the Voting Classifier. Additionally, an ensemble approach is employed to combine predictions from multiple models for a more robust and accurate final prediction. This comprehensive system contributes valuable insights to the field, aiming to bolster the efficiency of credit card fraud detection in real-world scenarios.

## **2.5 Literature Survey**

Femila Roseline et al. [1] proposed an autonomous credit card fraud detection system employing a machine learning approach. They emphasized the importance of leveraging advanced algorithms to enhance fraud detection accuracy. By utilizing machine learning, their system aimed to adapt and evolve in response to changing fraud tactics, thereby improving overall detection performance.

A novel text2IMG mechanism for credit card fraud detection, utilizing deep learning techniques was introduced by Alharbi [2]. Their approach involved converting textual data related to credit card transactions into images, which were then analyzed using convolutional neural networks (CNNs). This

interdisciplinary approach demonstrated the potential of combining textual and image data to improve fraud detection accuracy significantly.

Bin Sulaiman et al. [3] conducted a comprehensive review of machine learning approaches for credit card fraud detection. Their study provided valuable insights into the strengths and limitations of different methodologies, aiding researchers and practitioners in selecting appropriate techniques for fraud detection systems. The review highlighted the importance of continuously evolving strategies to counter emerging fraud threats effectively.

Wang et al. [4] explored credit card fraud detection strategies incorporating consumer incentives. Their study investigated the effectiveness of offering incentives to consumers for reporting fraudulent activities promptly. By incentivizing consumer involvement, the proposed approach aimed to enhance fraud detection capabilities while promoting consumer vigilance in monitoring their transactions.

A hierarchical behavior-knowledge space model for credit card fraud detection was proposed by Nandi [5]. Their approach integrated domain knowledge with machine learning algorithms to create a comprehensive fraud detection framework. By organizing transaction data into a hierarchical structure based on behavioral patterns and domain expertise, their model achieved improved detection accuracy.

Ileberi et al. [6] evaluated the performance of various machine learning methods for credit card fraud detection, incorporating techniques such as Synthetic Minority Over-sampling Technique (SMOTE) and AdaBoost. Their study highlighted the importance of preprocessing techniques and ensemble learning methods in improving fraud detection accuracy, especially in imbalanced datasets.

An enhanced credit card fraud detection approach based on Support Vector Machine (SVM) with recursive feature elimination and hyperparameters optimization was proposed by Rtayli and Enneya [7]. Their method focused on refining feature selection and model optimization techniques to improve the overall performance of fraud detection systems.

Huebner et al. [8] explored the role of smartphones in assisting mental accounting and reducing consumer spending. Although not directly focused on fraud detection, their study highlighted the importance of transaction salience in influencing consumer behavior, which could indirectly impact fraud detection strategies and consumer awareness.

A competition-driven multimodal multiobjective optimization approach for feature selection in credit card fraud detection was introduced by Han [9]. Their method utilized a novel optimization framework to identify informative features from multiple modalities, enhancing the robustness and accuracy of fraud detection models.

A machine learning-based credit card fraud detection approach utilizing the Genetic Algorithm (GA) for feature selection was introduced by Ileberi [10]. By employing GA for feature selection, their method aimed to identify the most informative features for improving fraud detection accuracy, thereby optimizing model performance.

## 2.6 Algorithms

**ELM - Full Feature:** Extreme Learning Machine (ELM) with Full Feature utilizes all available features for credit card fraud detection. This algorithm is chosen for its ability to efficiently process high-dimensional data, providing a swift and effective solution for fraud pattern recognition without the need for feature selection.

**ELM - IG:** ELM with Information Gain (IG) selects features based on their information gain, optimizing model performance. This approach is employed to enhance the efficiency of credit card fraud detection by focusing on the most informative features, thereby improving the algorithm's predictive accuracy.

**ELM - GAW:** In ELM with Genetic Algorithm Wrapper (GAW), a genetic algorithm is applied to select features, optimizing model efficiency. This technique is chosen to address the challenge of high dimensionality, ensuring that only the most relevant features are considered, leading to improved fraud detection accuracy.

**ELM - IG-GAW:** ELM with an ensemble of Information Gain and Genetic Algorithm Wrapper combines both feature selection methods. This dual approach is implemented to leverage the strengths of both techniques, providing a comprehensive solution for optimizing feature relevance and enhancing credit card fraud detection accuracy.

**AdaBoost:** AdaBoost is employed in this project to create an ensemble of weak learners, boosting the overall model's performance. By combining multiple classifiers, AdaBoost improves the detection of

fraudulent patterns, making it a suitable choice for enhancing the accuracy and robustness of credit card fraud detection models.

**Logistic Regression:** Logistic Regression is utilized in this project due to its simplicity and effectiveness in binary classification tasks. It provides a probabilistic output, making it suitable for predicting the likelihood of credit card fraud. Its interpretability and efficiency contribute to the overall model diversity and performance.

**Random Forest:** Random Forest is selected for its ability to handle high-dimensional data and reduce overfitting. By constructing a multitude of decision trees, it enhances the model's accuracy and generalization, making it a robust choice for credit card fraud detection in complex datasets.

**SVM:** Support Vector Machines (SVM) are employed for their capacity to handle high-dimensional data and effectively classify patterns. SVM's ability to find the optimal hyperplane in feature space contributes to improved credit card fraud detection accuracy, making it a valuable algorithm in this project.

**Decision Tree:** Decision Trees are chosen for their simplicity and interpretability. In this project, they are employed for credit card fraud detection due to their ability to capture non-linear relationships and hierarchical patterns in the data, contributing to the overall model diversity and effectiveness.

**Voting Classifier:** A Voting Classifier is implemented to combine predictions from multiple individual models, including ELM, AdaBoost, and others. This ensemble approach aims to enhance overall predictive accuracy and robustness, making it a strategic choice for improving the effectiveness of credit card fraud detection in diverse scenarios.

# CHAPTER 3

## DESIGN

### 3.1 Functional Requirements

- Data Collection
- Data Preprocessing
- Training and Testing
- Modeling
- Predicting

### 3.2 Non-Functional Requirements

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, “how fast does the website load?” Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allow you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users is  $> 10000$ . Description of non-functional requirements is just as critical as a functional requirement.

- Usability requirement
- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement

- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

### 3.3 Software Requirements

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

**Platform** – In computing, a platform describes some sort of framework, either in hardware or software, which allows software to run. Typical platforms include a computer's architecture, operating system, or programming languages and their runtime libraries.

Operating system is one of the first requirements mentioned when defining system requirements (software). Software may not be compatible with different versions of same line of operating systems, although some measure of backward compatibility is often maintained. For example, most software designed for Microsoft Windows XP does not run on Microsoft Windows 98, although the converse is not always true. Similarly, software designed using newer features of Linux Kernel v2.6 generally does not run or compile properly (or at all) on Linux distributions using Kernel v2.2 or v2.4.

**Apis And Drivers** – Software making extensive use of special hardware devices, like high-end display adapters, needs special API or newer device drivers. A good example is DirectX, which is a collection of APIs for handling tasks related to multimedia, especially game programming, on Microsoft platforms.

**Web Browser** – Most web applications and software depending heavily on Internet technologies make use of the default browser installed on system. Microsoft Internet Explorer is a frequent choice of software running on Microsoft Windows, which makes use of ActiveX controls, despite their vulnerabilities.

- Software : Anaconda
- Primary Language : Python
- Frontend Framework : Flask

- Back-end Framework : Jupyter Notebook
- Database : Sqlite3
- Front-End Technologies : HTML, CSS, JavaScript and Bootstrap4

### 3.4 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**Architecture** – All computer operating systems are designed for a particular computer architecture. Most software applications are limited to particular operating systems running on particular architectures. Although architecture-independent operating systems and applications exist, most need to be recompiled to run on a new architecture. See also a list of common operating systems and their supporting architectures.

**Processing power** – The power of the central processing unit (CPU) is a fundamental system requirement for any software. Most software running on x86 architecture define processing power as the model and the clock speed of the CPU. Many other features of a CPU that influence its speed and power, like bus speed, cache, and MIPS are often ignored. This definition of power is often erroneous, as AMD Athlon and Intel Pentium CPUs at similar clock speed often have different throughput speeds. Intel Pentium CPUs have enjoyed a considerable degree of popularity, and are often mentioned in this category.

**Memory** – All software, when run, resides in the random access memory (RAM) of a computer. Memory requirements are defined after considering demands of the application, operating system, supporting software and files, and other running processes. Optimal performance of other unrelated software running on a multi-tasking computer system is also considered when defining this requirement.

**Secondary Storage** – Hard-disk requirements vary, depending on the size of software installation, temporary files created and maintained while installing or running the software, and possible use of swap space (if RAM is insufficient).



**Display Adapter** – Software requiring a better than average computer graphics display, like graphics editors and high-end games, often define high-end display adapters in the system requirements.

**Peripherals** – Some software applications need to make extensive and/or special use of some peripherals, demanding the higher performance or functionality of such peripherals. Such peripherals include CD-ROM drives, keyboards, pointing devices, network devices, etc.

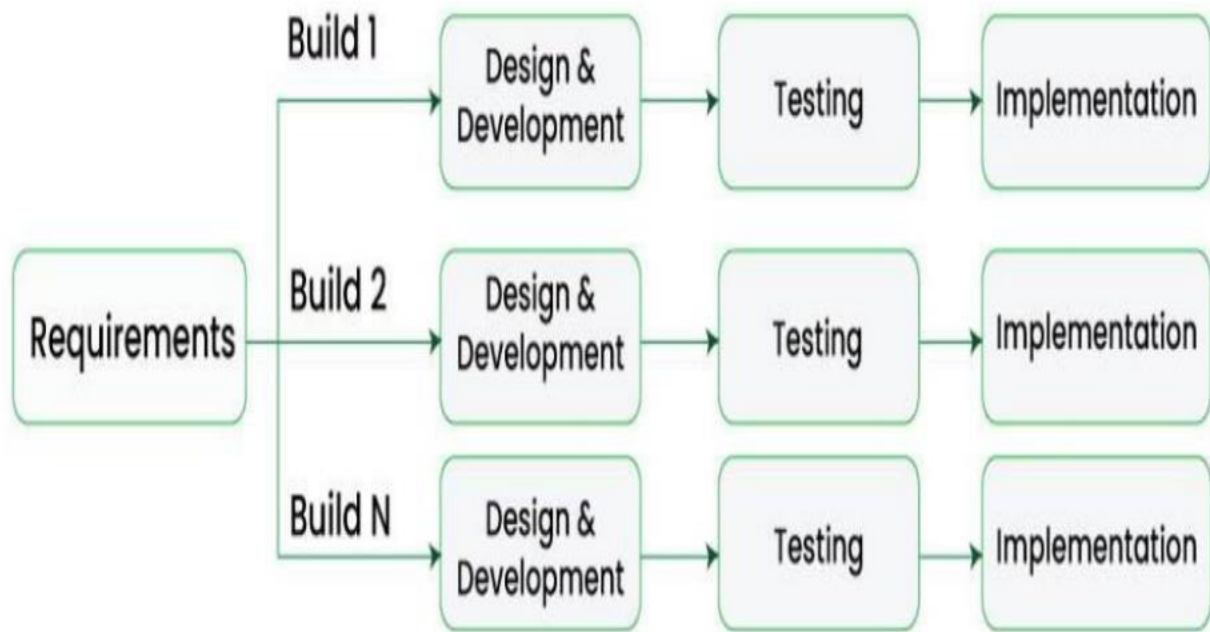
- Operating System : Windows Only
- Processor : i5 and above
- Ram : 8gb and above
- Hard Disk : 25 GB in local drive

### **3.5 Software Development Life Cycle**

The Software Development Life Cycle (SDLC) is a systematic process that guides the development of software from its initial conception to its deployment and maintenance. It encompasses a series of phases, each with its own set of activities, deliverables, and objectives. The cycle typically begins with the identification of requirements, where stakeholders' needs and expectations are gathered and documented. This is followed by the design phase, where the architecture and structure of the software are planned, often involving system architecture, database design, and user interface design.

