A

Major Project Report

On

UPI FRAUD DETECTION USING MACHINE LEARING

Submitted to ***CMREC, HYDERABAD***

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)

Submitted By

K.NIHALINI (218R1A6793)

K.SAICHARAN (218R1A6790)

MERAJ AHMED (218R1A67A9)

Under the Esteemed guidance of

Mrs. A.Shravani

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. Hyderabad501 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 **"UPI Fraud Detection using Machine Learning"** is a Bonafide work carried out by

K.NIHALINI	**(218R1A6793)**
K.SAICHARAN	**(218R1A6790)**
MERAJ AHMED	**(218R1A67A9)**

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**
Mrs. A. Shravani	**Mr. B. Kumara Swamy**	**Dr. M. Laxmaiah**	
Assistant Professor CSE (Data Science), CMREC	Assistant Professor CSE (Data Science), CMREC	Professor & H.O.D CSE (Data Science), CMREC	

DECLARATION

This is to certify that the work reported in the present Major project entitled " **UPI Fraud Detection using Machine Learning"** is a record of Bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, UGC Autonomous, 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.

K.NIHALINI	**(218R1A6793)**
K.SAICHARAN	**(218R1A6790)**
MERAJ AHMED	**(218R1A67A9)**

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 **Mrs. A.Shravani** , Assistant Professor, Internal Guide, Department of CSE(DS), for his/ her 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 **Mr. B .Kumara Swamy** Assistant Professor ,CSE(DS) Department , Minor Project Coordinator for his 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.

K.NIHALINI	**(218R1A6793)**
K.SAICHARAN	**(218R1A6790)**
MERAJ AHMED	**(218R1A67A9)**

ABSTRACT

Unified Payments Interface (UPI) transactions have become a crucial part of digital banking, making them a prime target for fraud. This project presents a machine learning-driven fraud detection system for UPI transactions, leveraging multiple classification models, including Logistic Regression, Decision Tree, XGBoost, SVM, KNN, and Naïve Bayes. We employ ensemble techniques such as Voting and Stacking Classifiers to enhance accuracy. The dataset undergoes preprocessing, feature selection, and exploratory data analysis before model training. Our results show that while XGBoost achieves 98.2% accuracy, the Stacking Classifier surpasses it with 99.4% accuracy. The system integrates a Flask-based frontend with user authentication using SQLite, allowing real-time fraud detection. User inputs are preprocessed and classified using the trained model, with predictions displayed through the interface. This project demonstrates the effectiveness of ensemble learning in fraud detection, providing a robust and scalable approach to securing digital transactions.

TABLE OF CONTENTS

1. INTRODUCTION

The rapid growth of digital banking has led to the widespread adoption of the Unified Payments Interface (UPI), a real-time payment system that enables seamless transactions between banks. However, the increasing volume of UPI transactions has also attracted cybercriminals, making fraud detection a critical concern. Traditional rule-based fraud detection systems often fail to adapt to evolving fraudulent tactics, necessitating the use of machine learning-based solutions to identify and mitigate fraudulent activities effectively.

This project presents a comprehensive machine learning-driven fraud detection system for UPI transactions, employing multiple classification models to enhance detection accuracy. The models used include Logistic Regression, Decision Tree, XGBoost, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Naïve Bayes. To further improve performance, ensemble techniques such as Voting and Stacking Classifiers are implemented. Through rigorous experimentation, we find that while XGBoost achieves an impressive accuracy of 98.2%, the Stacking Classifier surpasses it with a remarkable 99.4% accuracy, demonstrating the effectiveness of combining multiple models.

The dataset undergoes extensive preprocessing, including handling missing values, feature selection, and exploratory data analysis (EDA) to ensure high-quality input for model training. By leveraging advanced feature engineering techniques, we enhance the predictive capability of the models, improving fraud detection efficiency.

To provide a user-friendly interface, the system is integrated with a Flask-based web application that allows real-time fraud detection. Users can input transaction details, which are then preprocessed and classified using the trained model. The interface also incorporates authentication mechanisms using SQLite, ensuring secure access to the system.

Overall, this project highlights the power of ensemble learning in fraud detection and offers a scalable solution to enhance the security of UPI transactions, reducing financial losses and building user trust in digital payments.

1.1 Objective:

The objective of this project is to develop a robust machine learning-based fraud detection system for UPI transactions. By leveraging multiple classification models and ensemble techniques, the system aims to achieve high accuracy, enhance security, and provide a real-time fraud detection mechanism through a Flask-based user interface with secure authentication.

1.2 Problem Statement:

- Increasing UPI transaction frauds threaten digital banking security and trust.
- Traditional fraud detection methods struggle to detect evolving fraudulent tactics.
- Machine learning techniques improve fraud detection accuracy and efficiency.
- Multiple classification models enhance fraud identification and minimize false positives.
- Ensemble learning boosts model performance for better fraud detection accuracy.
- Preprocessing and feature selection refine data quality for optimal training.
- Real-time fraud detection is necessary for immediate transaction security.
- A user-friendly interface ensures accessibility and ease of fraud analysis.

1.3 SOFTWARE REQUIREMENTS

1) Software: Anaconda

2) Primary Language: Python

3) Frontend Framework: Flask

4) Back-end Framework: Jupyter Notebook

5) Database: Sqlite3

6) Front-End Technologies: HTML, CSS, JavaScript and Bootstrap4

1.4 HARDWARE REQUIREMENTS

1) Operating System: Windows Only

2) Processor: i5 and above

3) Ram: 8GB and above

4) Hard Disk: 25 GB in local drive

2. FEASIBILITY STUDY

A feasibility study—sometimes called a feasibility analysis or feasibility report—is a way to evaluate whether or not a project plan could be successful. A feasibility study evaluates the practicality of your project plan in order to judge whether or not you're able to move forward with the project.

Types of feasibility studies:

There are five main types of feasibility studies: technical feasibility, financial feasibility, market feasibility (or market fit), operational feasibility, and legal feasibility. Most comprehensive feasibility studies will include an assessment of all five of these areas.

Technical feasibility:

A technical feasibility study reviews the technical resources available for your project. This study determines if you have the right equipment, enough equipment, and the right technical knowledge to complete your project objectives. For example, if your project plan proposes creating 50,000 products per month, but you can only produce 30,000 products per month in your factories, this project isn't technically feasible.

Financial feasibility:

Financial feasibility describes whether or not your project is fiscally viable. A financial feasibility report includes a cost-benefit analysis of the project. It also forecasts an expected return on investment (ROI) and outlines any financial risks.

The goal at the end of the financial feasibility study is to understand the economic benefits the project will drive.

Market feasibility:

The market feasibility study is an evaluation of how your team expects the project's deliverables to perform in the market. This part of the report includes a market analysis, a market competition breakdown, and sales projections.

Operational feasibility:

An operational feasibility study evaluates whether or not your organization is able to complete this project. This includes staffing requirements, organizational structure, and any applicable legal requirements. At the end of the operational feasibility study, your team will have a sense of whether or not you have the resources, skills, and competencies to complete this work.

Legal feasibility:

A legal feasibility analysis assesses whether the proposed project complies with all relevant legal requirements and regulations. This includes examining legal and regulatory barriers, necessary permits, licenses, or certifications, potential legal liabilities or risks, and intellectual property considerations. The legal feasibility study ensures that the project can be completed without running afoul of any laws or incurring undue legal exposure for the organization.

3. LITERATURE SURVEY

3.1 Fraud detection model for illegitimate transactions:

https://kurj.kab.ac.ug/index.php/1/article/view/102

Due to advancements in network technologies, digital security is becoming a top priority worldwide. This project aims to study how machine learning techniques can be used to learn patterns in fraudulent and legitimate transactions in order to detect fraudulent transactions using Python programming language on Jupyter notebook as the integrated development environment (IDE). Scikit-learn was used to process the algorithm, and Streamlit and Heroku platforms were used for deployment of the algorithms. This was incorporated into a web application that allows the user to upload data that is analyzed by the system to detect fraud. The Classification report and Confusion matrix are used to evaluate each model's accuracy. The random forest model gave an accuracy of 99.95 %. At the end of this study, a web-based application was developed to allow users upload data and also to detect fraudulent online based transaction.

3.2 Fraud detection with natural language processing:

https://www.researchgate.net/publication/372466905_Fraud_detection_with_natural_language_processing

Automated fraud detection can assist organisations to safeguard user accounts, a task that is very challenging due to the great sparsity of known fraud transactions. Many approaches in the literature focus on credit card fraud and ignore the growing field of online banking. However, there is a lack of publicly available data for both. The lack of publicly available data hinders the progress of the field and limits the investigation of potential solutions. With this work, we: (a) introduce FraudNLP, the first anonymised, publicly available dataset for online fraud detection, (b) benchmark machine and deep learning methods with multiple evaluation measures, (c) argue that online actions do follow rules similar to natural language and hence can be approached successfully by natural language processing methods.

3.3 Application of Artificial Intelligence for Fraudulent Banking Operations Recognition:

https://www.mdpi.com/2504-2289/7/2/93

This study considers the task of applying artificial intelligence to recognize bank fraud. In recent years, due to the COVID-19 pandemic, bank fraud has become even more common due to the massive transition of many operations to online platforms and the creation of many charitable funds that criminals can use to deceive users. The present work focuses on machine learning algorithms as a tool well suited for analyzing and recognizing online banking transactions. The study's scientific novelty is the development of machine learning models for identifying fraudulent banking transactions and techniques for preprocessing bank data for further comparison and selection of the best results. This paper also details various methods for improving detection accuracy, i.e., handling highly imbalanced datasets, feature transformation, and feature engineering. The proposed model, which is based on an artificial neural network, effectively improves the accuracy of fraudulent transaction detection. The results of the different algorithms are visualized, and the logistic regression algorithm performs the best, with an output AUC value of approximately 0.946. The stacked generalization shows a better AUC of 0.954. The recognition of banking fraud using artificial intelligence algorithms is a topical issue in our digital society.

3.4 IMPLEMENTATION OF CASHLESS POLICY TO MINIMIZE FRAUD IN THE GOVERNMENT SECTOR: A SYSTEMATIC REVIEW:

https://www.researchgate.net/publication/365182536_IMPLEMENTATION_OF_CASHLESS_POLICY_TO_MINIMIZE_FRAUD_IN_THE_GOVERNMENT_SECTOR_A_SYSTEMATIC_REVIEW

Cashless financial transactions require information technology to transfer funds for the payment of needs, expenditures, and local government revenues. The potential that causes fraud and corruption can be minimized with non-cash transactions because financial transactions are more transparent. This study aims to identify fraud prevention strategies by implementing a cashless policy. This research was conducted systematically through an article search engine using the keywords "Cashless" and "Fraud," which was then entered into the Scopus journal search engine based on secondary data in the publish or perish application 8. Then, journals and articles were selected based on the title theme and looked at the quality of the article. The result of this study is that local government payment system innovations have led to changes in payment options by switching to non-cash transactions that are safer, more effective, and efficient. Cashless payments can prevent corrupt practices such as money laundering, bribery, and commissions for services or procurement. People will use non-cash transactions if the local government forces non-cash payments in urban and rural areas with the support of the internet and promising technology. Local governments must report further non-cash payments and prioritize non-cash payments and make policies and innovations such as payment of social assistance to the community using digital money and payment of retribution using non-cash to increase local revenue. Keywords: Cashless, Corruption, Fraud.

3.5 Digital payment fraud detection methods in digital ages and Industry 4.0:

https://www.researchgate.net/publication/359099847_Digital_payment_fraud_detection_methods_in_digital_ages_and_Industry_40

The advent of the digital economy and Industry 4.0 enables financial organizations to adapt their processes and mitigate the risks and losses associated with the fraud. Machine learning algorithms facilitate effective predictive models for fraud detection for Industry 4.0. This study aims to identify an efficient and stable model for fraud detection platforms to be adapted for Industry 4.0. By leveraging a real credit card transaction dataset, this study proposes and compares five different learning models: logistic regression, decision tree, k-nearest neighbors, random forest, and autoencoder. Results show that random forest and logistic regression outperform the other algorithms. Besides, the undersampling method and feature reduction using principal component analysis could enhance the results of the proposed models. The outcomes of the studies positively ascertain the effectiveness of using features selection and sampling methods for tackling business problems in the new age of digital economy and industrial 4.0 to detect fraudulent activities.

\Author(s) & Year	Title	Methodology	Key Findings	Limitations
Ibrahim Musibau Adekunle (2023)	Fraud detection model for illegitimate transactions	Data preprocessing, model training, evaluation, web app development, and deployment.	High accuracy, effective fraud detection, web deployment, user-friendly, real-time analysis.	Limited dataset, potential biases, real-time scalability, false positives, deployment dependencies.
John Pavlopoulos, Alexandros Xenos (2023)	Fraud detection with natural language processing	Dataset creation, anonymization, benchmarking models, evaluation, NLP-based fraud detection approach.	Public fraud dataset, effective NLP approach, strong benchmarks, evaluation metrics validated.	Data sparsity, limited public datasets, NLP assumptions, evaluation constraints, fraud complexity.
Oleksandr Tkachyk (2023)	Application of Artificial Intelligence for Fraudulent Banking Operations Recognition	Data preprocessing, feature engineering, model training, evaluation, ANN, logistic regression, stacking.	AI improves fraud detection, stacked model excels, logistic regression performs well.	Data imbalance, model biases, fraud complexity, real-time challenges, potential false positives.
Jurnal Akuntans (2022)	Implementation of cashless policy to minimize fraud in the government sector: A systematic review.	Systematic review, keyword search, Scopus database, journal selection, secondary data analysis.	Cashless transactions enhance transparency, reduce fraud, prevent corruption, and improve efficiency.	Internet dependency, rural accessibility, policy enforcement, adoption challenges, potential cybersecurity risks.
Alessandro Di Stefano	Digital payment fraud detection methods in digital ages and Industry 4.0	Dataset analysis, model selection, training, evaluation, under-sampling, feature reduction, comparison.	Random forest, logistic regression excel feature reduction, under-sampling improve detection.	Dataset bias, model limitations, real-time challenges, scalability, potential false positives.

4. SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

The existing system for UPI fraud detection relies on traditional rule-based methods, which often struggle to adapt to new fraud patterns and handle large-scale datasets efficiently. These systems typically use predefined thresholds to flag suspicious transactions based on transaction amount, frequency, or location. However, they can miss nuanced fraudulent behavior and are prone to high false-positive rates, leading to user inconvenience. Additionally, they may not handle imbalanced data effectively, resulting in poor fraud detection performance. The system often lacks the adaptability and precision that machine learning models, such as XGBoost, can provide in dynamic and complex fraud detection scenarios.

4.1.1 DISADVANTAGES OF EXISTING SYSTEM:

1. Limited adaptability to new and evolving fraud patterns, leading to missed detections.
2. High false-positive rates, causing unnecessary alarms and user inconvenience.
3. Inefficient handling of large, imbalanced datasets, resulting in poor detection accuracy.
4. Predefined rules struggle to identify complex, sophisticated fraud schemes.
5. Lack of real-time monitoring capabilities, delaying fraud detection and response.

4.2 PROPOSED SYSTEMs:

The proposed system utilizes machine learning models to detect fraudulent UPI transactions efficiently. It integrates multiple classification techniques, including Logistic Regression, Decision Tree, XGBoost, SVM, KNN, and Naïve Bayes, with ensemble methods like Voting and Stacking Classifiers to improve accuracy. The dataset undergoes preprocessing, feature engineering, and exploratory data analysis for enhanced predictive performance. A Flask-based web interface enables real-time fraud detection, allowing users to input transaction details and receive immediate classification results. Secure authentication using SQLite ensures restricted access. This scalable and robust system effectively mitigates fraud risks, ensuring safe and reliable digital transactions.

4.2.1 Advantages of proposed system:

1. Improved fraud detection accuracy using multiple machine learning classification models.
2. Real-time transaction monitoring ensures immediate fraud identification and prevention.
3. Ensemble learning enhances system performance by combining multiple model predictions.
4. Secure authentication mechanisms protect user data from unauthorized access attempts
5. Scalable solution adapts to increasing digital transaction volumes efficiently and effectively.

4.2.2 Extension:

As an extension we applied an ensemble method combining the predictions of multiple individual models to produce a more robust and accurate final prediction. However, we can further enhance the performance by exploring other ensemble techniques such as Stacking Classifier which got 99.4 of accuracy, As an extension we can build the front end using the flask framework for user testing and with user authentication.

4.2.3 Advantages of Extension:

- Stacking classifier improves fraud detection accuracy to an outstanding 99.4%.
- Flask-based front end ensures user-friendly interaction with fraud detection system.
- Real-time fraud detection enables instant response to suspicious transactions efficiently.
- Secure user authentication prevents unauthorized access to fraud detection features.
- Scalable architecture allows easy integration with existing digital banking infrastructures.

4.3 FUNCTIONALREQUIREMENTS

1. Data Collection
2. Data Pre-processing
3. Training and Testing
4. Modelling
5. Predicting

4.4. NON FUNCTIONALREQUIREMENTS

Performance

The system should efficiently process large volumes of IoT traffic in real time, ensuring minimal latency in detection and response to intrusions, thereby maintaining smooth operation and user experience without significant delays.

Scalability

The architecture must support scalability to accommodate the increasing number of IoT devices and data traffic without degradation in performance. It should handle expansion easily, allowing for future growth in network size and complexity.

Reliability

The system should demonstrate high reliability, ensuring continuous operation and accurate detection of intrusions even in the presence of network fluctuations or system failures, thereby minimizing downtime and maintaining consistent security coverage.

Usability

The user interface should be intuitive and user-friendly, allowing administrators to easily configure settings, monitor alerts, and manage the system without extensive training, ensuring effective operation and quick decision-making in response to threats.

Security

The system must adhere to strong security protocols to protect sensitive data and configurations from unauthorized access and tampering. Robust authentication and encryption mechanisms should be implemented to safeguard the integrity of the IDS.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

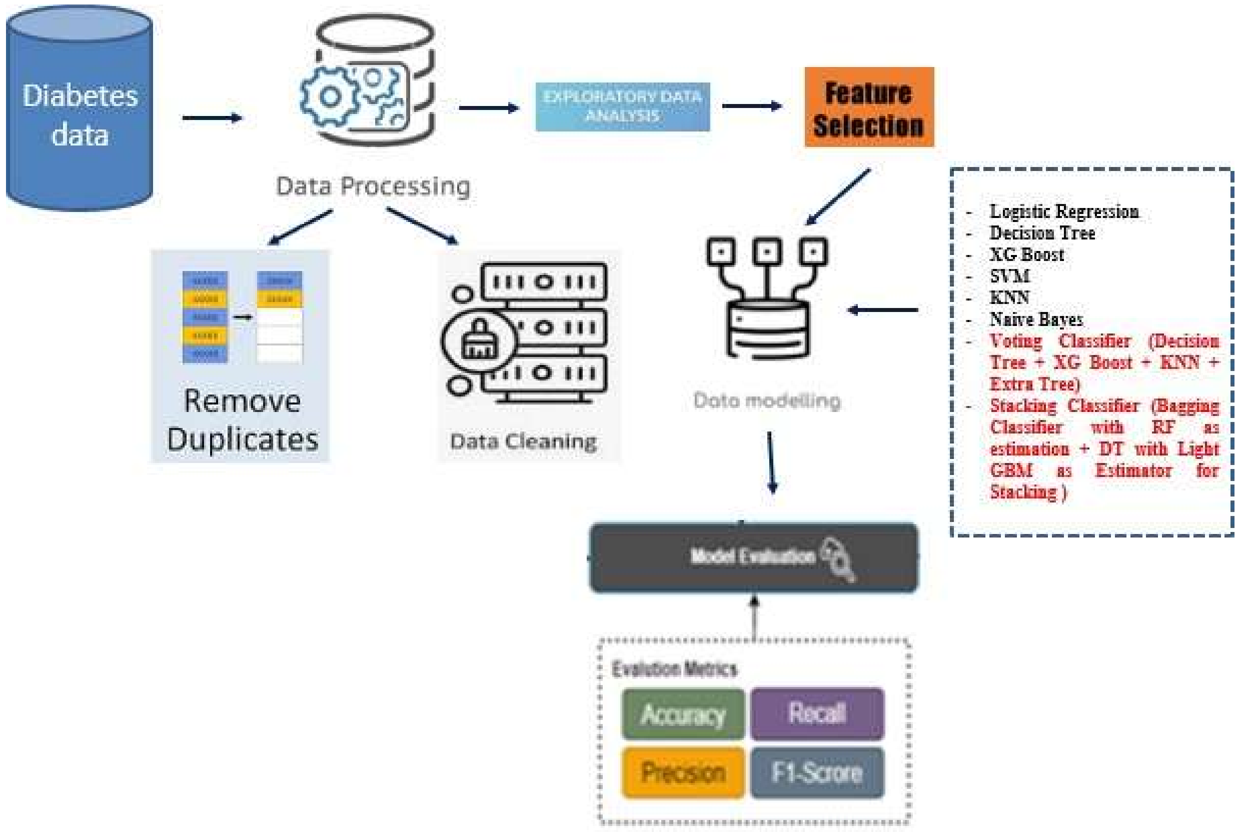


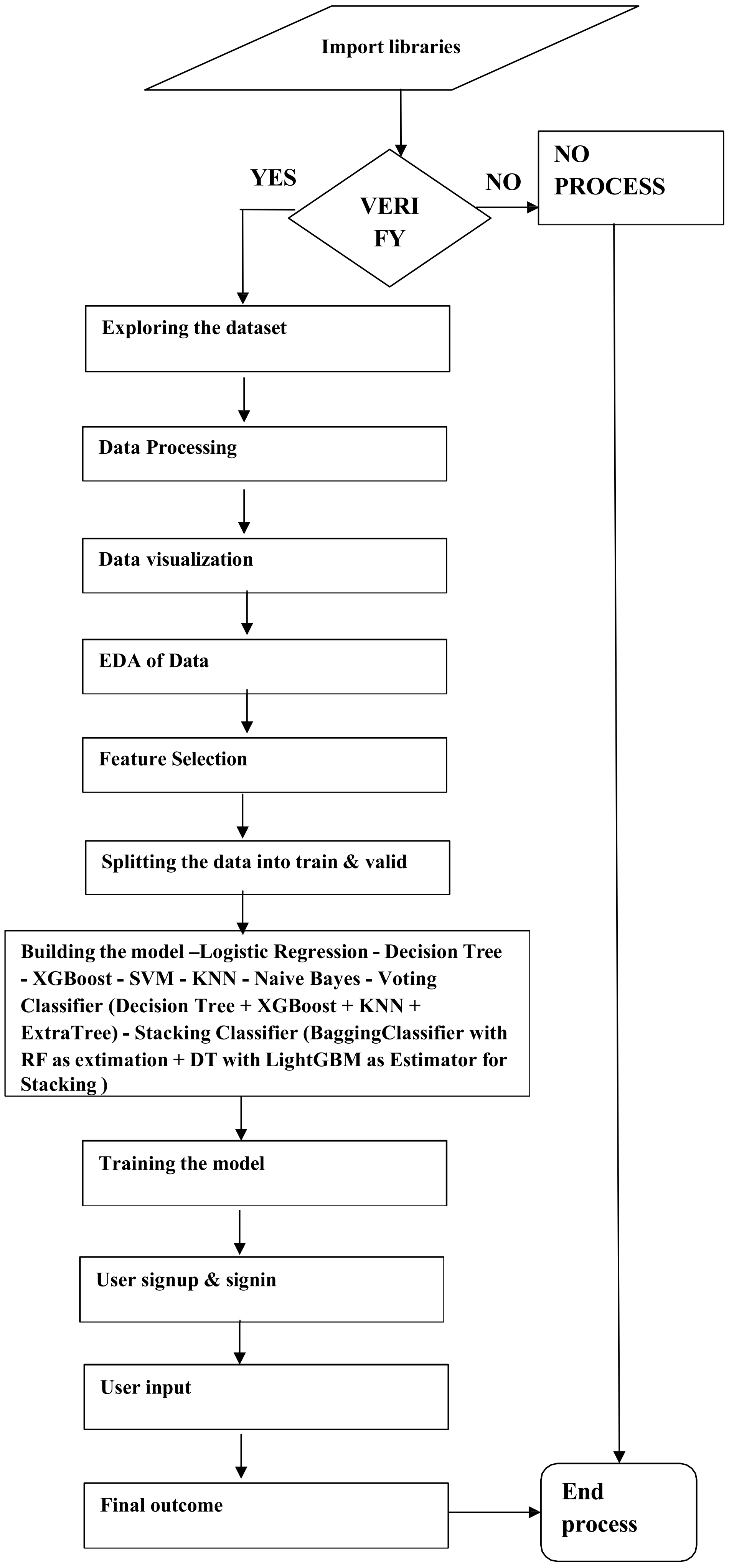
Fig.5.1.1 System architecture

DATA FLOW DIAGRAM:

A Data Flow Diagram (DFD) represents the flow of data within a system, illustrating how data moves between processes, data stores, and external entities. It helps in understanding system functionalities and identifying potential inefficiencies. The DFD of this fraud detection system consists of input data from users, preprocessing and feature extraction, classification using machine learning models, and the final fraud prediction output. The system also integrates real-time monitoring, secure authentication, and a Flask-based interface. By visualizing these processes, DFDs aid in designing efficient, structured, and scalable fraud detection solutions for digital banking security.

Goals of DFD

- To visually represent the flow of data within the fraud detection system.
- To identify key processes, data inputs, and outputs for system analysis
- To ensure a structured approach in designing fraud detection workflows.
- To highlight interactions between users, databases, and machine learning models.
- To facilitate better understanding, optimization, and scalability of the system architecture.


Import libraries
VERI FY
YES
NO
NO PROCESS
Exploring the dataset
Data Processing
Data visualization
EDA of Data
Feature Selection
Splitting the data into train & valid
Building the model –Logistic Regression - Decision Tree - XGBoost - SVM - KNN - Naive Bayes - Voting Classifier (Decision Tree + XGBoost + KNN + ExtraTree) - Stacking Classifier (BaggingClassifier with RF as extimation + DT with LightGBM as Estimator for Stacking)
Training the model
User signup & signin
User input
Final outcome
End process

5.2 UML DIAGRAMS

UML (Unified Modeling Language) diagrams visually represent the structure and behavior of a system. In the context of the fraud detection system, key UML diagrams include use case diagrams to depict user interactions, class diagrams for system components like the Flask frontend, SQLite for user authentication, and machine learning models. Sequence diagrams can illustrate the flow of data during transaction processing, while activity diagrams show the steps in fraud detection. These diagrams provide clarity on system architecture and functionality.

Characteristics of UML

UML provides a standardized set of notations and symbols for creating diagrams, ensuring consistency across projects.

UML diagrams offer a clear visual representation of system components, interactions, and workflows.

It covers both structural (e.g., class, component, deployment diagrams) and behavioral (e.g., use case, sequence, activity diagrams) aspects of a system.

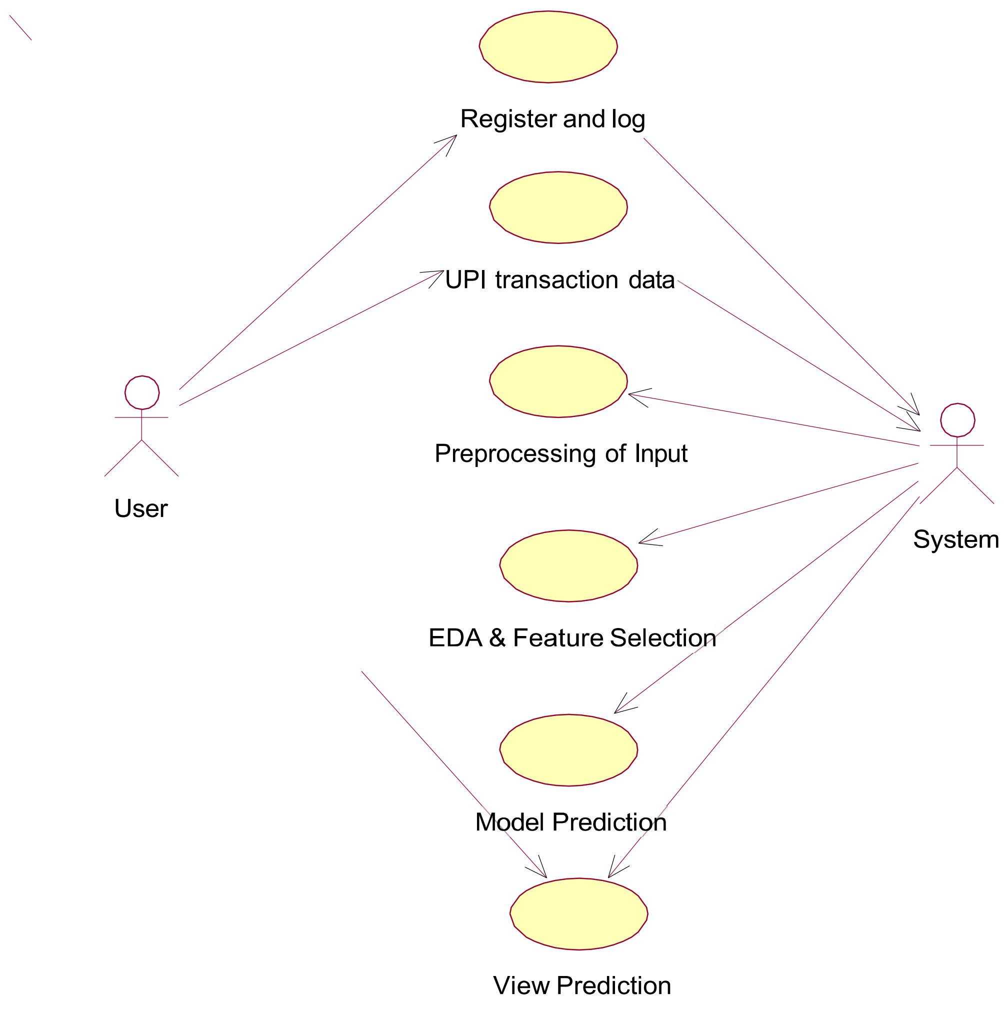
UML can be applied to various types of software systems and domains, from object-oriented design to business processes.