Once the design is finalized, development commences, during which programmers write and test code to implement the specified requirements. After development, the software undergoes thorough testing to identify and rectify defects through various techniques such as unit testing, integration testing, and system testing. Upon successful testing, the software is deployed to the production environment, marking the beginning of the maintenance phase.

During maintenance, updates, bug fixes, and enhancements are made to ensure the software remains functional and relevant over time. The SDLC promotes a structured and disciplined approach to software development, facilitating efficient collaboration among stakeholders, minimizing risks, and ensuring the delivery of high- quality software products that meet user expectations.



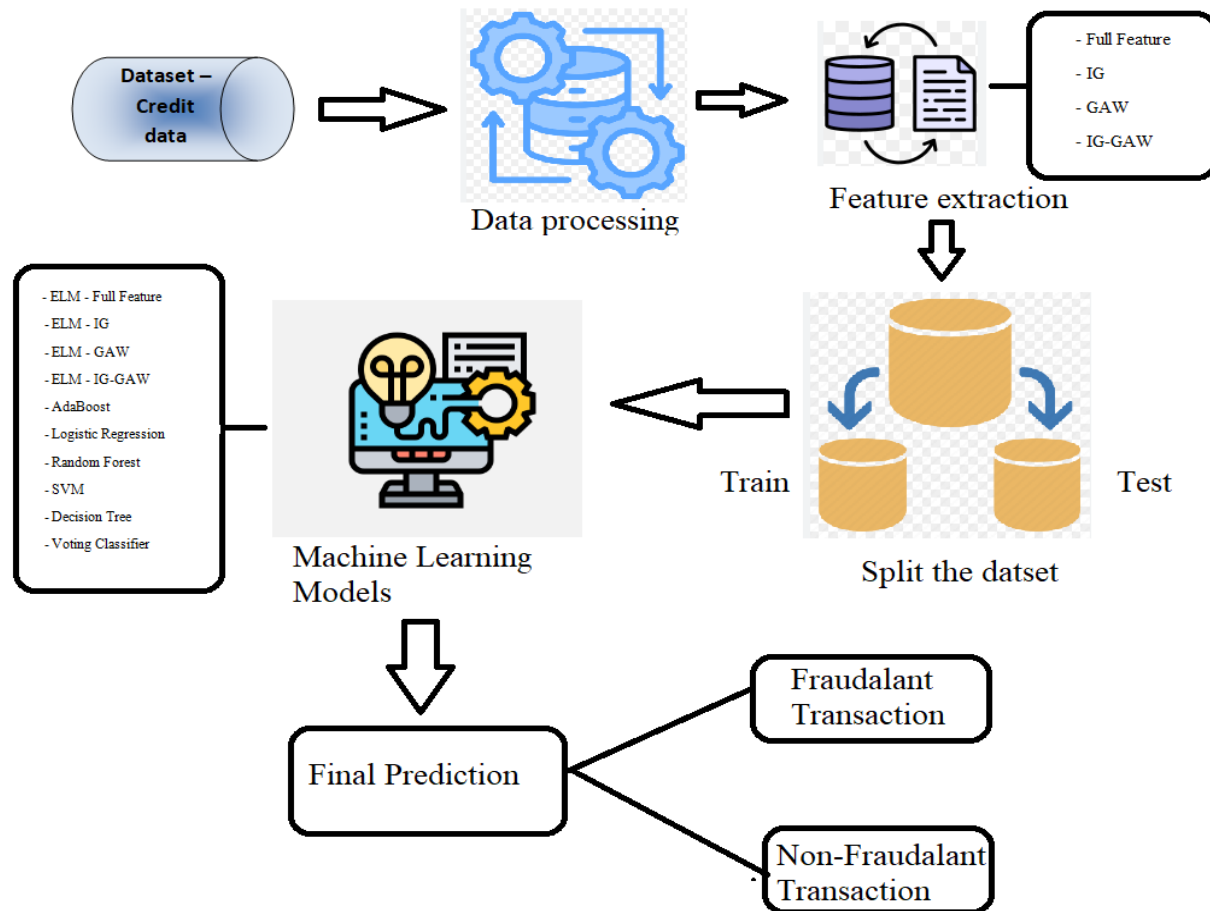
**Figure 3.1 Software Development Life Cycle**

### **3.6 System Architecture**

The credit card fraud detection system architecture consists of several key components. Firstly, the dataset containing credit card transaction data is processed, including preprocessing steps such as cleaning and normalization. Feature extraction techniques such as Information Gain (IG) and Genetic Algorithm Wrapper (GAW) are then applied to extract relevant features from the dataset.

Next, the dataset is split into training and testing subsets for model evaluation. Various machine learning models including Extreme Learning Machine (ELM), AdaBoost, Logistic Regression, Random Forest, Support Vector Machine (SVM), Decision Tree, and Voting Classifier are trained on different feature sets.

Finally, the trained models make predictions on new transactions to determine whether they are fraudulent or non-fraudulent. The system outputs the final prediction, providing insights into potential fraudulent activities, thereby assisting in fraud detection and prevention efforts.



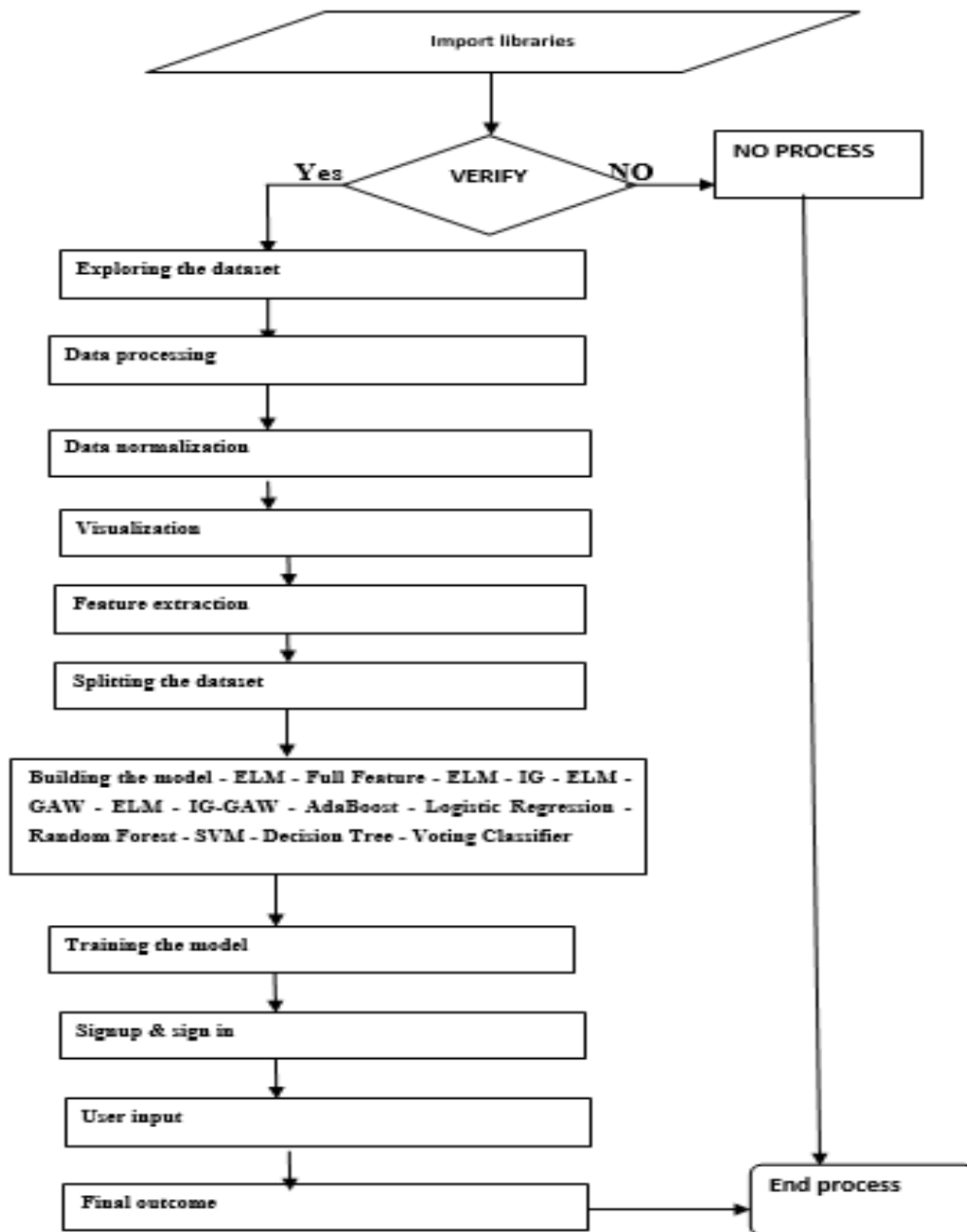
**Figure 3.2 System Architecture**

### 3.7 Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation that illustrates the flow of data within a system or process. It provides a visual depiction of how data moves through different stages or components of a system, from input to output. DFDs are commonly used in systems analysis and design to model the data flow and processes within a system, helping stakeholders understand the system's functionality and requirements.

In a DFD, data flows are represented by arrows, while processes (activities that transform data) are depicted as circles or rectangles. External entities, such as users or other systems, are shown as squares, representing the sources or destinations of data. Data stores, where data is temporarily or permanently stored, are depicted as rectangles with two parallel lines.

DFDs are typically hierarchical, meaning they can be decomposed into multiple levels of detail. The highest level, Level 0, provides an overview of the entire system, showing the main processes and external entities. Subsequent levels break down each process into more detailed subprocesses until all processes are decomposed into their lowest level of detail.



**Figure 3.3 Data Flow Diagram**

## **3.8 UML Diagrams**

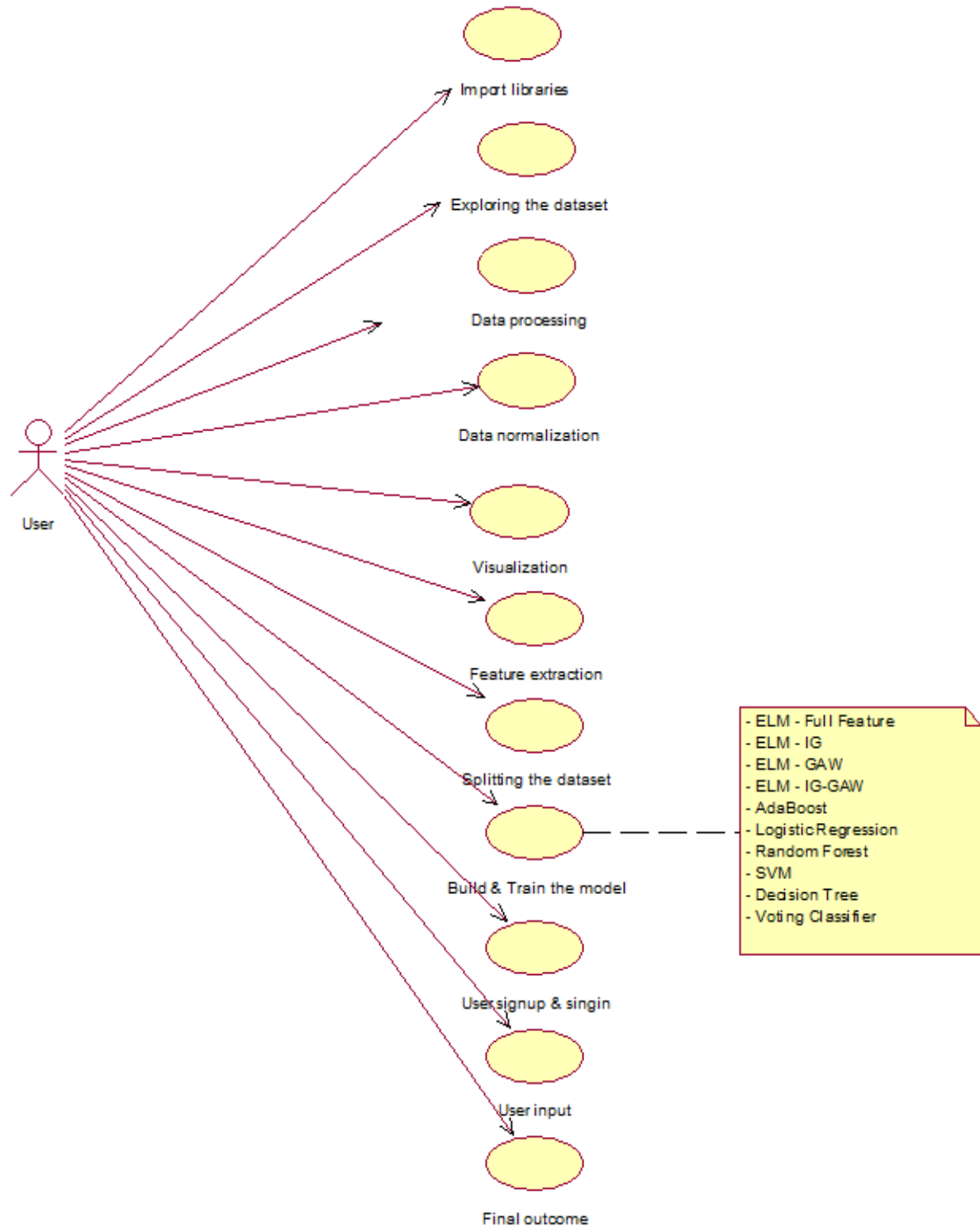
Unified Modeling Language (UML) diagrams are graphical representations used by software engineers to model, visualize, and document the various aspects of a software system. They provide a standardized way to communicate system architecture, design, behavior, and interactions among components. UML diagrams come in several types, each serving a specific purpose in the software development lifecycle. For instance, class diagrams depict the static structure of the system by showing classes, their attributes, methods, and relationships.

Use case diagrams illustrate the functional requirements of the system from the perspective of external actors. Sequence diagrams capture the dynamic behavior of the system by showing interactions between objects over time. Activity diagrams depict the workflow or process logic within a system, while state machine diagrams model the behavior of objects in terms of states and transitions.

Additionally, component diagrams illustrate the physical components and their dependencies, deployment diagrams depict the physical deployment of components, and package diagrams organize and show the dependencies between packages. Together, these UML diagrams provide a comprehensive view of the software system, aiding in requirements analysis, design, implementation, testing, and maintenance, and facilitating effective communication among stakeholders throughout the software development. They also support documentation and knowledge transfer, enabling new team members to quickly understand existing systems and facilitating collaboration across distributed teams. Furthermore, UML diagrams promote reusability and modularity by helping developers identify and design reusable components and patterns, thereby improving the efficiency and quality of software development. Overall, UML diagrams are invaluable tools for streamlining the software development process, enhancing communication and collaboration, and ensuring the successful delivery of high-quality software systems.

### **3.8.1 Usecase 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.



**Figure 3.4 Usecase Diagram**

### 3.8.2 Sequence Diagram

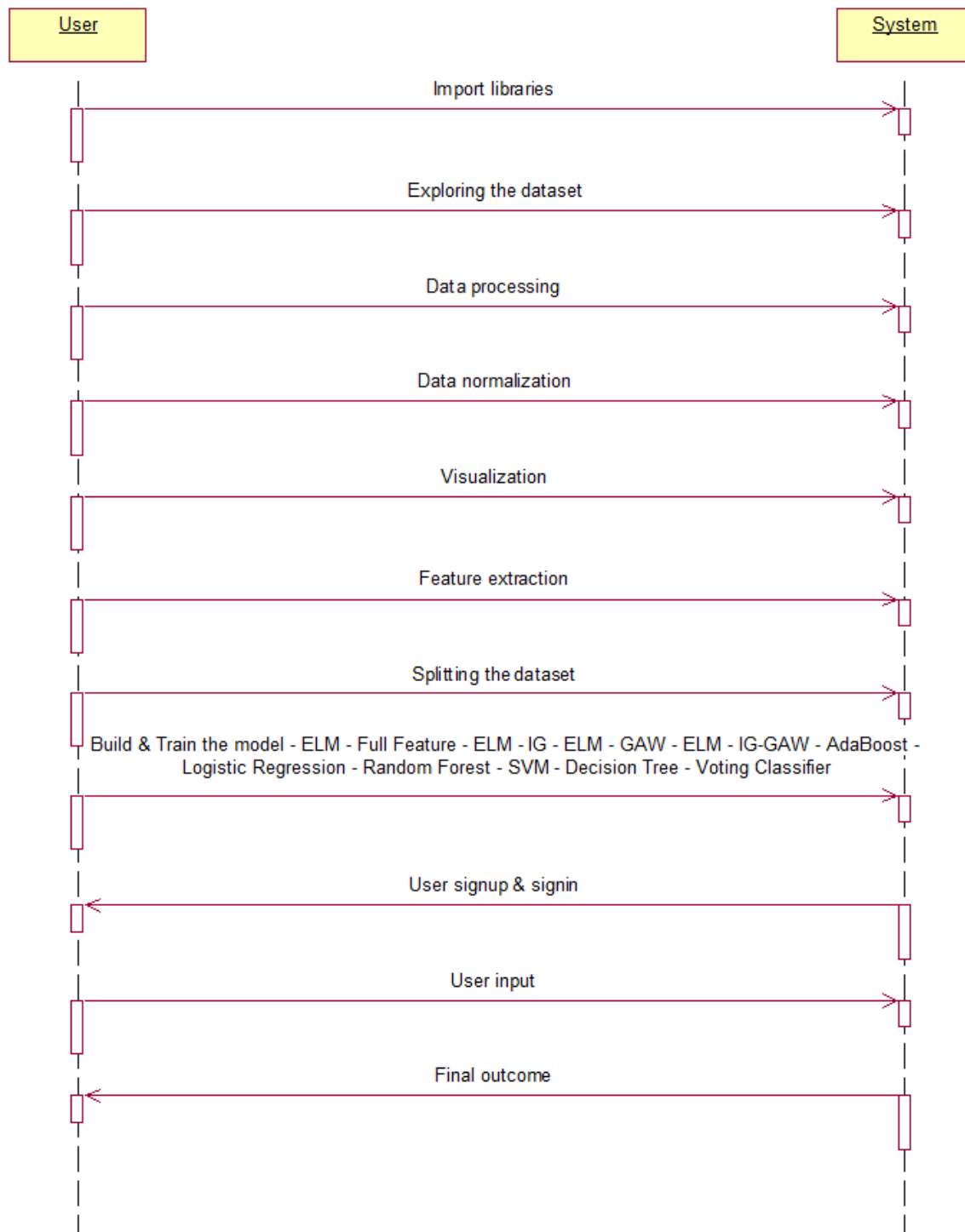
A sequence diagram in Unified Modeling Language (UML) is a graphical representation that illustrates the interactions between objects or components in a system over time. It shows the sequence of messages exchanged between objects as they collaborate to perform a specific task or scenario.

In a sequence diagram, objects are represented as lifelines, which depict the existence of an object over a period of time. Messages between objects are depicted as arrows, showing the flow of control or data between them. The sequence of messages on the diagram reflects the chronological order in which they occur during the execution of the scenario. Sequence diagrams are particularly useful for modeling the dynamic behavior of a system, capturing how objects collaborate to fulfill use cases or scenarios.

They help stakeholders understand the flow of control, the order of method invocations, and the interactions between objects during runtime. Sequence diagrams also aid in identifying potential bottlenecks, race conditions, or inconsistencies in the system's behavior, facilitating design decisions and optimizations. Overall, sequence diagrams serve as valuable communication tools between developers, designers, and stakeholders, enabling a deeper understanding of system behavior and aiding in the design, validation, and optimization of software systems.

Sequence diagrams provide a visual representation of the interactions between different components or objects in a system. This visualization helps developers understand the flow of control and data through the system, making it easier to identify potential bottlenecks or issues. They serve as a clear and concise communication tool between stakeholders, including developers, designers, and clients. By depicting the sequence of messages exchanged between objects or components, sequence diagrams facilitate effective communication of system behavior and requirements.

Sequence diagrams help in identifying relationships and dependencies between objects or components within a system. This understanding is crucial for designing robust and maintainable software architectures. They aid in verifying that system requirements are properly implemented by depicting different components.