UML allows for different levels of abstraction, from high-level overviews to detailed specifications.

UML is widely supported by various tools, allowing easy integration into different stages of software development.

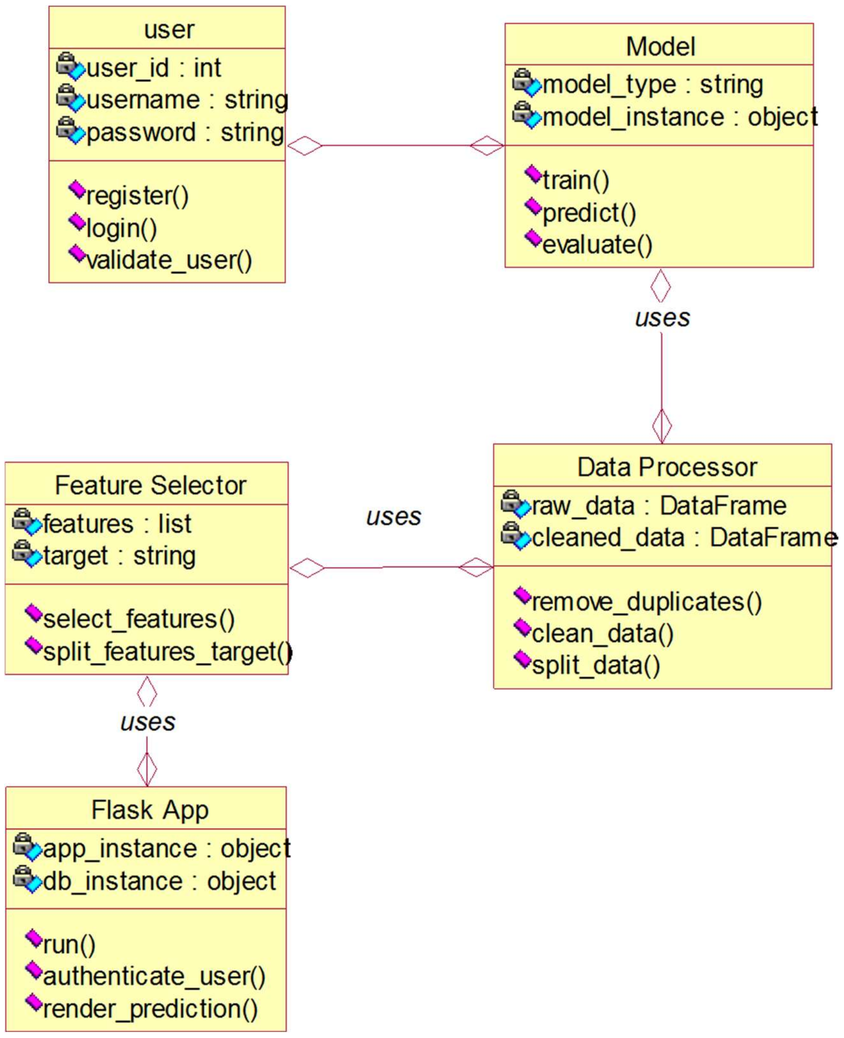
Use Case Diagram

The Use Case Diagram for the UPI fraud detection system illustrates the interactions between users and the system. Key actors include the "User" and the "System." The "User" can perform actions like logging in, entering transaction details, and receiving fraud detection results. The "System" handles user authentication, transaction data preprocessing, model classification, and fraud prediction. It also displays results through the frontend. The diagram highlights the system's focus on real-time fraud detection, ensuring secure UPI transactions through seamless user interactions.



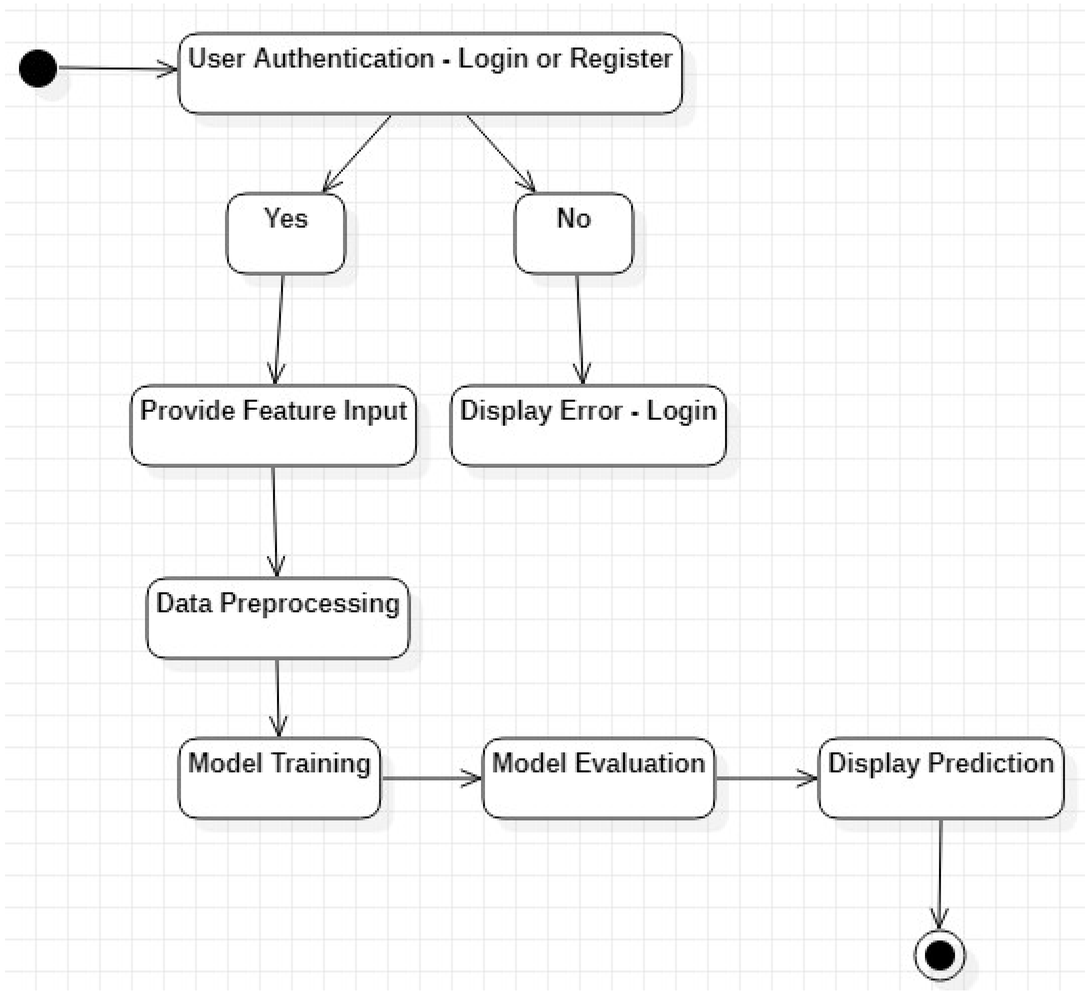
Class Diagram

The Class Diagram for the UPI fraud detection system includes key classes like User, Transaction, Model, and Prediction. The User class manages user authentication and stores user details. The Transaction class contains transaction attributes such as amount, merchant, and timestamp. The Model class handles different classification models (Logistic Regression, XGBoost, etc.) and model training. The Prediction class processes input data and outputs fraud detection results. Relationships between these classes show dependencies for data flow and functionality within the system.



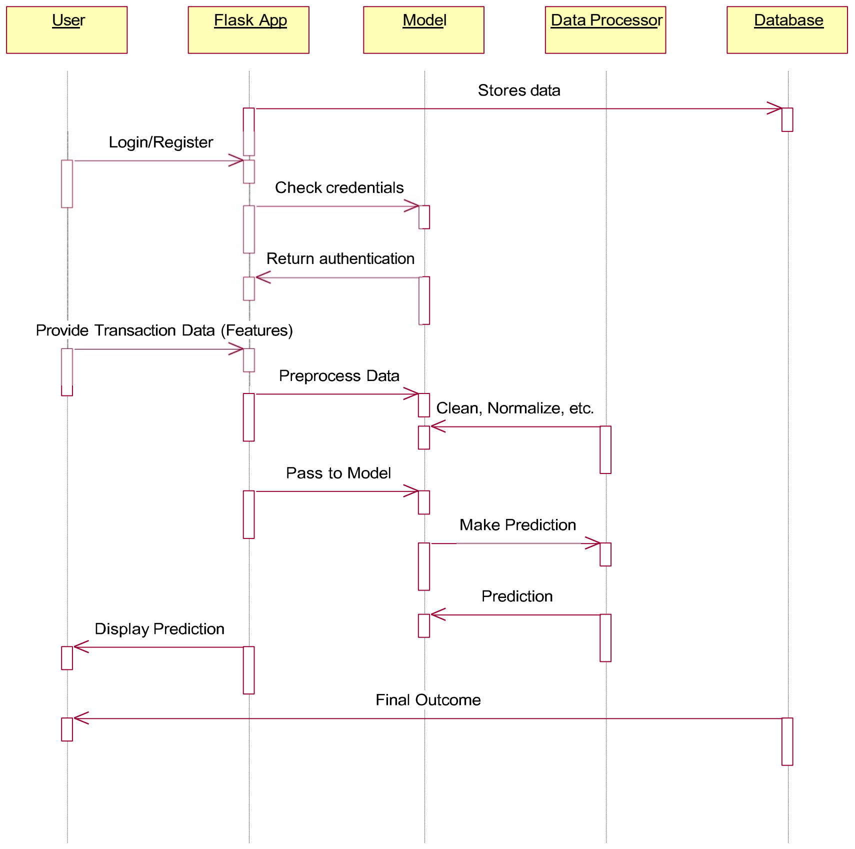
Activity Diagram

The Activity Diagram for the UPI fraud detection system outlines the flow of activities from user interaction to fraud prediction. The process begins with user login and authentication, followed by entering transaction details. The system preprocesses the data, selects features, and passes it through trained classification models. The models predict whether the transaction is fraudulent or not. The result is displayed to the user, completing the transaction flow. The diagram also includes error handling and ensures smooth system operations for real-time fraud detection.



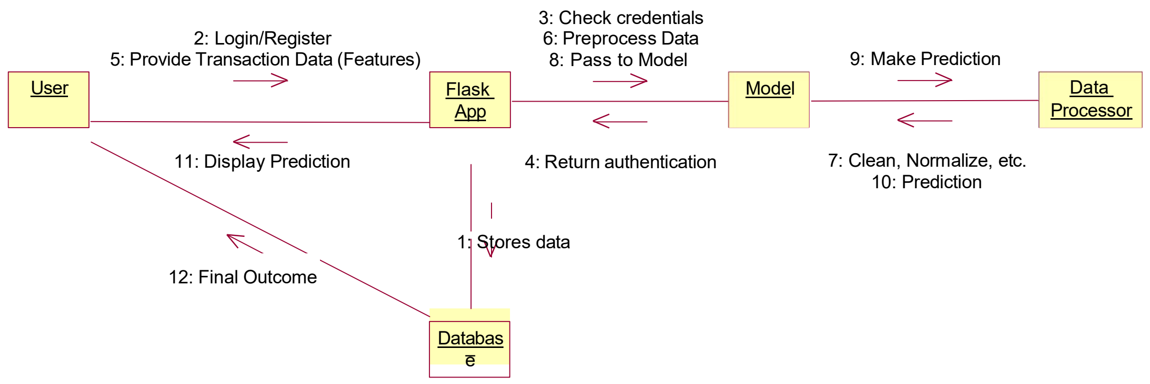
Sequence Diagram

The Sequence Diagram for the UPI fraud detection system illustrates the interactions between the User, Frontend, Backend, and Model classes. The User initiates the process by logging in, followed by entering transaction details via the Frontend. The Frontend sends the data to the Backend, which handles data preprocessing and invokes the trained Model for classification. The Model predicts the fraud status, and the result is sent back to the Frontend, which then displays the fraud detection outcome to the User.



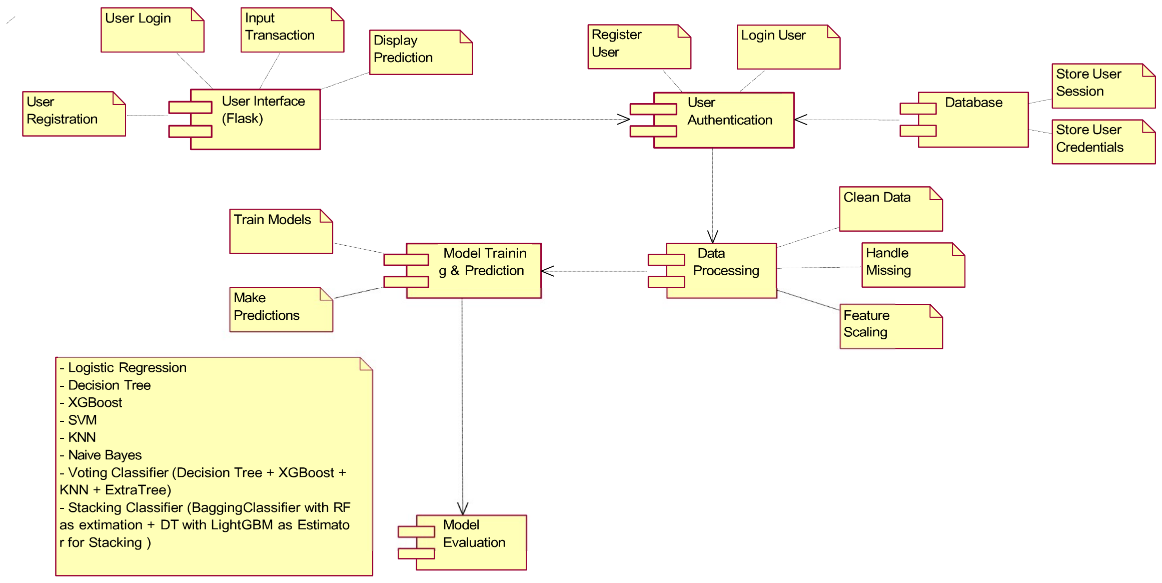
Collaboration Diagram

The Collaboration Diagram for the UPI fraud detection system depicts the interactions between key objects: User, Frontend, Backend, and Model. The User initiates the process by logging in and entering transaction details via the Frontend. The Frontend collaborates with the Backend to send the transaction data for preprocessing. The Backend then interacts with the Model to perform classification and generate fraud predictions. Finally, the Backend sends the prediction result back to the Frontend, which presents it to the User.