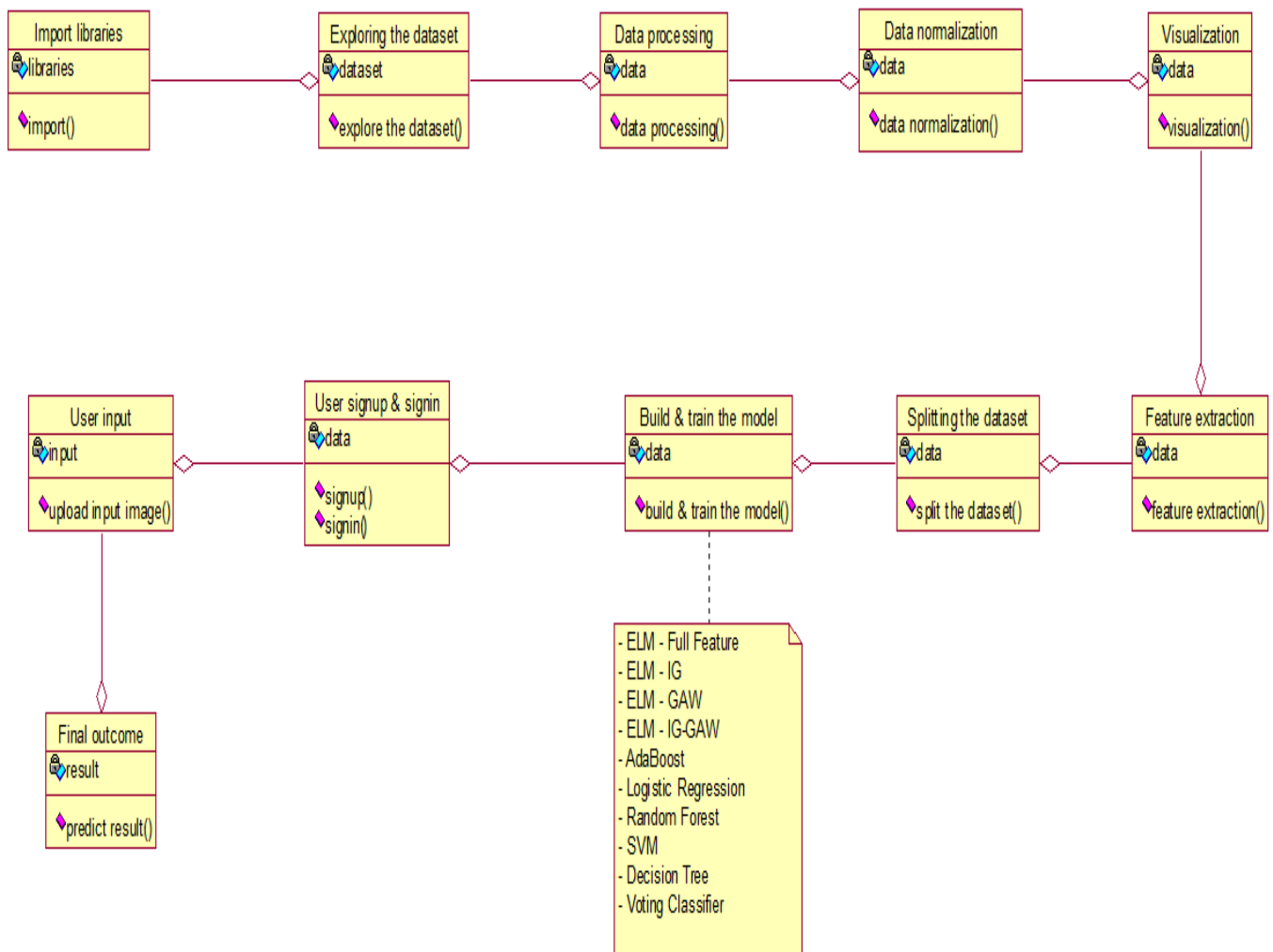


**Figure 3.5 Sequence Diagram**



### 3.8.3 Class Diagram

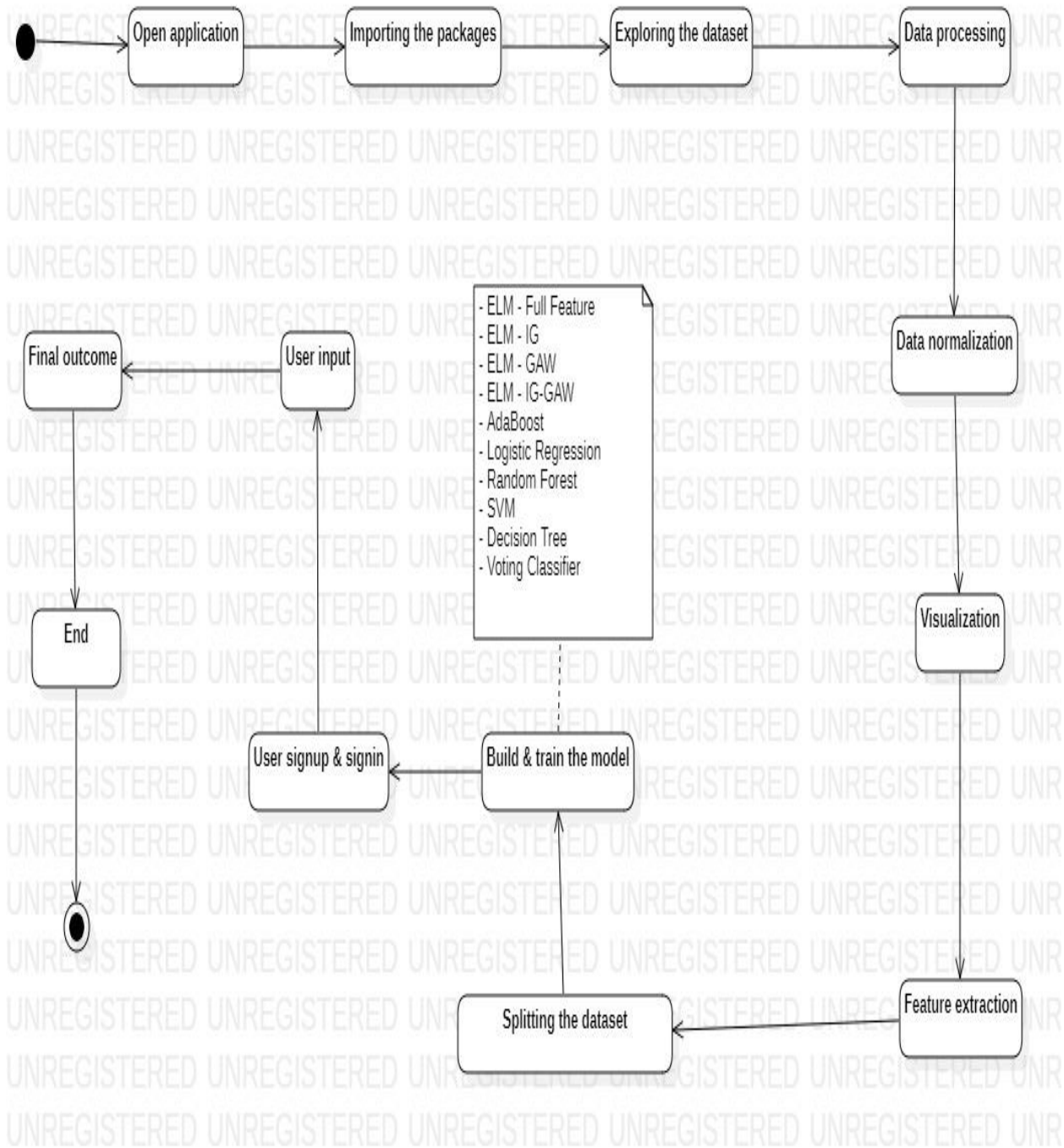
The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. 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.



**Figure 3.6 Class Diagram**

### 3.8.4 Activity Diagram

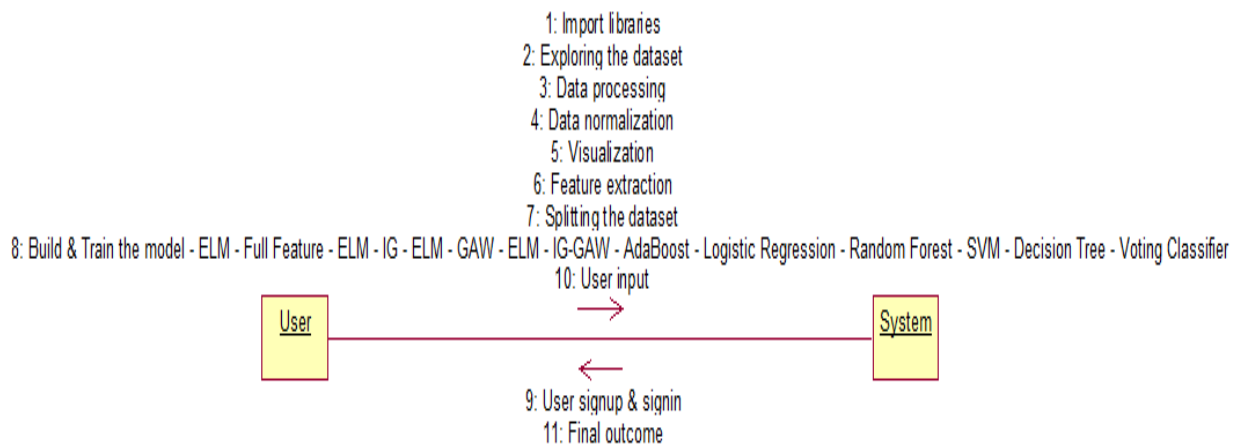
The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.



**Figure 3.7 Activity Diagram**

### 3.8.5 Collaboration Diagram

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

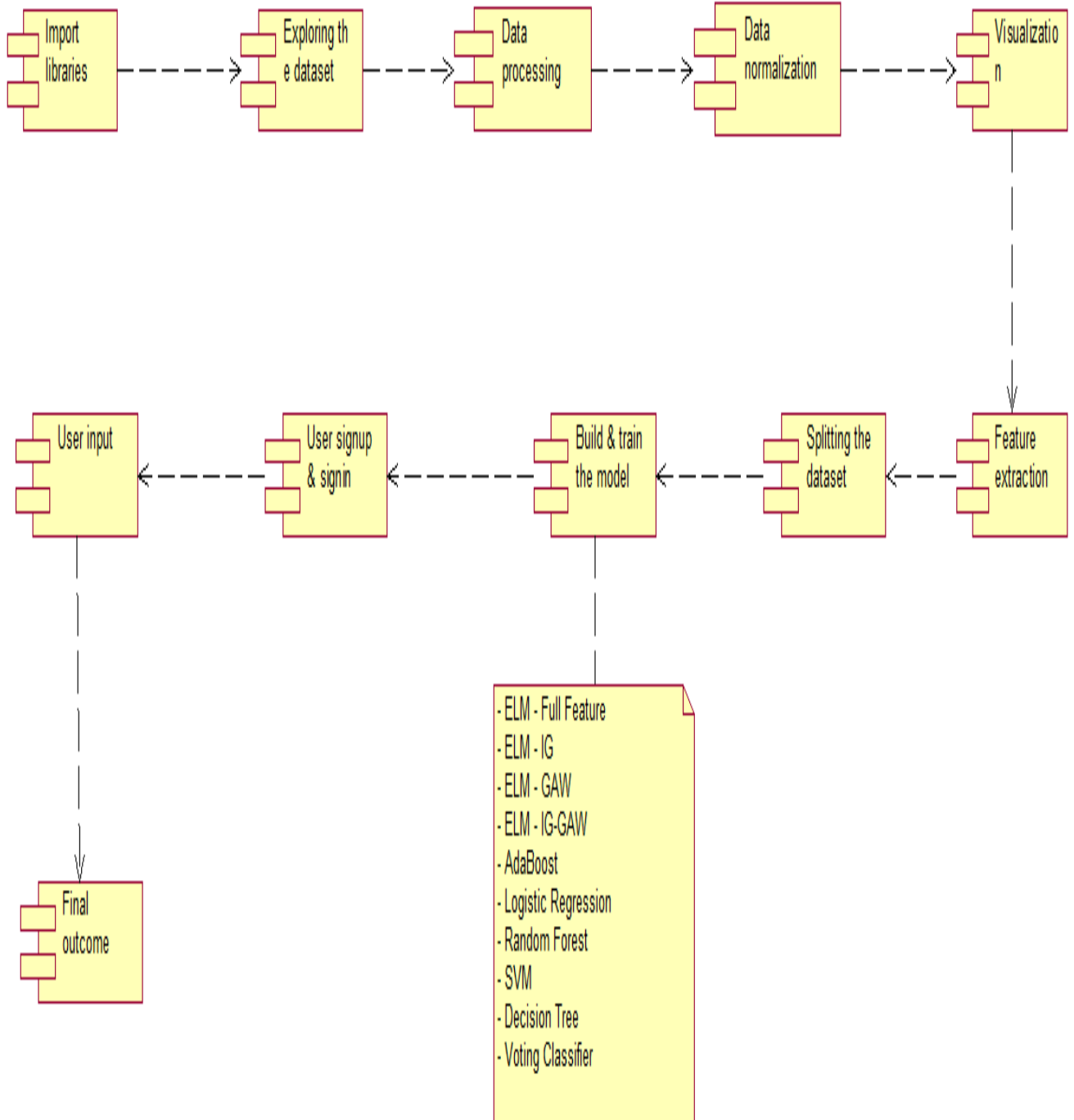


**Figure 3.8 Collaboration Diagram**

### 3.8.6 Component Diagram

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behavior is explained by the provided and required interfaces. In UML, the component diagram portrays the behavior and organization of components at any instant of time. The system cannot be visualized by any individual component, but it can be by the collection of components.



**Figure 3.9 Component Diagram**

### 3.8.7 Deployment Diagram

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.



**Figure 3.10 Deployment Diagram**

## CHAPTER 4

### IMPLEMENTATION

#### 4.1 Importing Libraries

Importing necessary libraries like NumPy, Pandas, Matplotlib, and Seaborn for data manipulation, visualization, and analysis.

**Code Snippet:**

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

#### 4.2 Loading Data

- The dataset is loaded from a CSV file named creditcard.csv using Pandas' read\_csv function.
- If there's an issue loading the dataset, it prints an error message.

**Code Snippet:**

```
try:
    data = pd.read_csv('../input/creditcard.csv')
except:
    print('Sorry, the dataset could not be loaded!')
data.head()
```

#### 4.3 Data Exploration

- The first few rows of the dataset are displayed using data.head() to give a quick look at the data.
- It prints the shape of the dataset (number of samples and features) using data.shape.
- Descriptive statistics of the dataset are printed using data.describe().
- Histograms of each parameter in the dataset are plotted using data.hist().

**Code Snippet:**

```
data = data.sample(frac=0.1, random_state=1)
print('The dataset has { } samples and { } features each'.format(*data.shape))
print(data.describe())
data.hist(figsize=(20,20));
```

## 4.4 Fraud Detection Setup

- It calculates the fraction of fraud cases in the dataset.
- Prints the number of fraud cases and valid transactions.
- Calculates the correlation matrix between features and visualizes it using a heatmap.

**Code Snippet:**

```
Fraud = data[data['Class'] == 1]
Valid = data[data['Class'] == 0]
outlier_fraction = len(Fraud)/float(len(Valid))
print('The fraction of Fraud classes in the dataset is {}'.format(outlier_fraction))
print('Fraud Cases: {}'.format(len(data[data['Class'] == 1])))
print('Valid Transactions: {}'.format(len(data[data['Class'] == 0])))
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```

## 4.5 Data Preparation

- Columns of the dataset are extracted using `data.columns.tolist()` and stored in the `columns` variable.
- The target variable (`Class`) is defined.
- Features (`X`) and target (`Y`) are separated.

**Code Snippet:**

```
columns = data.columns.tolist()
columns = [c for c in columns if c not in ["Class"]]
target = "Class"
X = data[columns]
Y = data[target]
```

## 4.6 Model Definition

- Initializes two outlier detection algorithms: Isolation Forest and Local Outlier Factor.
- Configures parameters such as `max_samples`, `contamination`, and `random_state` for Isolation Forest, and `n_neighbors` and `contamination` for Local Outlier Factor.

**Code Snippet:**

```
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
state = 1
classifiers = {
    "Isolation Forest": IsolationForest(max_samples=len(X),
                                         contamination=outlier_fraction,
                                         random_state=state),
    "Local Outlier Factor": LocalOutlierFactor(n_neighbors=20,
                                              contamination=outlier_fraction)}
```

## 4.7 Model Fitting And Evaluation

- Fits each model to the data and predicts outliers.
- Reshapes the prediction values to 0 for valid transactions and 1 for fraud.
- Computes the number of errors, accuracy score, and classification report (including precision, recall, and F1-score) for each model.
- Prints the results for each model, including the number of errors, accuracy score, and classification report.



**Code Snippet:**