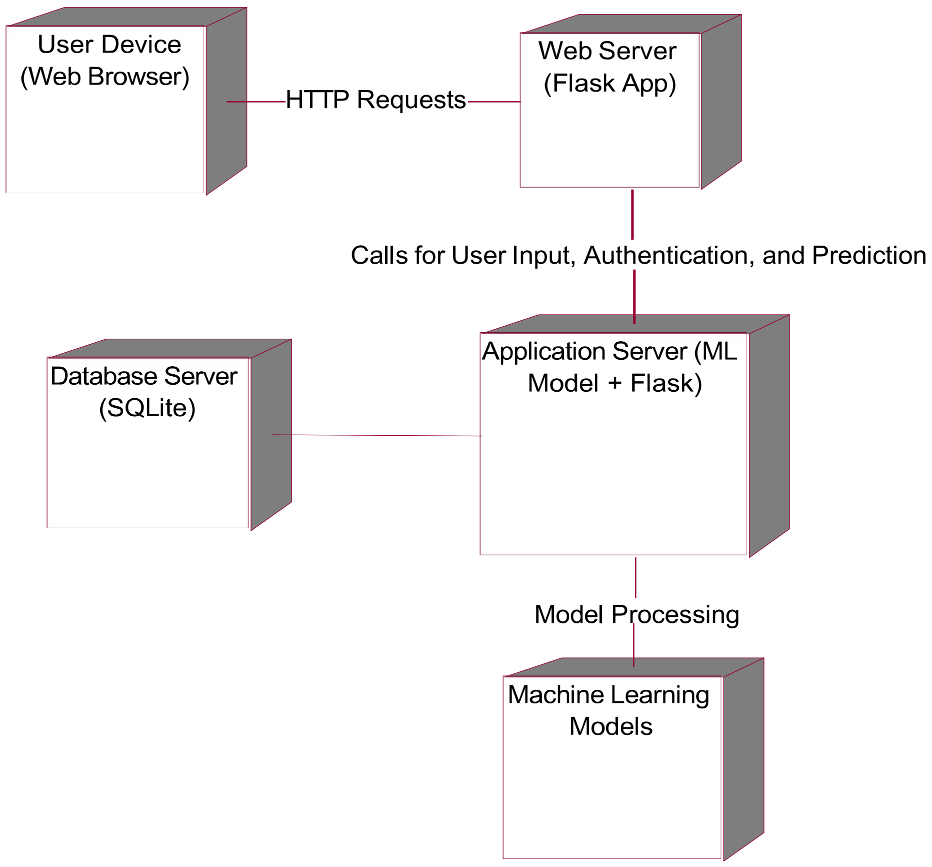
Component Diagram

The Component Diagram for the UPI fraud detection system illustrates the system's main components and their relationships. It includes components like User Interface (Frontend), Authentication Service, Transaction Service, Model Service, and Database. The Frontend component handles user interactions, sending transaction data to the Backend. The Backend includes Authentication for user login and Transaction Service for data preprocessing. The Model Service performs fraud detection using trained models, and the Database stores user and transaction details, supporting the entire system's functionality.



Deployment Diagram

The Deployment Diagram for the UPI fraud detection system depicts the physical deployment of software components on hardware nodes. It includes a User Device (mobile or web) running the Frontend application, connected to a Server hosting the Backend, which manages user authentication, transaction processing, and model interactions. The Database is deployed on a separate server, storing user and transaction data. The Model service resides on the same server as the Backend, handling fraud detection through the trained models. The entire system operates in a cloud-based environment for scalability and security.


User Device (Web Browser)
HTTP Requests
Web Server (Flask App)
Calls for User Input, Authentication, and Prediction
Database Server (SQLite)
Application Server (ML Model + Flask)
Model Processing
Machine Learning Models

6. IMPLEMENTATION

6.1 MODULES:

- **Importing the packages:** Import necessary libraries and packages for data manipulation, visualization, and machine learning tasks.
- **Exploring the dataset – Diabetes data:** Analyze the Diabetes data to understand its structure, features, and potential issues.
- **Data Processing:** Remove duplicate records and perform data cleaning to ensure quality data for training..
- **Feature Selection:** Select important features based on correlation, importance, or other criteria to reduce dimensionality.
- **Splitting the dataset in train and validation:** Divide the dataset into training and validation sets to evaluate model performance accurately.
- **Building the model for all data:** Design machine learning models for both binary and multi-class classification tasks using deep learning.
  - - Logistic Regression - Decision Tree - XGBoost - SVM - KNN - Naive Bayes - Voting Classifier (Decision Tree + XGBoost + KNN + ExtraTree)
- **Training and Building the model:** Train the model using the training dataset, optimizing parameters for performance.
- **Evaluation of all the models with accuracy, precision, recall, F1 score:** Assess model performance using evaluation metrics like accuracy, precision, recall, and F1 score.
- **Comparison graphs are generated with scores of all models:** Generate comparison graphs to visualize the performance metrics of each model for analysis.

- **Frontend is developed with help of Flask Framework, along with Registration and Login setup:** Build a user-friendly frontend using Flask, with features like user registration and login.
- **User gives input as Feature Values:** User provides feature values through the frontend interface for model prediction.
- **The given input is preprocessed for prediction:** Preprocess user input, including normalization and encoding, for compatibility with the trained model.
- **Trained model is used for prediction:** Use the trained model to predict the outcome based on the user's input features.
- **Final outcome is displayed through frontend:** Display the model's prediction result to the user via the Flask frontend interface.

Algorithms:

1. Multi-class Classification Algorithms

Logistic Regression

Logistic Regression is a linear model used for binary classification tasks. It predicts the probability of an outcome using the logistic function, which maps any real-valued number into a probability between 0 and 1. The model estimates the parameters by minimizing a cost function, typically using gradient descent. It works well when the relationship between features and the target is linear and is often used in situations like spam detection and medical diagnosis.

Decision Tree

A Decision Tree is a non-linear model used for both classification and regression. It splits data into subsets based on feature values, creating a tree-like structure where each node represents a feature test, and each leaf represents a class label or value. Trees are built by choosing the feature that maximizes information gain (for classification) or minimizes variance (for regression). Decision trees are interpretable but prone to overfitting without pruning or regularization techniques.

XGBoost

XGBoost (Extreme Gradient Boosting) is a highly efficient implementation of gradient boosting for classification and regression tasks. It builds a series of decision trees, each correcting the errors of the previous one, by minimizing a loss function using gradient descent. XGBoost incorporates regularization (L1 and L2) to prevent overfitting and can handle sparse data efficiently. It’s widely used in competitive machine learning for its speed, accuracy, and ability to handle large datasets.

SVM (Support Vector Machine)

SVM is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates data points of different classes in a high-dimensional space, maximizing the margin between them. SVM can efficiently handle non-linear data using the kernel trick, which maps data into higher dimensions where it becomes linearly separable. It's effective in high-dimensional spaces and often used in text classification and image recognition tasks.

KNN (K-Nearest Neighbors)

KNN is a simple, non-parametric algorithm used for classification and regression. It works by assigning the class or value of a sample based on the majority vote or average of its nearest neighbors. The number of neighbors (k) is a key parameter. KNN is effective in non-linear data but can be computationally expensive with large datasets. It is sensitive to the choice of distance metric and performs poorly with high-dimensional data (curse of dimensionality).

Naive Bayes

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming that features are conditionally independent given the class label. It calculates the probability of a sample belonging to each class and selects the class with the highest probability. Naive Bayes is particularly efficient with text classification tasks, such as spam detection or sentiment analysis. Despite its simplifying assumption of independence, it often performs surprisingly well in practice, especially with high-dimensional data.

Voting Classifier (Decision Tree + XGBoost + KNN + ExtraTree)

A Voting Classifier combines multiple base classifiers (e.g., Decision Tree, XGBoost, KNN, Extra Trees) to make predictions based on a majority or weighted vote. It can be used for classification tasks, where each model independently makes predictions, and the final decision is based on the consensus of the models. This approach enhances accuracy by reducing the risk of overfitting from individual models and is particularly effective when base models complement each other.

Stacking Classifier (BaggingClassifier with RF + DT with LightGBM)

Stacking Classifier is an ensemble technique that combines multiple models (e.g., BaggingClassifier with Random Forest and Decision Tree with LightGBM). The idea is to train base models on the training data and then use a meta-model to learn how to combine their predictions for improved performance. The base models are often trained on different subsets of the data, and the meta-model combines their predictions. Stacking is powerful for complex datasets, often outperforming individual models.

6.2 SAMPLE CODE:

```
# 1. Importing the packages
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import KNNImputer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import VotingClassifier, BaggingClassifier,
ExtraTreesClassifier, RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from sklearn.ensemble import StackingClassifier
from flask import Flask, render_template, request
```

```
import sqlite3

# 2. Exploring the dataset (Assuming diabetes dataset)
df = pd.read_csv('diabetes.csv')  # Change the file name as required
print(df.head())

# 3. Data Processing - Removing Duplicate Data and Drop Cleaning
df = df.drop_duplicates()  # Removing duplicate rows
df = df.dropna()  # Handling missing values

# 4. EDA of Data
import seaborn as sns
import matplotlib.pyplot as plt

# Correlation Matrix
correlation_matrix = df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()

# Sample Outcome
sns.countplot(x='Outcome', data=df)
plt.title("Sample Outcome Distribution")
plt.show()

# 5. Feature Selection (Selecting the X and y Data)
X = df.drop(columns=['Outcome'])  # Features
```

```
y = df['Outcome']  # Target variable


# 6. Split the Dataset into Train and Test
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)


# 7. Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


# 8. Building the models
models = {
    'Logistic Regression': LogisticRegression(),
    'Decision Tree': DecisionTreeClassifier(),
    'XGBoost': XGBClassifier(),
    'SVM': SVC(),
    'KNN': KNeighborsClassifier(),
    'Naive Bayes': GaussianNB()
}


# 9. Voting Classifier (Decision Tree + XGBoost + KNN + ExtraTrees)
voting_clf = VotingClassifier(estimators=[
    ('dt', DecisionTreeClassifier()),
    ('xgb', XGBClassifier()),
    ('knn', KNeighborsClassifier()),
    ('et', ExtraTreesClassifier())
], voting='hard')
```

```
# 10. Stacking Classifier (BaggingClassifier with RF + DT with
LightGBM)
stacking_clf = StackingClassifier(
    estimators=[
        ('rf',
BaggingClassifier(base_estimator=RandomForestClassifier())),
        ('dt', DecisionTreeClassifier())
    ],
    final_estimator=XGBClassifier()
)

# 11. Training and Comparing Models
results = {}

for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    results[name] = {
        'accuracy': accuracy_score(y_test, y_pred),
        'precision': precision_score(y_test, y_pred),
        'recall': recall_score(y_test, y_pred),
        'f1score': f1_score(y_test, y_pred)
    }

# Evaluate Voting and Stacking Classifier
voting_clf.fit(X_train_scaled, y_train)
y_pred_voting = voting_clf.predict(X_test_scaled)
results['Voting Classifier'] = {
    'accuracy': accuracy_score(y_test, y_pred_voting),
    'precision': precision_score(y_test, y_pred_voting),
    'recall': recall_score(y_test, y_pred_voting),
    'f1score': f1_score(y_test, y_pred_voting)
}

stacking_clf.fit(X_train_scaled, y_train)
y_pred_stacking = stacking_clf.predict(X_test_scaled)
results['Stacking Classifier'] = {
    'accuracy': accuracy_score(y_test, y_pred_stacking),
    'precision': precision_score(y_test, y_pred_stacking),
    'recall': recall_score(y_test, y_pred_stacking),
    'f1score': f1_score(y_test, y_pred_stacking)
}

# 12. Compare Models (Graph)
results_df = pd.DataFrame(results).T
results_df.plot(kind='bar', figsize=(10, 6))
plt.title("Comparison of Models")
plt.ylabel("Score")
plt.xticks(rotation=45)
plt.show()

# 13. Save the Best Model (Stacking Classifier)
import joblib
joblib.dump(stacking_clf, 'stacking_classifier_model.pkl')

# 14. Frontend Development with Flask (basic setup)
```

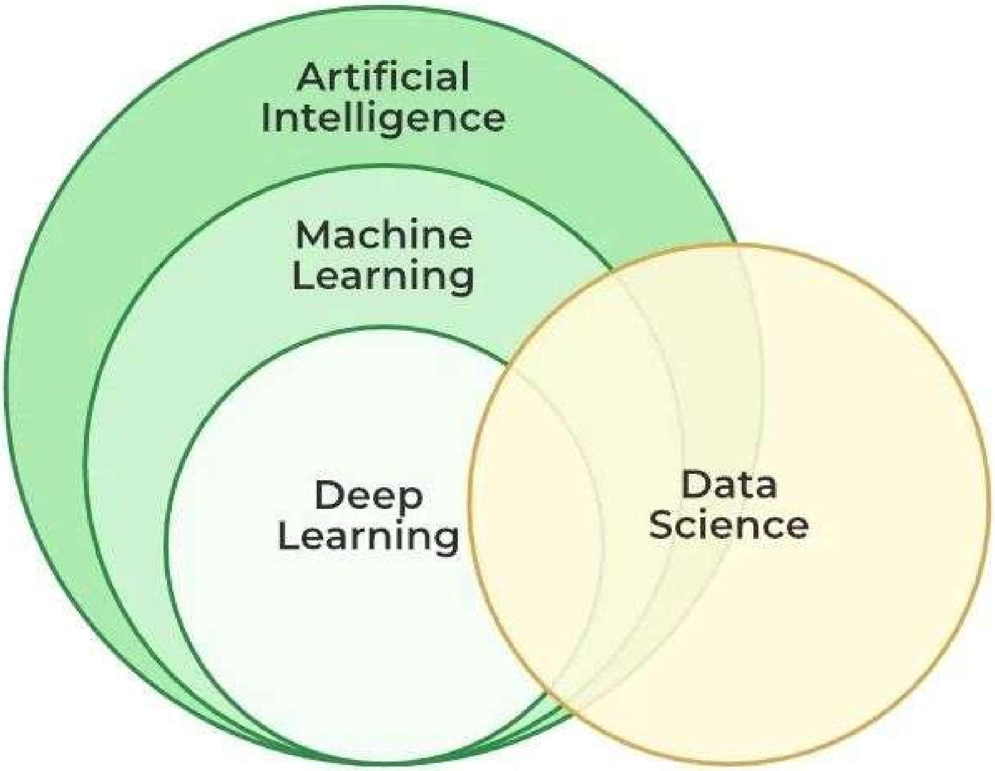
```
app = Flask(__name__)


# 15. User input for prediction
@app.route('/')
def home():
    return render_template('index.html')


@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        features = [float(x) for x in request.form.values()]
        features = np.array(features).reshape(1, -1)
        features_scaled = scaler.transform(features)


        model = joblib.load('stacking_classifier_model.pkl')
        prediction = model.predict(features_scaled)


        return render_template('index.html',
prediction_text=f'Predicted Outcome: {prediction[0]}')


# 16. User Authentication and SQLite setup
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']


        conn = sqlite3.connect('user_data.db')
        c = conn.cursor()
        c.execute('CREATE TABLE IF NOT EXISTS users (username TEXT,
password TEXT)')
        c.execute('INSERT INTO users (username, password) VALUES (?,
?)', (username, password))
        conn.commit()
        conn.close()

        return render_template('index.html', prediction_text='User
Registered Successfully!')
    return render_template('register.html')

if __name___== '__main__':
    app.run(debug=True)
```