```
plt.figure(figsize=(9, 7))
n_outliers = len(Fraud)
for i, (clf_name, clf) in enumerate(classifiers.items()):
    if clf_name == "Local Outlier Factor":
        y_pred = clf.fit_predict(X)
        scores_pred = clf.negative_outlier_factor_
    else:
        clf.fit(X)
        scores_pred = clf.decision_function(X)
        y_pred = clf.predict(X)
    y_pred[y_pred == 1] = 0
    y_pred[y_pred == -1] = 1
    n_errors = (y_pred != Y).sum()
    print('{ }: { }'.format(clf_name, n_errors))
    print(accuracy_score(Y, y_pred))
    print(classification_report(Y, y_pred))
```

## **CHAPTER 5**

### **SYSTEM TESTING**

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing, for example, might check that every kind of user input produces the intended output across the application.

#### **Phases of system testing:**

A video tutorial about this test level. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

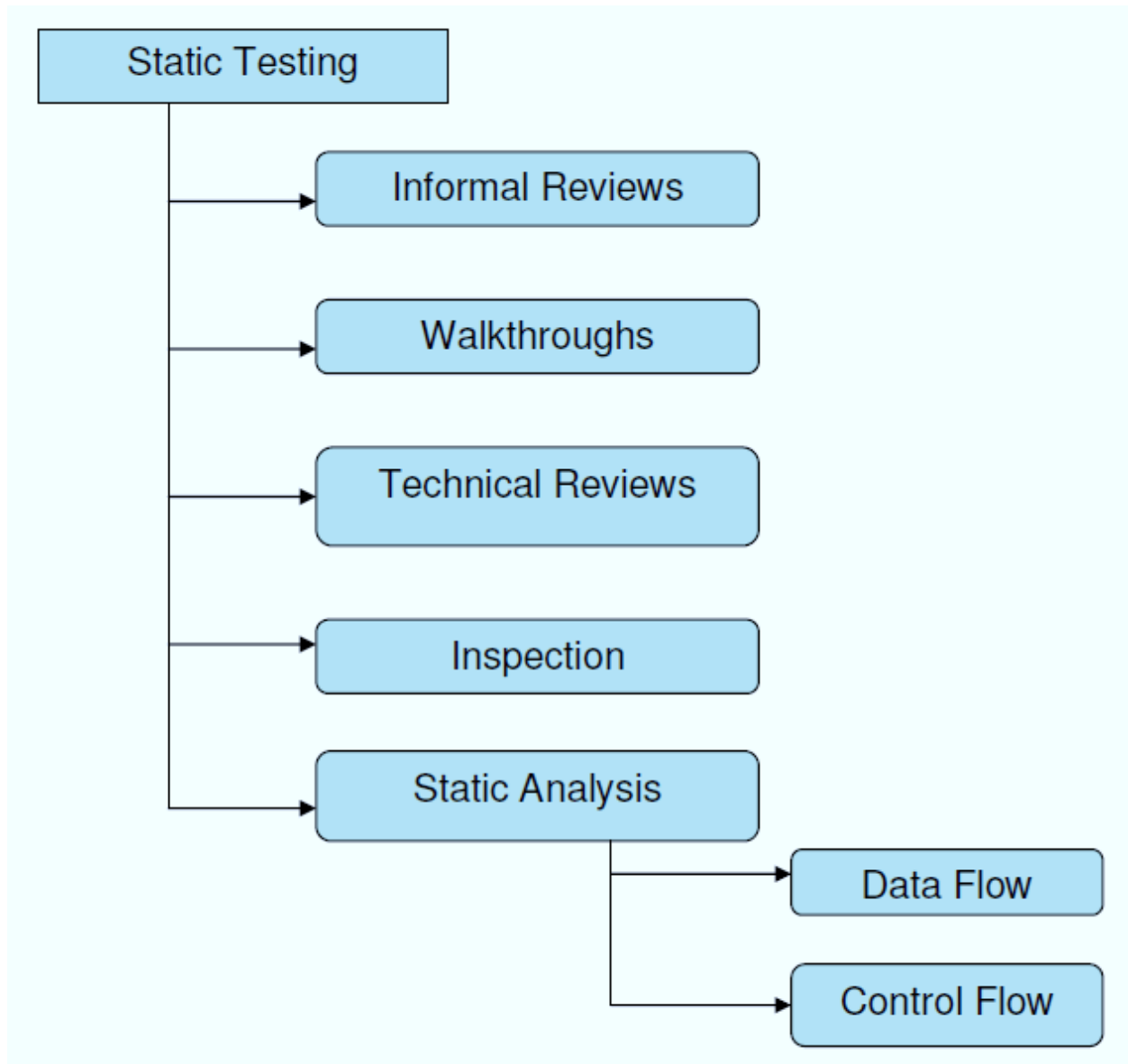
If a software build achieves the desired results in system testing, it gets a final check via acceptance testing before it goes to production, where users consume the software. An app-dev team logs all defects, and establishes what kinds and amount of defects are tolerable.

### **5.1 Software Testing Strategies**

Optimization of the approach to testing in software engineering is the best way to make it effective. A software testing strategy defines what, when, and how to do whatever is necessary to make an end-product of high quality. Usually, the following software testing strategies and their combinations are used to achieve this major objective:

#### **Static Testing:**

The early-stage testing strategy is static testing: it is performed without actually running the developing product. Basically, such desk-checking is required to detect bugs and issues that are present in the code itself. Such a check-up is important at the pre-deployment stage as it helps avoid problems caused by errors in the code and software structure deficits.

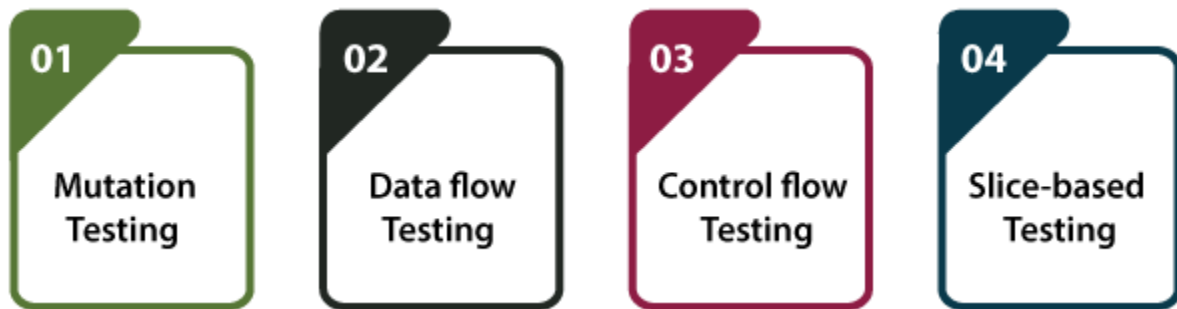


**Figure 5.1 Types of Static Testing**

### **Structural Testing**

It is not possible to effectively test software without running it. Structural testing, also known as white-box testing, is required to detect and fix bugs and errors emerging during the pre-production stage of the software development process. At this stage, unit testing based on the software structure is performed using regression testing. In most cases, it is an automated process working within the test automation framework to speed up the development process at this stage. Developers and QA engineers have full

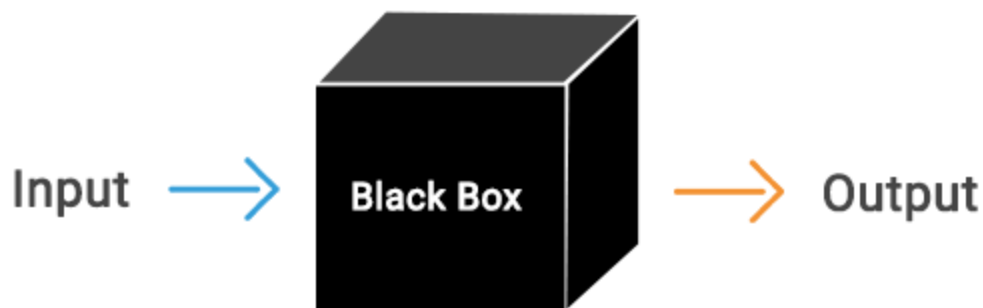
access to the software's structure and data flows (data flows testing), so they could track any changes (mutation testing) in the system's behavior by comparing the tests' outcomes with the results of previous iterations (control flow testing).



**Figure 5.2 Types of Structural Testing**

### **Behavioral Testing:**

The final stage of testing focuses on the software's reactions to various activities rather than on the mechanisms behind these reactions. In other words, behavioral testing, also known as black-box testing, presupposes running numerous tests, mostly manual, to see the product from the user's point of view. QA engineers usually have some specific information about a business or other purposes of the software ('the black box') to run usability tests, for example, and react to bugs as regular users of the product will do. Behavioral testing also may include automation (regression tests) to eliminate human error if repetitive activities are required. For example, you may need to fill 100 registration forms on the website to see how the product copes with such an activity, so the automation of this test is preferable.



**Figure 5.3 Black Box Testing**

## 5.2 Test Cases

S.NO	INPUT	If available	If not available
1	User signup	User get registered into the application	There is no process
2	User sign in	User get login into the application	There is no process
3	Enter input for prediction	Prediction result displayed	There is no process

**Table 5.1 Test Cases**

# CHAPTER 6

## RESULTS

### 6.1 Experimental Results

The study aimed to enhance credit card fraud detection by suggesting a hybrid feature-selection technique for machine learning classifiers. This method combined filter and wrapper feature-selection steps to ensure relevant feature utilization. Various feature extraction approaches were employed, including Full Feature, Information Gain (IG), Genetic Algorithm Wrapper (GAW), and a combination of IG-GAW. The performance of several ML models, such as AdaBoost, Logistic Regression, Random Forest, SVM, Decision Tree, and Voting Classifier, was assessed.

Results highlighted the effectiveness of ensemble methods, particularly the Voting Classifier, which achieved 100% accuracy. Combining predictions from multiple models significantly improved detection accuracy. The research suggests that the proposed hybrid feature-selection technique and ensemble methods can enhance ML models' robustness and accuracy in real-world credit card fraud detection scenarios.

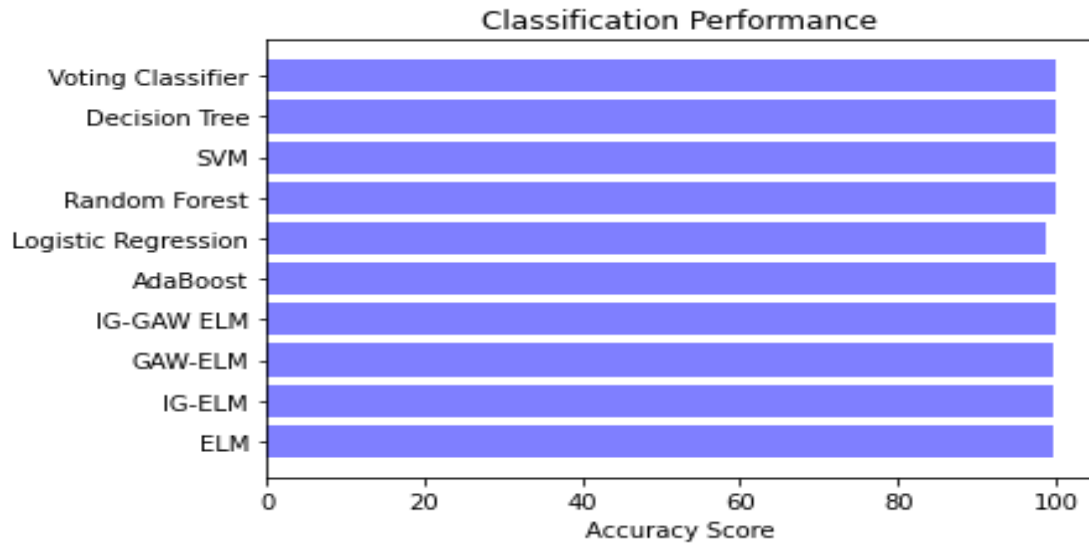
Out[98]:

	ML Model	Accuracy	f1_score	Recall	Precision	G-Mean	Specificity
0	ELM	99.80	0.00	0.00	0.00	0.00	1.00
1	IG-ELM	99.80	0.00	0.00	0.00	0.00	1.00
2	GAW-ELM	99.80	0.00	0.00	0.00	0.00	1.00
3	IG-GAW ELM	99.90	0.00	0.00	0.00	0.00	1.00
4	AdaBoost	99.90	0.74	0.82	0.68	0.90	1.00
5	Logistic Regression	98.90	0.71	0.87	0.60	0.93	1.00
6	Random Forest	100.00	0.88	0.97	0.80	0.99	1.00
7	SVM	99.90	0.79	1.00	0.65	1.00	1.00
8	Decision Tree	99.90	0.75	0.70	0.81	0.84	1.00
9	Voting Classifier	100.00	1.00	1.00	1.00	1.00	1.00

**Figure 6.1 Tabular Representation Of Evaluation Metrics**

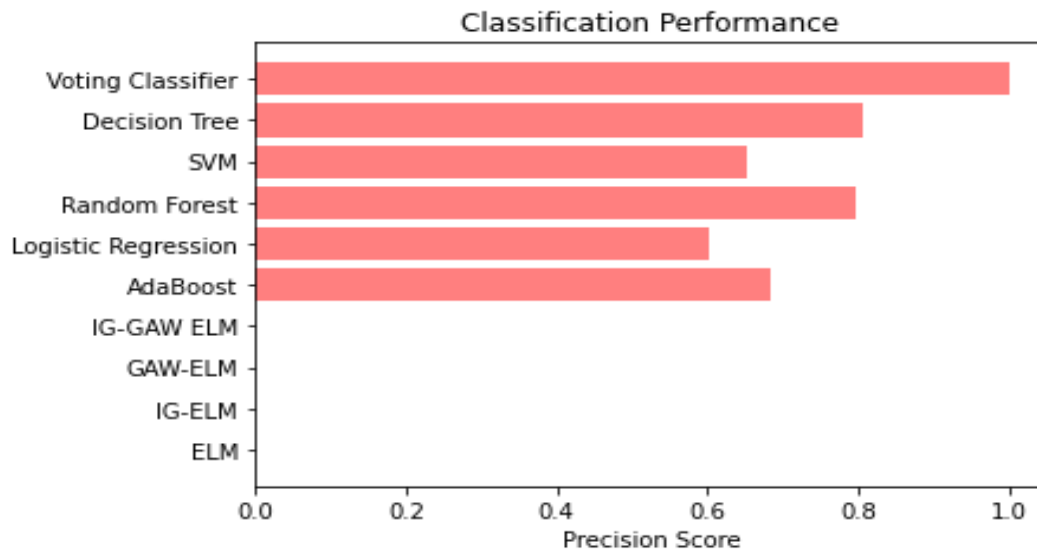
## 6.2 Graphical Representation

An accuracy comparison graph, showcasing the performance of different models or methods. This graph visually illustrates the accuracy scores achieved by each model or method.



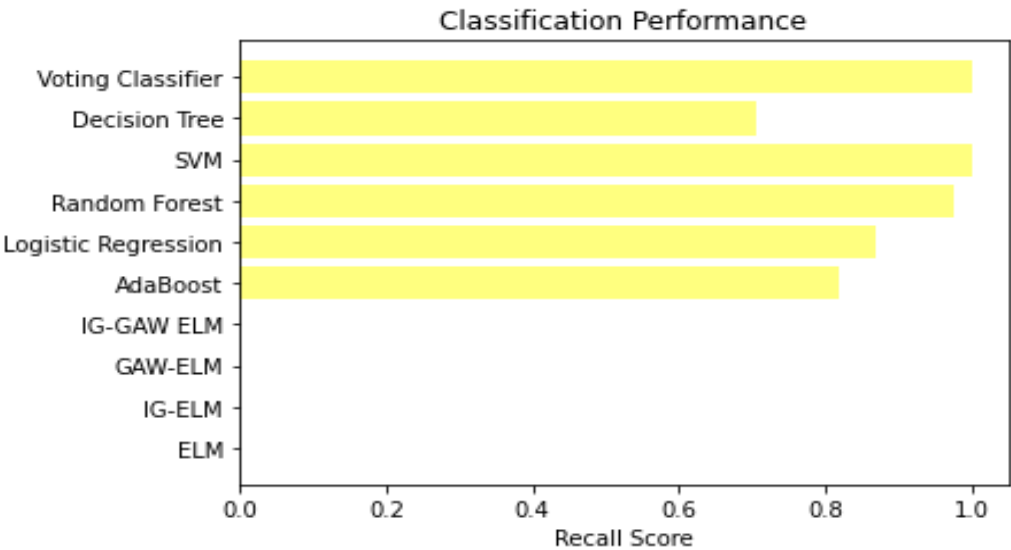
**Figure 6.2 Accuracy Comparison Graph**

A precision comparison graph, illustrating the precision scores achieved by various models or methods. This graph visually represents the precision performance of each model or method



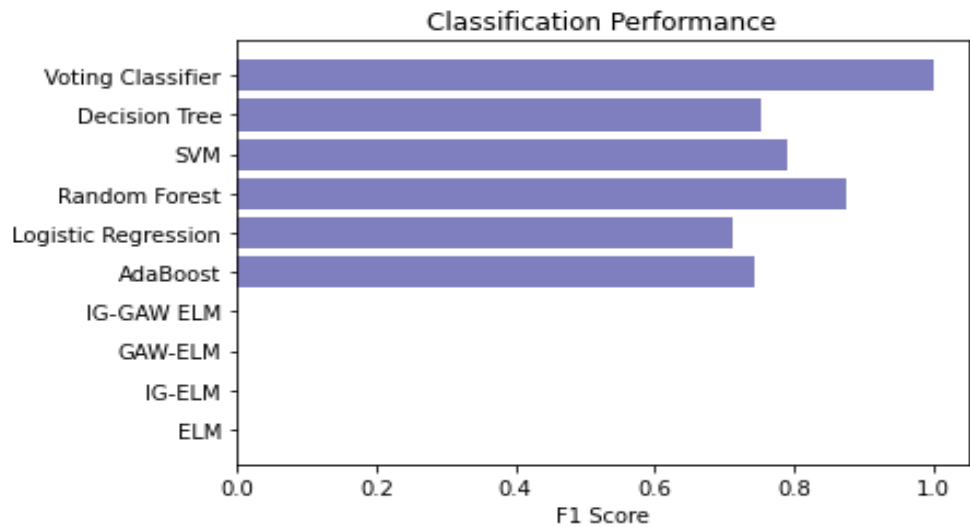