7. SOFTWARE ENVIRONMENT

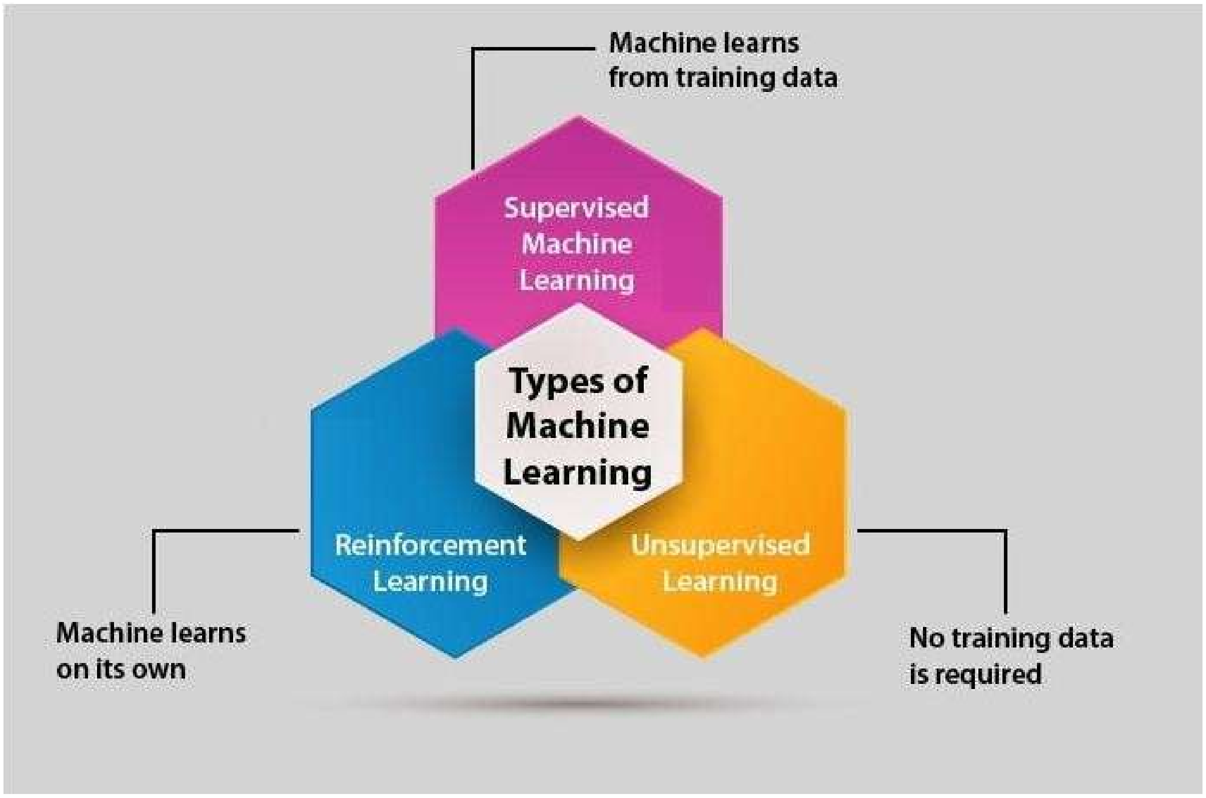
Machine Learning:

What is Machine Learning?

Machine learning is a branch of artificial intelligence that enables algorithms to uncover hidden patterns within datasets, allowing them to make predictions on new, similar data without explicit programming for each task. Traditional machine learning combines data with statistical tools to predict outputs, yielding actionable insights. This technology finds applications in diverse fields such as image and speech recognition, natural language processing, recommendation systems, fraud detection, portfolio optimization, and automating tasks.



Types of Machine Learning:


Machine learns from training data
Supervised Machine Learning
Types of Machine Learning
Reinforcement Learning
Unsupervised Learning
Machine learns on its own
No training data is required

1. Supervised Machine Learning:

Supervised learning is a type of machine learning in which the algorithm is trained on the labeled dataset. It learns to map input features to targets based on labeled training data. In supervised learning, the algorithm is provided with input features and corresponding output labels, and it learns to generalize from this data to make predictions on new, unseen data.

There are two main types of supervised learning:

Regression: Regression is a type of supervised learning where the algorithm learns to predict continuous values based on input features. The output labels in regression are continuous values, such as stock prices, and housing prices. The different regression algorithms in machine learning are: Linear Regression, Polynomial

Regression, Ridge Regression, Decision Tree Regression, Random Forest Regression, Support Vector Regression, etc

Classification: Classification is a type of supervised learning where the algorithm learns to assign input data to a specific category or class based on input features. The output labels in classification are discrete values. Classification algorithms can be binary, where the output is one of two possible classes, or multiclass, where the output can be one of several classes. The different Classification algorithms in machine learning are: Logistic Regression, Naive Bayes, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), etc

2. Unsupervised Machine Learning:

Unsupervised learning is a type of machine learning where the algorithm learns to recognize patterns in data without being explicitly trained using labeled examples. The goal of unsupervised learning is to discover the underlying structure or distribution in the data.

There are two main types of unsupervised learning:

Clustering: Clustering algorithms group similar data points together based on their characteristics. The goal is to identify groups, or clusters, of data points that are similar to each other, while being distinct from other groups. Some popular clustering algorithms include K-means, Hierarchical clustering, and DBSCAN.

Dimensionality reduction: Dimensionality reduction algorithms reduce the number of input variables in a dataset while preserving as much of the original information as possible. This is useful for reducing the complexity of a dataset and making it easier to visualize and analyze. Some popular dimensionality reduction algorithms include Principal Component Analysis (PCA), t-SNE, and Auto encoders.

3. Reinforcement Machine Learning:

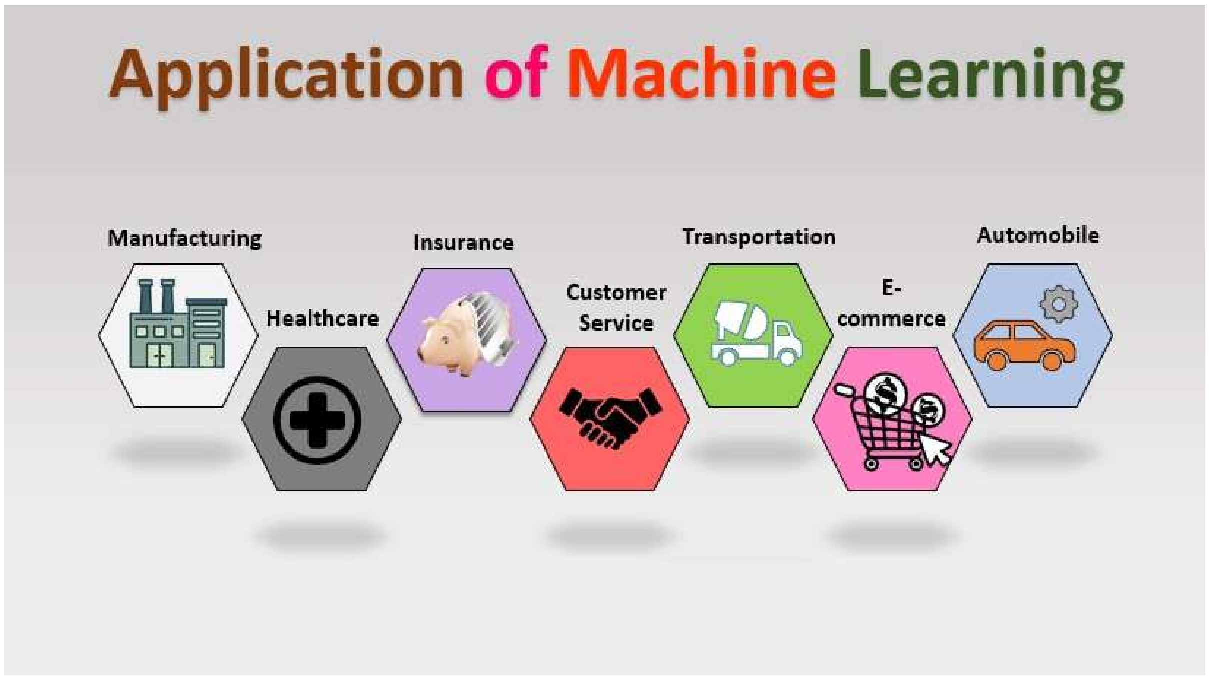
Reinforcement learning is a type of machine learning where an agent learns to interact with an environment by performing actions and receiving rewards or penalties based on its actions. The goal of reinforcement learning is to learn a policy, which is a mapping from states to actions, that maximizes the expected cumulative reward over time.

There are two main types of reinforcement learning:

Model-based reinforcement learning: In model-based reinforcement learning, the agent learns a model of the environment, including the transition probabilities between states and the rewards associated with each state-action pair. The agent then uses this model to plan its actions in order to maximize its expected reward. Some popular model-based reinforcement learning algorithms include Value Iteration and Policy Iteration.

Model-free reinforcement learning: In model-free reinforcement learning, the agent learns a policy directly from experience without explicitly building a model of the environment. The agent interacts with the environment and updates its policy based on the rewards it receives. Some popular model-free reinforcement learning algorithms include Q-Learning, SARSA, and Deep Reinforcement Learning.

Applications of Machine Learning:



1. Automation: Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots perform the essential process steps in manufacturing plants.
2. Finance Industry: Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.
3. Government organization: The government makes use of ML to manage public safety and utilities. Take the example of China with its massive face recognition. The government uses Artificial intelligence to prevent jaywalking.
4. Healthcare industry: Healthcare was one of the first industries to use machine learning with image detection.
5. Marketing: Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical

tools like Bayesian analysis to estimate the value of a customer. With the boom of data, the marketing department relies on AI to optimize customer relationships and marketing campaigns.

6. Retail industry: Machine learning is used in the retail industry to analyze customer behavior, predict demand, and manage inventory. It also helps retailers to personalize the shopping experience for each customer by recommending products based on their past purchases and preferences.
7. Transportation: Machine learning is used in the transportation industry to optimize routes, reduce fuel consumption, and improve the overall efficiency of transportation systems. It also plays a role in autonomous vehicles, where ML algorithms are used to make decisions about navigation and safety.

Advantages of Machine Learning:

1. Improved Accuracy and Precision: One of the most significant benefits of machine learning is its ability to improve accuracy and precision in various tasks. ML models can process vast amounts of data and identify patterns that might be overlooked by humans. For instance, in medical diagnostics, ML algorithms can analyze medical images or patient data to detect diseases with a high degree of accuracy.

2. Automation of Repetitive Tasks: Machine learning enables the automation of repetitive and mundane tasks, freeing up human resources for more complex and creative endeavors. In industries like manufacturing and customer service, ML-driven automation can handle routine tasks such as quality control, data entry, and customer inquiries, resulting in increased productivity and efficiency.

3. Enhanced Decision-Making: ML models can analyze large datasets and provide insights that aid in decision-making. By identifying trends, correlations, and

anomalies, machine learning helps businesses and organizations make data-driven decisions. This is particularly valuable in sectors like finance, where ML can be used for risk assessment, fraud detection, and investment strategies.

4. Personalization and Customer Experience: Machine learning enables the personalization of products and services, enhancing customer experience. In e-commerce, ML algorithms analyze customer behavior and preferences to recommend products tailored to individual needs. Similarly, streaming services use ML to suggest content based on user viewing history, improving user engagement and satisfaction.

5. Predictive Analytics: Predictive analytics is a powerful application of machine learning that helps forecast future events based on historical data. Businesses use predictive models to anticipate customer demand, optimize inventory, and improve supply chain management. In healthcare, predictive analytics can identify potential outbreaks of diseases and help in preventive measures.

Disadvantages of Machine Learning:

1. Data Dependency: Machine learning models require vast amounts of data to train effectively. The quality, quantity, and diversity of the data significantly impact the model's performance. Insufficient or biased data can lead to inaccurate predictions and poor decision-making. Additionally, obtaining and curating large datasets can be time-consuming and costly.

2. High Computational Costs: Training ML models, especially deep learning algorithms, demands significant computational resources. High-performance hardware such as GPUs and TPUs are often required, which can be expensive. The

energy consumption associated with training large models is also substantial, raising concerns about the environmental impact.

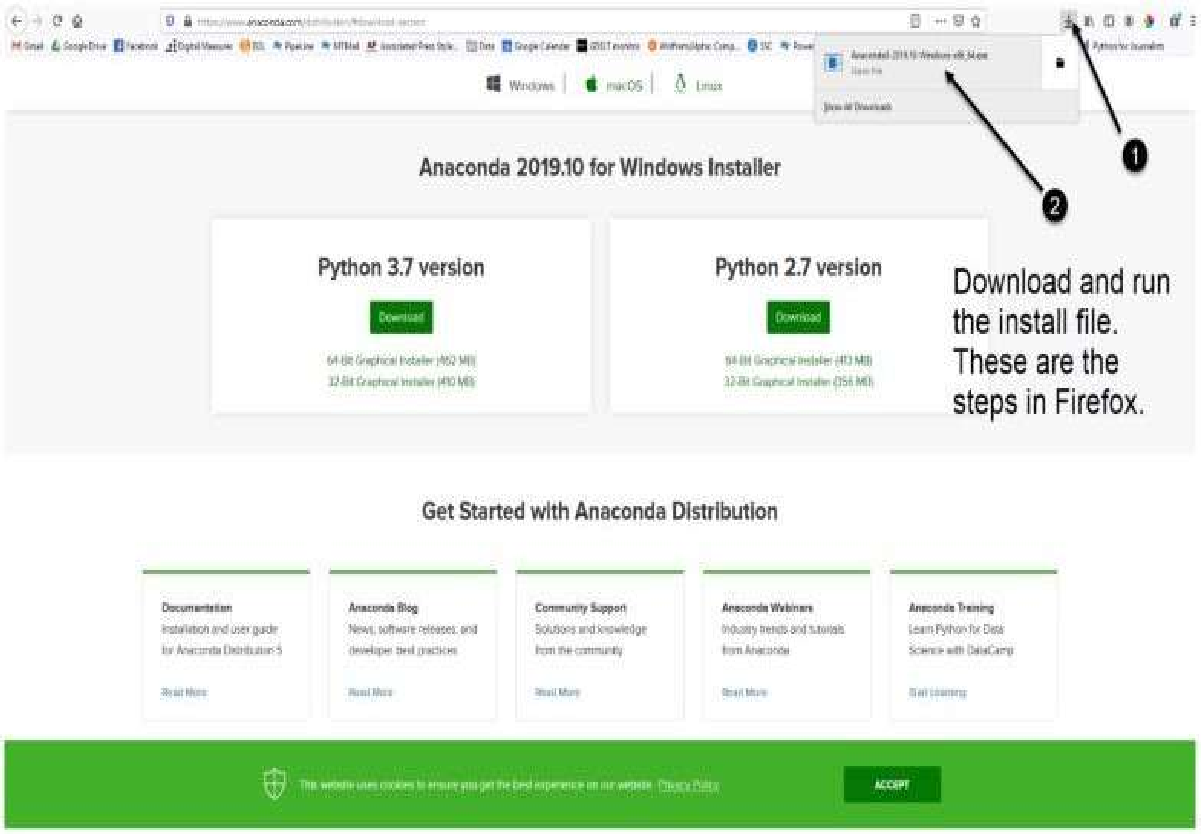
3. Complexity and Interpretability: Many machine learning models, particularly deep neural networks, function as black boxes. Their complexity makes it difficult to interpret how they arrive at specific decisions. This lack of transparency poses challenges in fields where understanding the decision-making process is critical, such as healthcare and finance.

4. Overfitting and Underfitting: Machine learning models can suffer from overfitting or underfitting. Overfitting occurs when a model learns the training data too well, capturing noise and anomalies, which reduces its generalization ability to new data. Underfitting happens when a model is too simple to capture the underlying patterns in the data, leading to poor performance on both training and test data.

5. Ethical Concerns: ML applications can raise ethical issues, particularly concerning privacy and bias. Data privacy is a significant concern, as ML models often require access to sensitive and personal information. Bias in training data can lead to biased models, perpetuating existing inequalities and unfair treatment of certain groups.

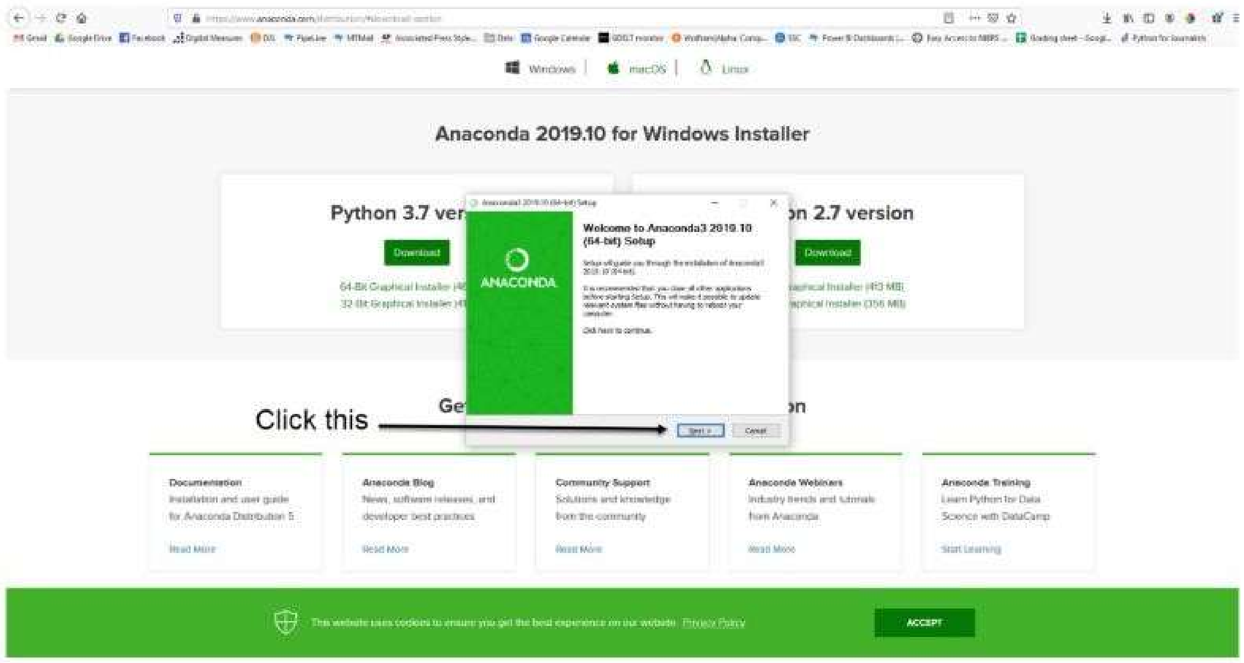
Python Anaconda Installation

1. Go to this link and download Anaconda for Windows, Mac, or Linux: – Download anaconda

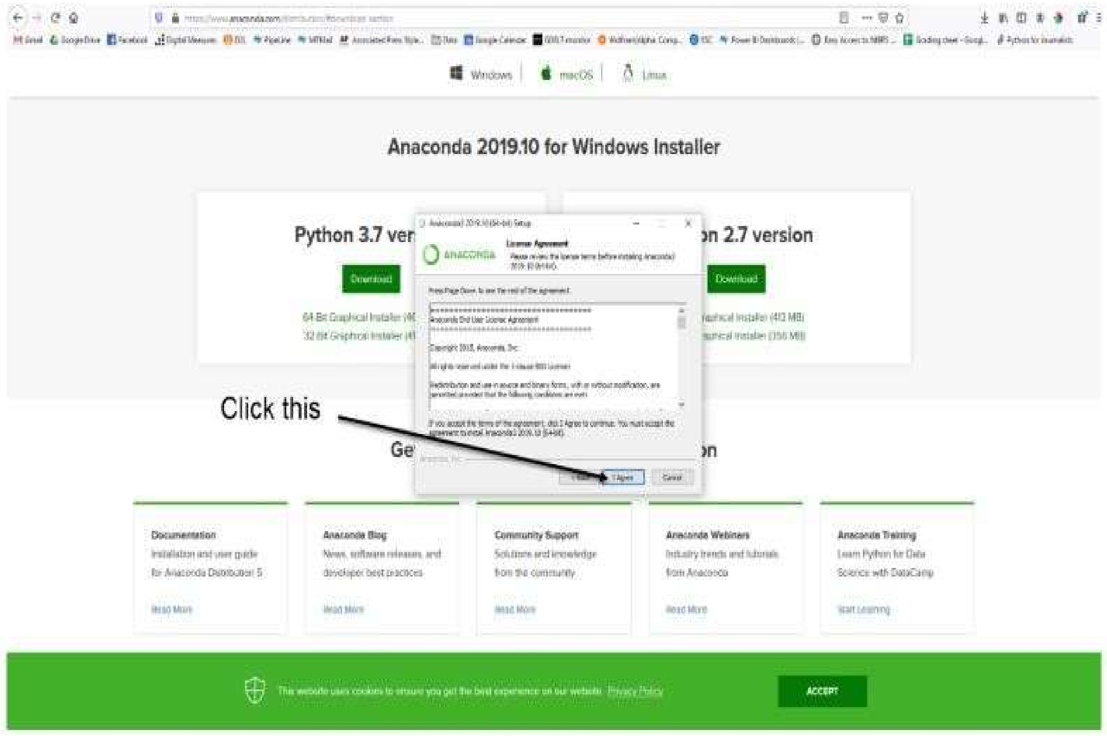

Windows
macOS
Linux
Anaconda 2019.10 for Windows Installer
Python 3.7 version
Download
64-Bit Graphical Installer (462 MB)
32-Bit Graphical Installer (410 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)
Download and run the install file. These are the steps in Firefox.
1
2
Get Started with Anaconda Distribution
Documentation
Anaconda Blog
Community Support
Anaconda Webinars
Anaconda Training
ACCEPT

You can download the installer for Python 3.7 or for Python 2.7 (at the time of writing). And you can download it for a 32-bit or 64-bit machine.

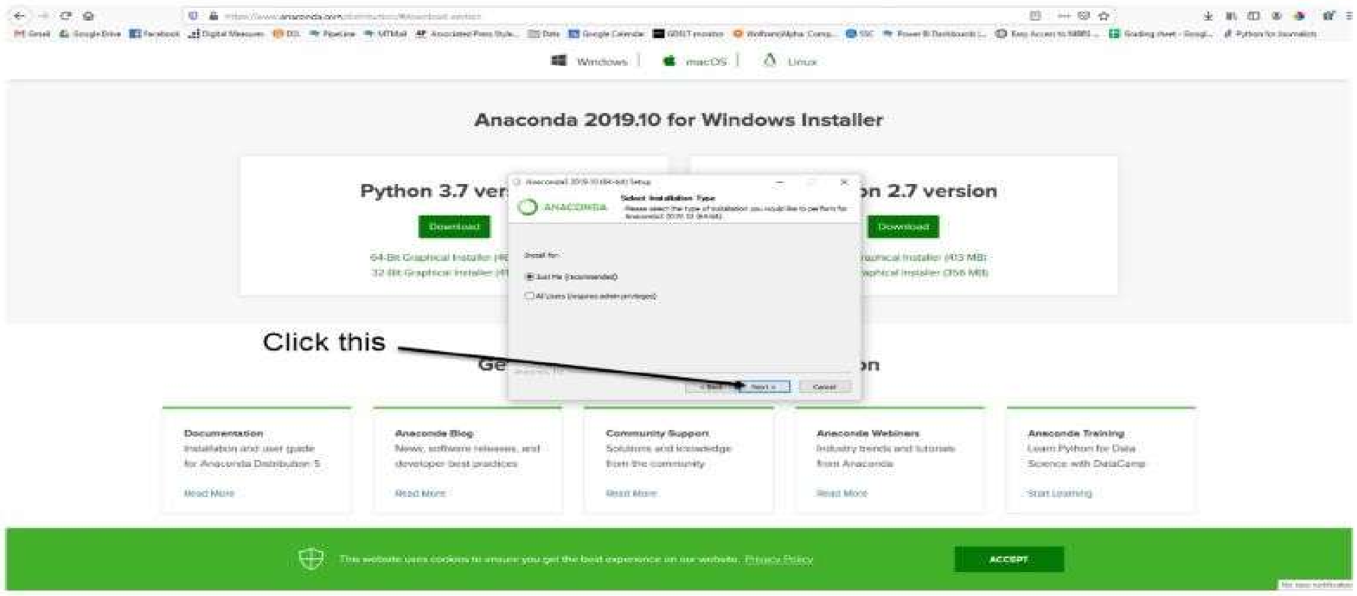
2. Click on the downloaded .exe to open it. This is the Anaconda setup. Click next.



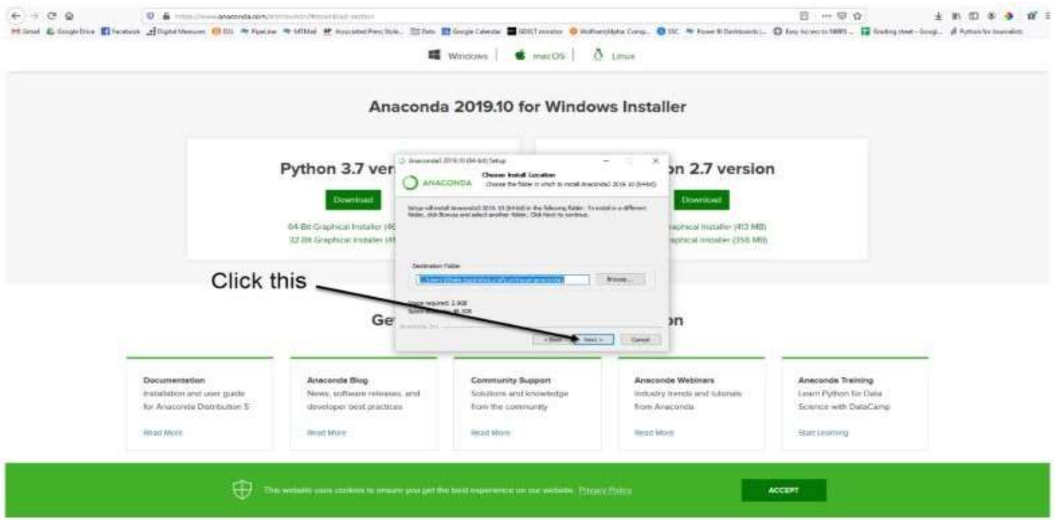
3. Now, you’ll see the license agreement. Click on ‘I Agree’.



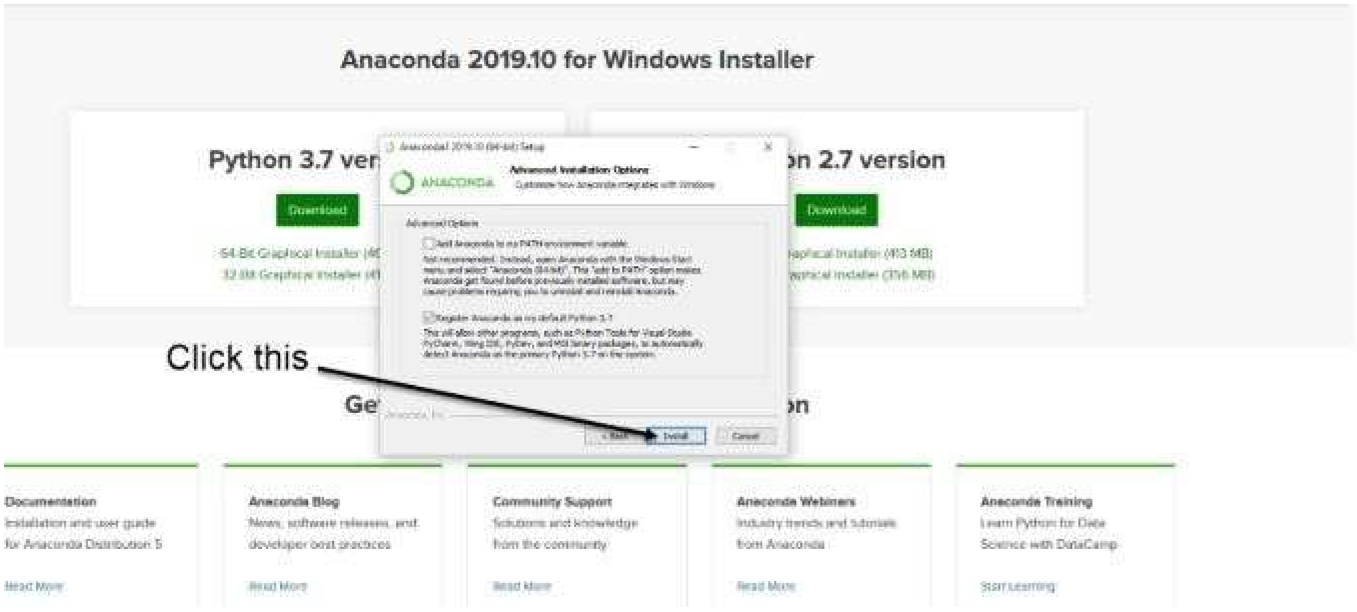
4. You can install it for all users or just for yourself. If you want to install it for all users, you need administrator privileges.