**Figure 6.3 Precision Comparison Graph**

A recall comparison graph, depicting the recall scores achieved by different models or methods. This graph visually presents the ability of each model or method to correctly identify positive cases out of all actual positive cases.



**Figure 6.4 Recall Comparison Graph**

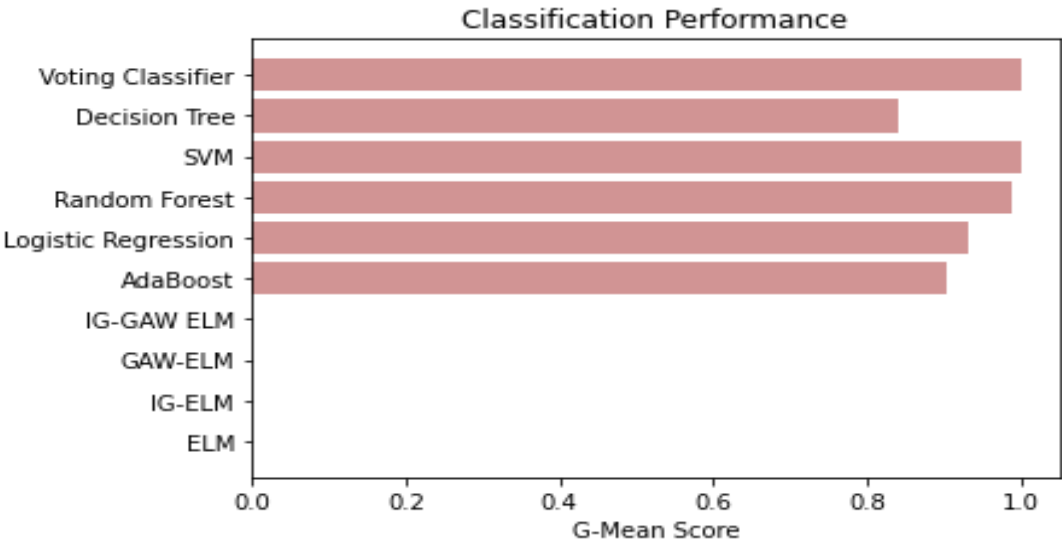
An F1 score comparison graph, showcasing the harmonic mean of precision and recall scores achieved by different models or methods. This graph visually illustrates the overall performance of each model or method in terms of balancing precision and recall.



**Figure 6.5 F1 Score Comparison Graph**

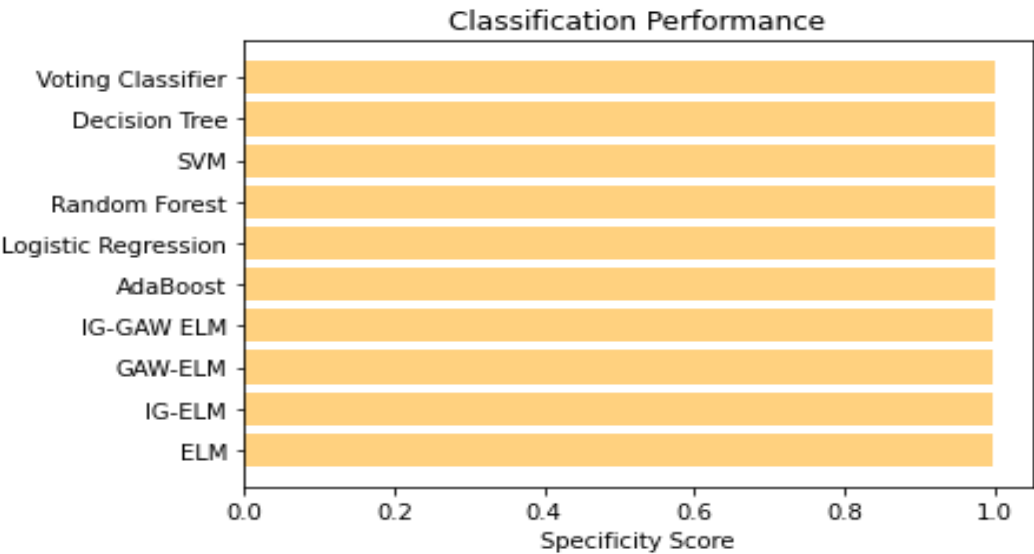


A G-mean comparison graph, illustrating the geometric mean scores achieved by different models or methods. This graph visually represents the balance between sensitivity and specificity achieved by each model or method.



**Figure 6.6 G-Mean Comparison Graph**

A Specificity comparison graph, showcasing the specificity scores achieved by different models or methods. This graph visually illustrates the ability of each model or method to correctly identify negative cases out of all actual negative cases



**Figure 6.7 Specificity Comparison Graph**

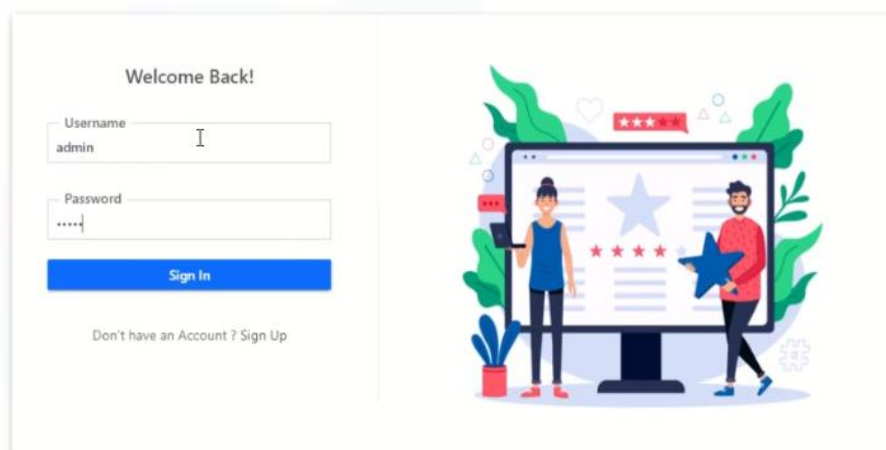
## 6.3 Output Screens

As soon as the model is executed, the following screen is shown:



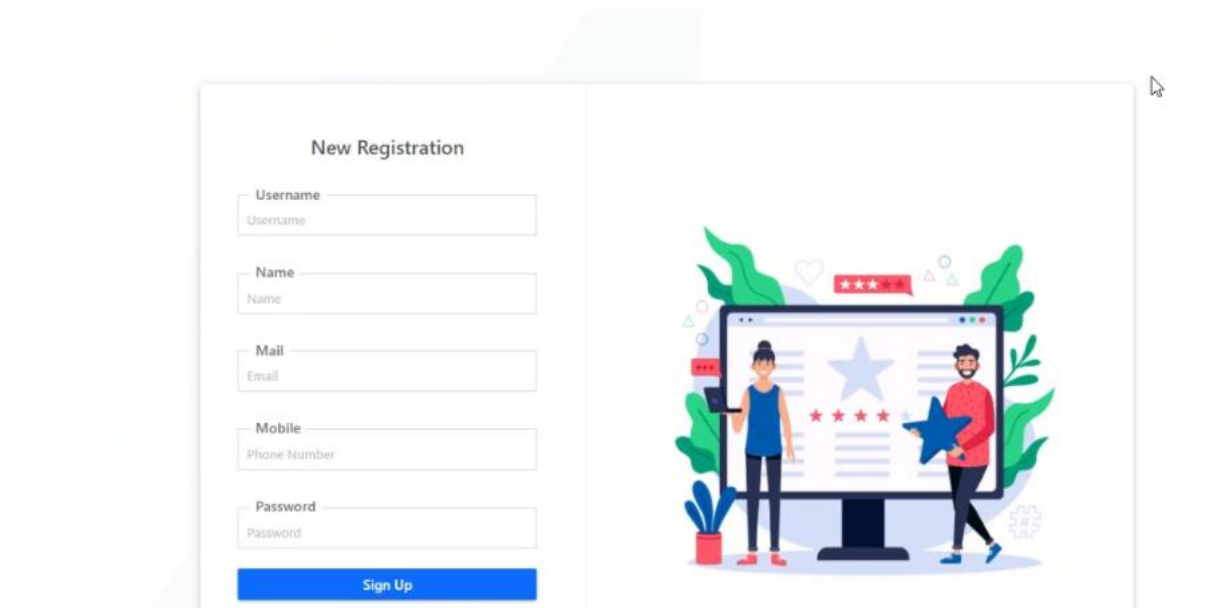
**Screen 6.1 Home Page**

After clicking on "Sign Up" on the homepage, the following screen is displayed:



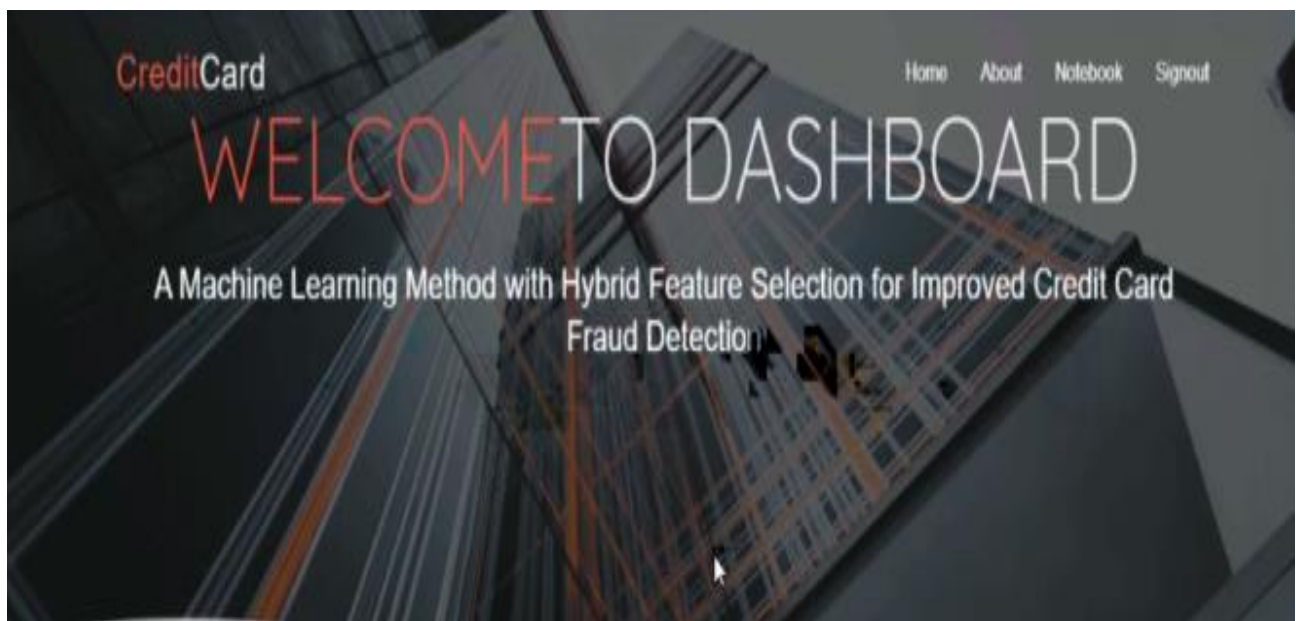
**Screen 6.2 Login Page**

If a user doesn't have an account, they can click on "Don't have an account" on the login page. Upon doing so, they will be directed to a following screen where they can register and create a new account.

The image shows a 'New Registration' form on the left and a colorful illustration on the right. The form has fields for Username, Name, Mail, Mobile, and Password, each with a placeholder text. A blue 'Sign Up' button is at the bottom. The illustration depicts two people standing next to a large monitor displaying a star rating, surrounded by green foliage and decorative elements.

### Screen 6.3 Registration Page

After successfully registering, the user will be redirected to the login page to access their account. After logging in, they will be directed to the following screen:



### Screen 6.4 Main Page

When the user clicks on "About" on the main page, the following screen is displayed:



## Screen 6.5 About Page

To predict the output on the main page, input values are required on the following screen. Users can input relevant data, and the system generates predictions based on that information.

**CreditCard** Home About Notebook Signup

8.584971796

V4: 9.505593515

V5: -13.79381853

V8: 7.517343904

V10: -14.11918444

V14: -9.373858584

V16: -19.23626237

V17: 1.190738695

V21: -2.018575249

V27: -1.04280417

V28: -1.04280417

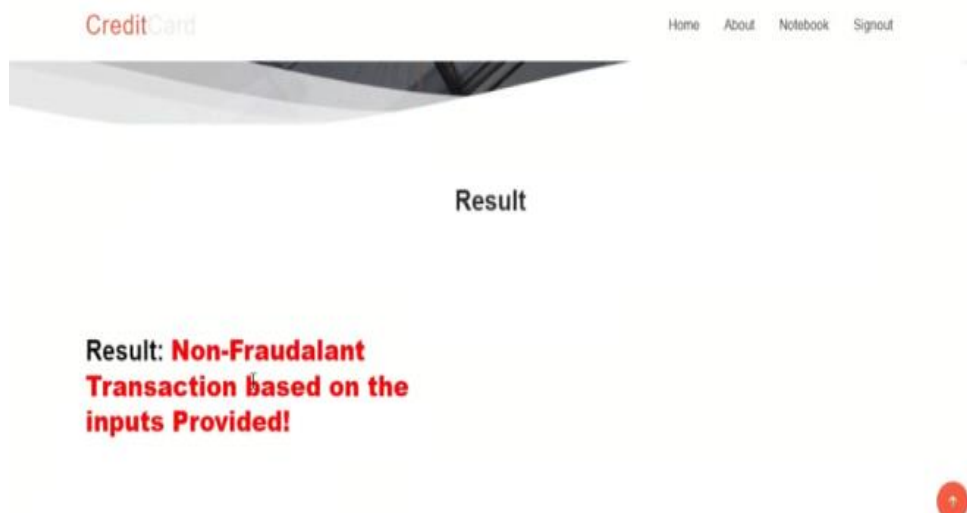
predict

## Screen 6.6 Upload Input Values

Based on the input values provided, the system predicts whether the transaction is fraudulent or non-fraudulent. The predicted result is then displayed accordingly, informing users whether the transaction is flagged as fraudulent or deemed legitimate.



**Screen 6.7 Predict Result As Fraudalant Transaction**



**Screen 6.8 Predict Result as Non-Fraudalant Transaction**

## 6.4 Comparison

Comparison	Existing System	Proposed System
Feature Selection	Hybrid approach utilizing Information Gain (IG) for attribute ranking and Genetic Algorithm (GA) wrapper for feature selection. Top-ranked features identified by IG are input into GA for optimal subset selection.	Hybrid technique integrating filter and wrapper steps to streamline feature relevance. Investigates Full Feature, IG, GAW, and IG-GAW ensemble methods.
Learning Algorithm	Extreme Learning Machine (ELM) utilized within GA wrapper in the existing system.	ELM employed alongside diverse ML models including AdaBoost, Logistic Regression, Random Forest, SVM, Decision Tree, and Voting Classifier. Ensemble method employed for improved accuracy and robustness.
Evaluation Metrics	Sensitivity , Specificity, and G-mean are evaluated to address imbalanced classification issues.	Evaluation metrics expanded to include Accuracy, Precision, Recall, F1-Score, Specificity, and G-mean. Provides a more comprehensive assessment of fraud detection effectiveness.
Focus	Focuses on addressing redundant and irrelevant features through hybrid feature-selection approach. Emphasizes G-mean for assessing fraud detection effectiveness.	Aims to enhance credit card fraud detection amidst the growth of electronic commerce. Introduces ensemble techniques and evaluates multiple ML models for improved accuracy and robustness. Considers a wider

		range of evaluation metrics for a more thorough analysis.
Advancements	Implements GA wrapper with ELM and G-mean evaluation metric in existing system to handle imbalanced classification.	Introduces a comprehensive system with various feature extraction methods, ensemble techniques, and a broader range of evaluation metrics. Offers insights into optimizing fraud detection in electronic commerce.
Overall Contribution	Contributes to addressing challenges in credit card fraud detection by optimizing feature selection and evaluating effectiveness using appropriate metrics.	Provides advancements in credit card fraud detection by exploring a wider array of techniques and metrics, enhancing the accuracy and robustness of fraud detection systems in electronic commerce environments.

**Table 6.1 Comparison Table**

The Existing System achieved a sensitivity and specificity of 0.997 and 0.994, respectively, outperforming other baseline techniques and methods in the recent literature.

The Proposed method evaluates the performance of Extreme Learning Machine (ELM) and diverse ML models, such as AdaBoost, Logistic Regression, Random Forest, SVM, Decision Tree, and Voting Classifier. Notably, the extension of the research incorporates an ensemble method, demonstrating superior accuracy by combining predictions from multiple models. The Voting Classifier, in particular, attains a remarkable 100% accuracy a specificity of 1.00, a precision of 1.00, recall of 1.00, f1\_score of 1.00.