5. Choose where you want to install it. Here, you can see the available space and how much you need.



6. Now, you'll get some advanced options. You can add Anaconda to your system's PATH environment variable, and register it as the primary system Python 3.7. If you add it to PATH, it will be found before any other installation. Click on 'Install'.



Anaconda 2019.10 for Windows Installer

Anaconda3 2019.10 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

☐ Add Anaconda to my PATH environment variable

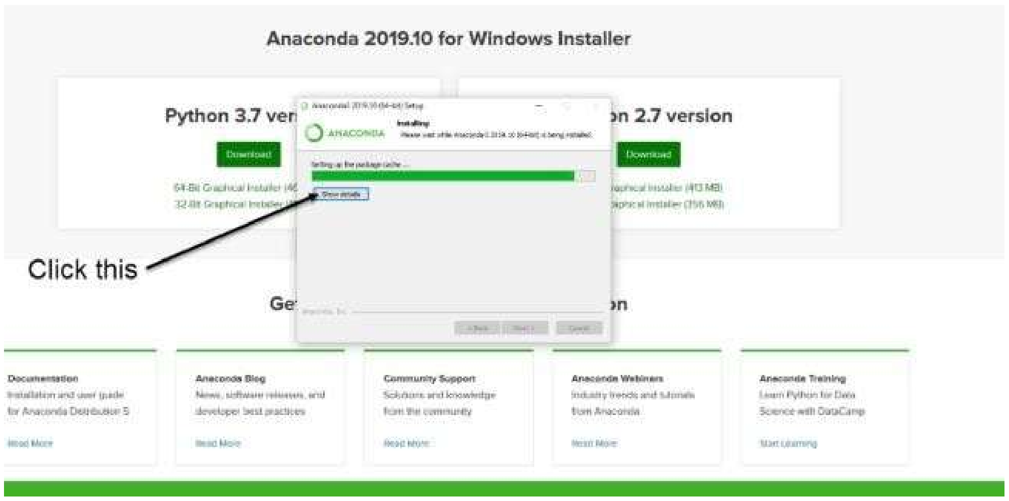
Not recommended. Instead, open Anaconda with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda as my default Python 3.7

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.7 on the system.

Python 3.7 ver
Download
64-Bit Graphical Installer (46
32-Bit Graphical Installer (4
on 2.7 version
Download
aphical Installer (413 MB)
aphical Installer (356 MB)
Click this
Ge
on
Anaconda, Inc.
< Back
Install
Cancel
Documentation
Installation and user guide for Anaconda Distribution 5
Read More
Anaconda Blog
News, software releases, and developer best practices
Read More
Community Support
Solutions and knowledge from the community
Read More
Anaconda Webinars
Industry trends and tutorials from Anaconda
Read More
Anaconda Training
Learn Python for Data Science with DataCamp
Start Learning

7. It will unpack some packages and extract some files on your machine. This will take a few minutes.



Anaconda 2019.10 for Windows Installer

Anaconda3 2019.10 (64-bit) Setup

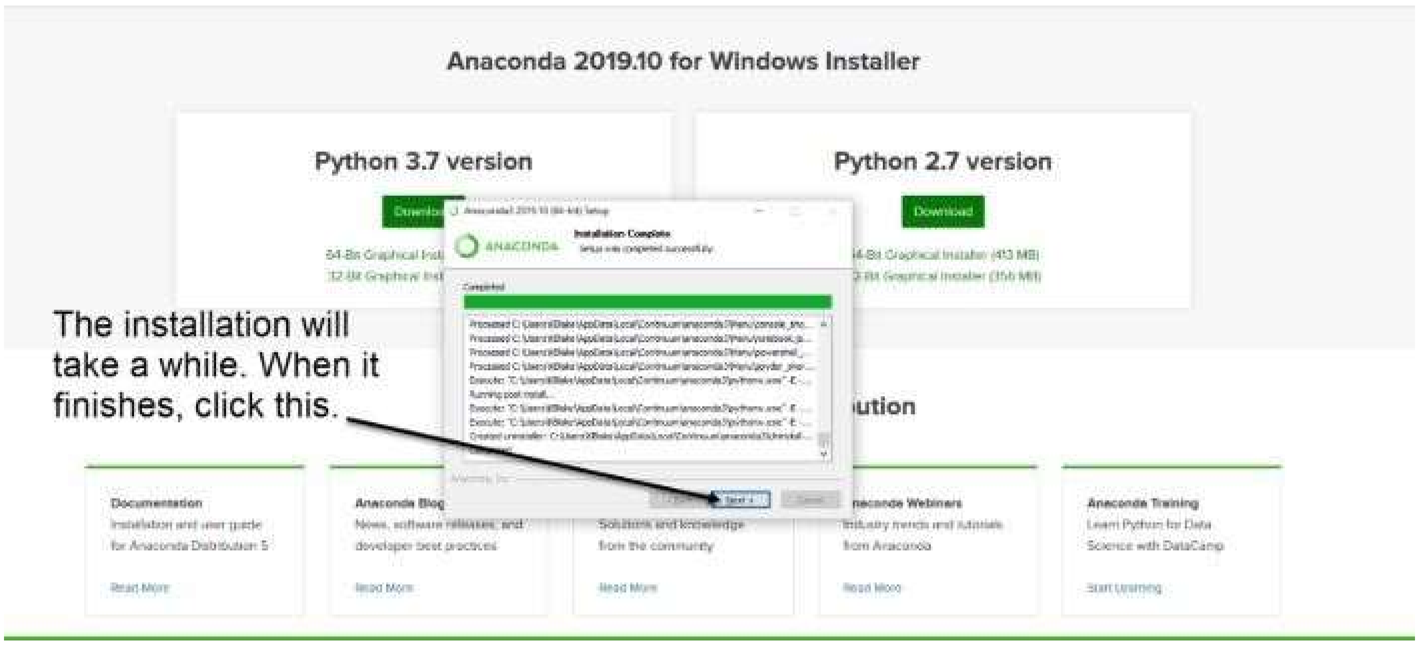
Installing

Please wait while Anaconda3 2019.10 (64-bit) is being installed.

Setting up the package cache ...

Python 3.7 ver
Download
64-Bit Graphical Installer (46
32-Bit Graphical Installer (4
on 2.7 version
Download
aphical Installer (413 MB)
aphical Installer (356 MB)
Show details
Click this
Ge
on
Anaconda, Inc.
< Back
Next >
Cancel
Documentation
Installation and user guide for Anaconda Distribution 5
Read More
Anaconda Blog
News, software releases, and developer best practices
Read More
Community Support
Solutions and knowledge from the community
Read More
Anaconda Webinars
Industry trends and tutorials from Anaconda
Read More
Anaconda Training
Learn Python for Data Science with DataCamp
Start Learning

8. The installation is complete. Click Next.



Anaconda 2019.10 for Windows Installer

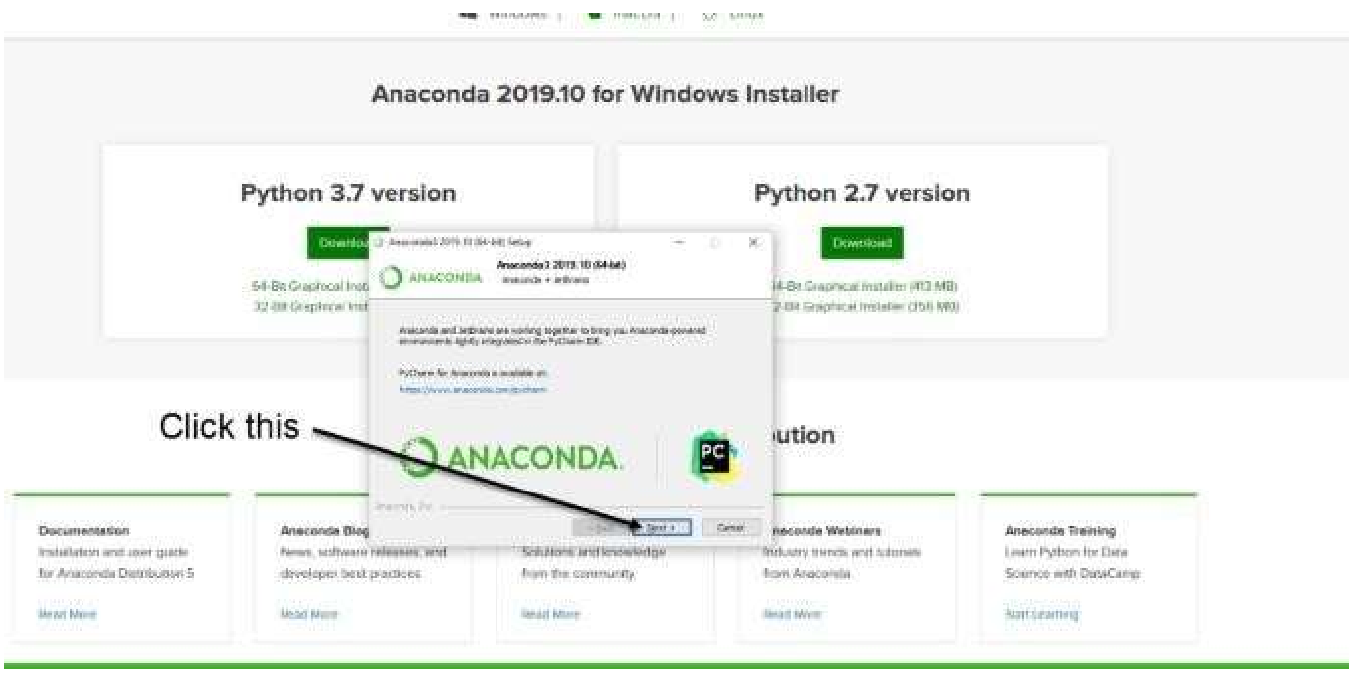
Installation Complete

Setup was completed successfully.

The installation will take a while. When it finishes, click this.

Python 3.7 version
Python 2.7 version
Download
64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)
Anaconda3 2019.10 (64-bit) Setup
ANACONDA
Completed
Next >
Cancel
Documentation
Installation and user guide for Anaconda Distribution 5
Read More
Anaconda Blog
News, software releases, and developer best practices
Solutions and knowledge from the community
Anaconda Webinars
Industry trends and tutorials from Anaconda
Anaconda Training
Learn Python for Data Science with DataCamp
Start Learning

9. This screen will inform you about PyCharm. Click Next.



Anaconda 2019.10 for Windows Installer

Anaconda3 2019.10 (64-bit)

Anaconda + JetBrains

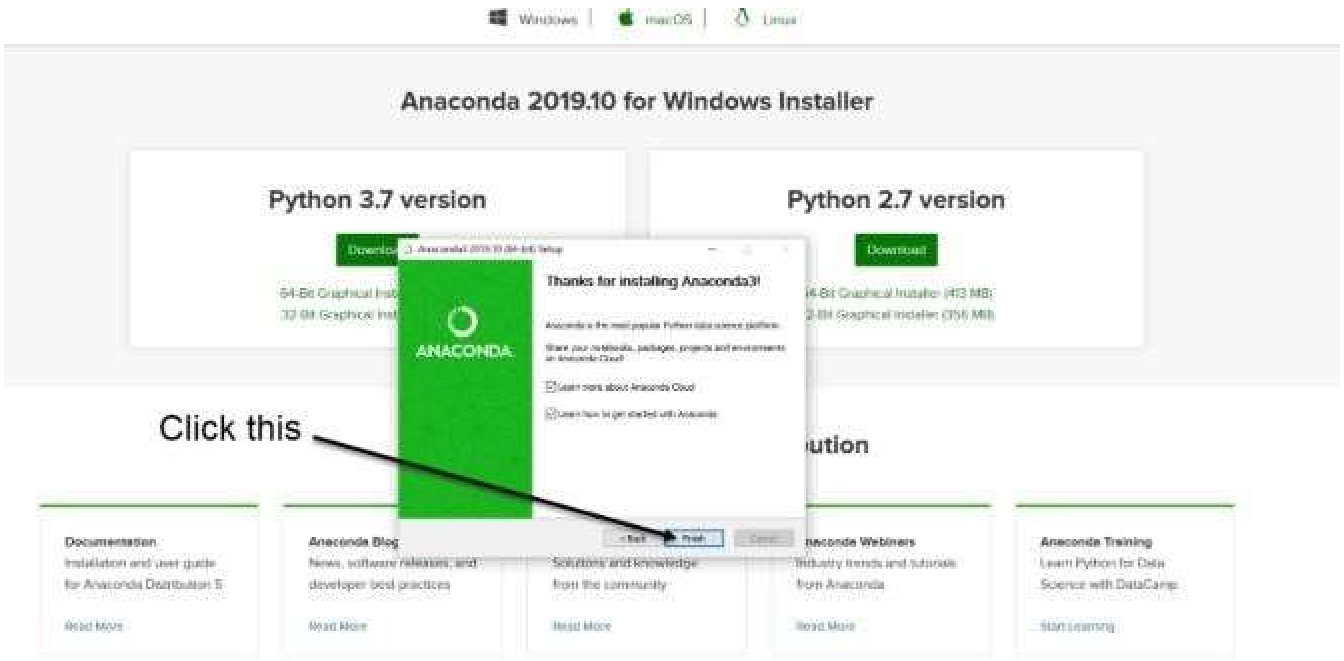
Anaconda and JetBrains are working together to bring you Anaconda-powered environments tightly integrated in the PyCharm IDE.

PyCharm for Anaconda is available at:
https://www.anaconda.com/pycharm

Click this

Windows
macOS
Linux
Python 3.7 version
Python 2.7 version
Download
64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)
Anaconda3 2019.10 (64-bit) Setup
ANACONDA
PC
Next >
Cancel
Documentation
Installation and user guide for Anaconda Distribution 5
Read More
Anaconda Blog
News, software releases, and developer best practices
Solutions and knowledge from the community
Anaconda Webinars
Industry trends and tutorials from Anaconda
Anaconda Training
Learn Python for Data Science with DataCamp
Start Learning

10. The installation is complete. You can choose to get more information about Anaconda cloud and how to get started with Anaconda. Click Finish.