## **CHAPTER 7**

### **CONCLUSION AND SUMMARY**

#### **7.1 Conclusion**

In conclusion, the study addresses the intricate challenge of detecting fraudulent credit card transactions by proposing a novel hybrid approach. Leveraging the strengths of various feature-selection methods and machine learning (ML) techniques, including information gain, genetic algorithms, and extreme learning machines, the proposed system offers a comprehensive solution.

The hybrid feature-selection technique, integrating both filter and wrapper steps, ensures the utilization of pertinent features, addressing the issue of suboptimal classifier performance attributed to redundant and irrelevant data. The evaluation of diverse ML models, such as AdaBoost, Logistic Regression, Random Forest, SVM, Decision Tree, and the Voting Classifier, in conjunction with the ensemble approach, highlights the system's adaptability and robustness.

The exploration of feature extraction methods, including Full Feature, IG, GAW, and IG-GAW, further contributes to the system's efficiency in credit card fraud detection. By combining cutting-edge methodologies and embracing a holistic perspective, the proposed system stands as a valuable contribution to the field. The study's findings offer insights that can significantly enhance the detection rate of fraudulent activities in electronic commerce, providing a reliable and adaptable solution for real-world scenarios in the dynamic landscape of digital payments.

#### **7.2 Future Enhancements**

The future scope involves expanding research by exploring additional combinations of evolutionary algorithms and machine learning-based feature selection techniques to advance credit card fraud detection capabilities. Further efforts will focus on incorporating novel aspects and refining the proposed hybrid approach. Obtaining more recent and diverse datasets will be a priority to ensure ML models are trained on up-to-date information, enhancing their adaptability to evolving fraud patterns and strengthening the overall effectiveness of fraud detection systems in real-world scenarios.



## REFERENCES

- [1] Femila Roseline, J.; Naidu, G.; Samuthira Pandi, V.; Alamelu alias Rajasree, S.; Mageswari, N. Autonomous credit card fraud detection using machine learning approach. *Comput. Electr. Eng.* 2022, 102, 108132.
- [2] Alharbi, A.; Alshammari, M.; Okon, O.D.; Alabrah, A.; Rauf, H.T.; Alyami, H.; Meraj, T. A Novel text2IMG Mechanism of Credit Card Fraud Detection: A Deep Learning Approach. *Electronics* 2022, 11, 756.
- [3] Bin Sulaiman, R.; Schetinin, V.; Sant, P. Review of Machine Learning Approach on Credit Card Fraud Detection. *Hum.-Centric Intell. Syst.* 2022, 2, 55–68.
- [4] Wang, D.; Chen, B.; Chen, J. Credit card fraud detection strategies with consumer incentives. *Omega* 2019, 88, 179–195.
- [5] Nandi, A.K.; Randhawa, K.K.; Chua, H.S.; Seera, M.; Lim, C.P. Credit card fraud detection using a hierarchical behavior-knowledge space model. *PLoS ONE* 2022, 17, e0260579.
- [6] Ileberi, E.; Sun, Y.; Wang, Z. Performance Evaluation of Machine Learning Methods for Credit Card Fraud Detection Using SMOTE and AdaBoost. *IEEE Access* 2021, 9, 165286–165294.
- [7] Rtayli, N.; Enneya, N. Enhanced credit card fraud detection based on SVM-recursive feature elimination and hyper-parameters optimization. *J. Inf. Secur. Appl.* 2020, 55, 102596.
- [8] Huebner, J.; Fleisch, E.; Ilic, A. Assisting mental accounting using smartphones: Increasing the salience of credit card transactions helps consumer reduce their spending. *Comput. Hum. Behav.* 2020, 113, 106504.
- [9] Han, S.; Zhu, K.; Zhou, M.; Cai, X. Competition-Driven Multimodal Multiobjective Optimization and Its Application to Feature Selection for Credit Card Fraud Detection. *IEEE Trans. Syst. Man Cybern. Syst.* 2022, 52, 7845–7857.
- [10] Ileberi, E.; Sun, Y.; Wang, Z. A machine learning based credit card fraud detection using the GA algorithm for feature selection. *J. Big Data* 2022, 9, 24.

## Appendix

### Paper Publication



# CERTIFICATE

OF PUBLICATION

## International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

*(A Monthly, Peer Reviewed, Referred, Multidisciplinary, Scholarly Indexed, Open  
Access Journal since 2012)*



The Board of IJIRSET is hereby awarding this certificate to

**GADDAM SHARMILA**

**Undergraduate Student of Computer Science and Engineering, Vasireddy  
Venkatadri Institute of Technology, Guntur, A.P., India**

***In Recognition of publication of the paper entitled***

***“A Machine Learning method with Hybrid Feature Selection  
for Improved Credit Card Fraud Detection”***

***in IJIRSET, Volume 13, Issue 3, March 2024***



e-ISSN: 2319-8753  
p-ISSN: 2347-6710

**ISSN** INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

*P. Kumar*  
**Editor-in-Chief**

www.ijirset.com    ijirset@gmail.com



# CERTIFICATE

OF PUBLICATION

## International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

*(A Monthly, Peer Reviewed, Referred, Multidisciplinary, Scholarly Indexed, Open  
Access Journal since 2012)*



The Board of IJIRSET is hereby awarding this certificate to

**BATHULA MUNNA**

**Undergraduate Student of Computer Science and Engineering, Vasireddy  
Venkatadri Institute of Technology, Guntur, A.P., India**

***In Recognition of publication of the paper entitled***

***“A Machine Learning method with Hybrid Feature Selection  
for Improved Credit Card Fraud Detection”***

***in IJIRSET, Volume 13, Issue 3, March 2024***



e-ISSN: 2319-8753  
p-ISSN: 2347-6710

**ISSN** INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

*P. Kumar*  
**Editor-in-Chief**

www.ijirset.com    ijirset@gmail.com

# CERTIFICATE

OF PUBLICATION

## International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

*(A Monthly, Peer Reviewed, Referred, Multidisciplinary, Scholarly Indexed, Open  
Access Journal since 2012)*



The Board of IJIRSET is hereby awarding this certificate to

**DIYYAKOLU VAMSI**

**Undergraduate Student of Computer Science and Engineering, Vasireddy  
Venkatadri Institute of Technology, Guntur, A.P., India**

***In Recognition of publication of the paper entitled***

***“A Machine Learning method with Hybrid Feature Selection  
for Improved Credit Card Fraud Detection”***

***in IJIRSET, Volume 13, Issue 3, March 2024***



e-ISSN: 2319-8753  
p-ISSN: 2347-6710

**ISSN** INTERNATIONAL  
STANDARD  
SERIAL  
NUMBER  
INDIA

*P. Kumar*  
**Editor-in-Chief**

www.ijirset.com    ijirset@gmail.com