Windows | macOS | Linux
Anaconda 2019.10 for Windows Installer
Python 3.7 version
Python 2.7 version
Download
64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)
ANACONDA
Click this
Documentation
Installation and user guide for Anaconda Distribution 5
Read More
Anaconda Blog
News, software releases, and developer best practices
Solutions and knowledge from the community
Anaconda Webinars
Industry trends and tutorials from Anaconda
Anaconda Training
Learn Python for Data Science with DataCamp
Start Learning
< Back
Finish
Cancel

Thanks for installing Anaconda3!

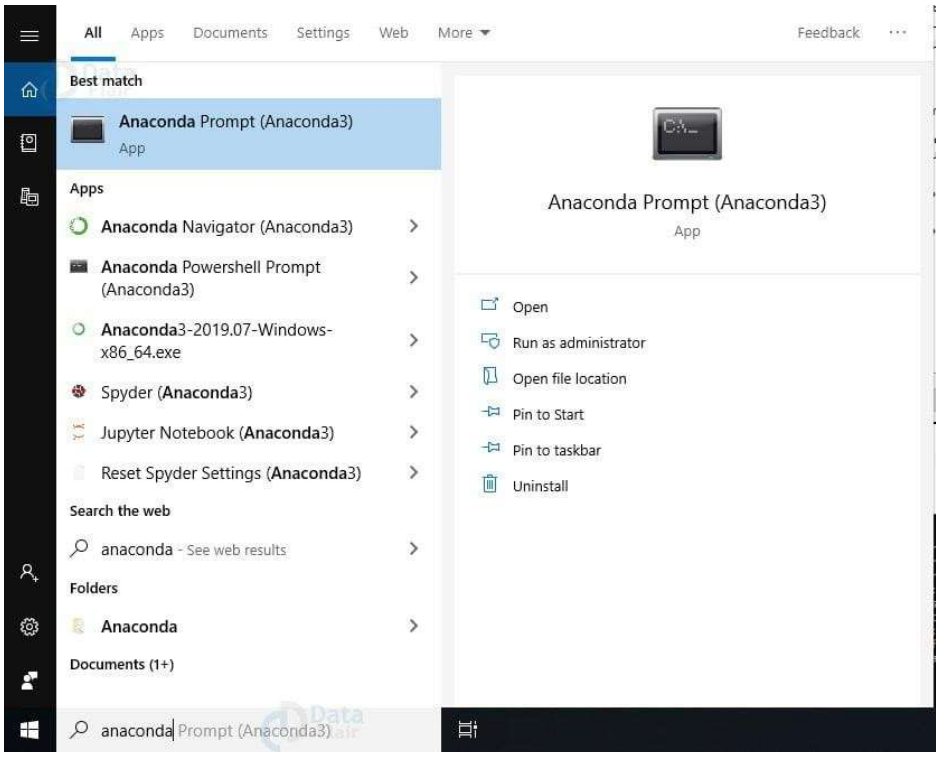
Anaconda is the most popular Python data science platform.

Share your notebooks, packages, projects and environments on Anaconda Cloud!

☑ Learn more about Anaconda Cloud

☑ Learn how to get started with Anaconda

11. If you search for Anaconda now, you will see the following options:


All
Apps
Documents
Settings
Web
More
Feedback
Best match
Anaconda Prompt (Anaconda3)
App
Apps
Anaconda Navigator (Anaconda3)
Anaconda Powershell Prompt (Anaconda3)
Anaconda3-2019.07-Windows-x86_64.exe
Spyder (Anaconda3)
Jupyter Notebook (Anaconda3)
Reset Spyder Settings (Anaconda3)
Search the web
anaconda - See web results
Folders
Anaconda
Documents (1+)
Anaconda Prompt (Anaconda3)
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall
anaconda Prompt (Anaconda3)

LIBRARIES/PACKGES :-

1. NumPy:

Provides support for large, multi-dimensional arrays and matrices. Includes a vast collection of mathematical functions to operate on these arrays.

2. Pandas:

Offers data structures like DataFrames for efficient data manipulation and analysis. Simplifies data cleaning, preparation, and munging tasks for data science projects.

3. SciPy:

Builds on NumPy by adding a collection of algorithms for scientific and technical computing.Covers areas such as integration, optimization, interpolation, eigenvalue problems, and others.

4. Matplotlib:

Enables the creation of static, animated, and interactive visualizations in Python. Commonly used for plotting graphs and charts, making it a staple for data visualization.

5. Scikit-learn:

Provides simple and efficient tools for data mining and data analysis. Supports various machine learning algorithms, including classification, regression, and clustering.

6. TensorFlow:

An open-source library for numerical computation and large-scale machine learning. Used for building and deploying machine learning models, especially deep neural networks.

7. Keras:

A high-level neural networks API, written in Python and capable of running on top of TensorFlow. Facilitates fast experimentation with deep neural networks due to its user-friendly, modular nature.

8. SYSTEM TESTING

System testing, also called black box testing, focuses on the external parts of the system and tests and validates the fully integrated software system. Computer systems are typically made with software integration, although the software is just a single element of any computer system. Software is usually developed in units and interfaced with other software elements and hardware, creating the complete computer system. So, computer systems consist of a set of different software applications, each designed to perform a different task, but only if they are interfaced with compatible hardware. System testing is a series of tests that examine the entire workings and behavior of the integrated software computer system against the end user's needs and requirements.

System Testing Levels

Four levels of software testing make up the hierarchy of the testing process. They are:

Unit testing: Unit testing tests a single software application.

Integration testing: Integration testing tests a group of software units.

System testing: System testing tests the entire system and is our focus for today.

Acceptance testing: Acceptance testing tests the business requirements' acceptability.

So, we see that system testing occurs after integration testing but before the acceptance testing. The testing process evaluates the end-to-end system specifications.

8.1 Different Types of System Testing in Software Testing

Although there are over 50 different types of system testing, the following seven types are the most often used.

Usability Testing: As the name implies, usability testing chiefly focuses on how easy it is for the end user to use the application, flexibility in handling controls, and the system's ability to achieve its objectives.

Load Testing: Load testing shows how a software solution will perform under real-life loads.

Regression Testing: Regression testing ensures that no changes made during development have created new bugs. Additionally, regression testing ensures no old bugs resurface after adding software modules.

Recovery Testing: Software is prone to crashing, so recovery testing is done to ascertain whether a software solution is trustworthy and can successfully recover from possible crashes.

Migration Testing: Migration testing ensures that the software can be shifted from an older system infrastructure to the current one without spawning any issues.

Functional Testing: Also called functional completeness testing, functional testing involves brainstorming any possible missing functions. For example, testers might create a list of additional functionalities to be incorporated into the application to improve it during functional testing.

Hardware/Software Testing: IBM calls Hardware/Software testing "HW/SW Testing." This testing involves focusing attention on the interactions between the hardware and software.

System Testing Process

There are two kinds of software testing: Black Box and White Box.White box testing tests a software application’s internal workings or code. On the other hand, black box (or system testing) does the opposite, focusing on externals. System testing deals with the software’s external workings from the user’s perspective.

The system testing process is broken down into the following steps:

Test Planning: First, there needs to be a plan. The test plan is a document with all the testing information, such as strategies, objectives, entry-exit criteria, testing tools, software requirements, guidelines, etc.

Test Case Design and Execution: Next, the testers create a test case for every feature, including the test scenarios, process, description, etc. The testers then perform and execute the tests.

Defect Tracking and Defect Management: The testers then track and record any defects using the defect-tracking tools like Bugzilla, JIRA, Trello, etc. Also, the testers must document the defects so the developers can deal with and resolve them.

Reporting and Communication: Finally, the testers create bug and defect reports and send them to the developers.

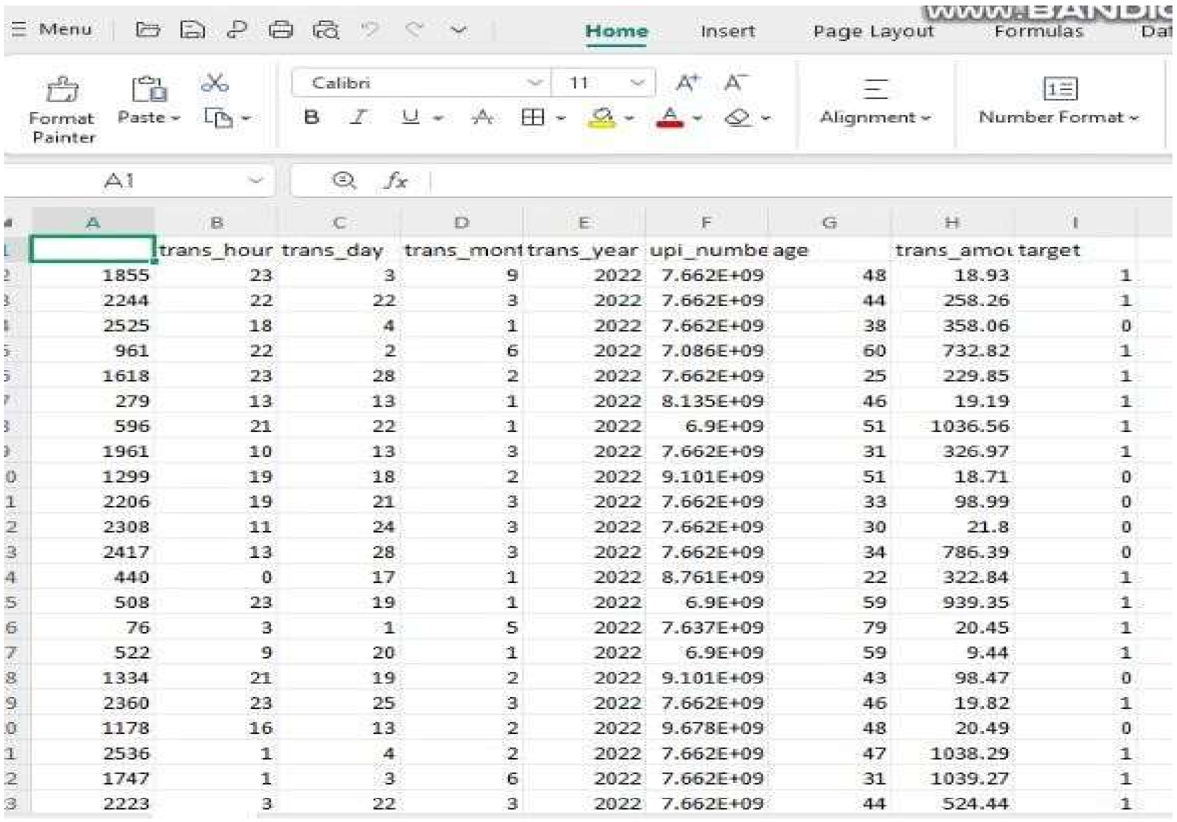
8.2 TEST CASES:

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

9. SCREENSHOTS

Tables:

DATASET



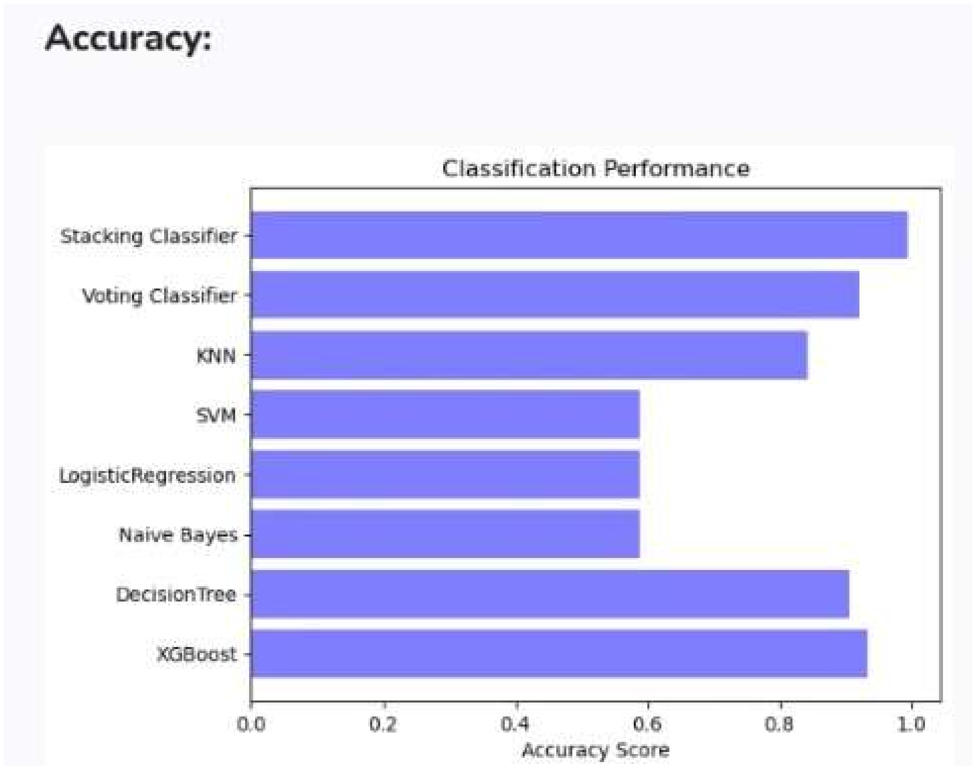
	trans_hour	trans_day	trans_mont	trans_year	upi_numbe	age	trans_amou	target
1855	23	3	9	2022	7.662E+09	48	18.93	1
2244	22	22	3	2022	7.662E+09	44	258.26	1
2525	18	4	1	2022	7.662E+09	38	358.06	0
961	22	2	6	2022	7.086E+09	60	732.82	1
1618	23	28	2	2022	7.662E+09	25	229.85	1
279	13	13	1	2022	8.135E+09	46	19.19	1
596	21	22	1	2022	6.9E+09	51	1036.56	1
1961	10	13	3	2022	7.662E+09	31	326.97	1
1299	19	18	2	2022	9.101E+09	51	18.71	0
2206	19	21	3	2022	7.662E+09	33	98.99	0
2308	11	24	3	2022	7.662E+09	30	21.8	0
2417	13	28	3	2022	7.662E+09	34	786.39	0
440	0	17	1	2022	8.761E+09	22	322.84	1
508	23	19	1	2022	6.9E+09	59	939.35	1
76	3	1	5	2022	7.637E+09	79	20.45	1
522	9	20	1	2022	6.9E+09	59	9.44	1
1334	21	19	2	2022	9.101E+09	43	98.47	0
2360	23	25	3	2022	7.662E+09	46	19.82	1
1178	16	13	2	2022	9.678E+09	48	20.49	0
2536	1	4	2	2022	7.662E+09	47	1038.29	1
1747	1	3	6	2022	7.662E+09	31	1039.27	1
2223	3	22	3	2022	7.662E+09	44	524.44	1

Performance Evaluation – Multi Class

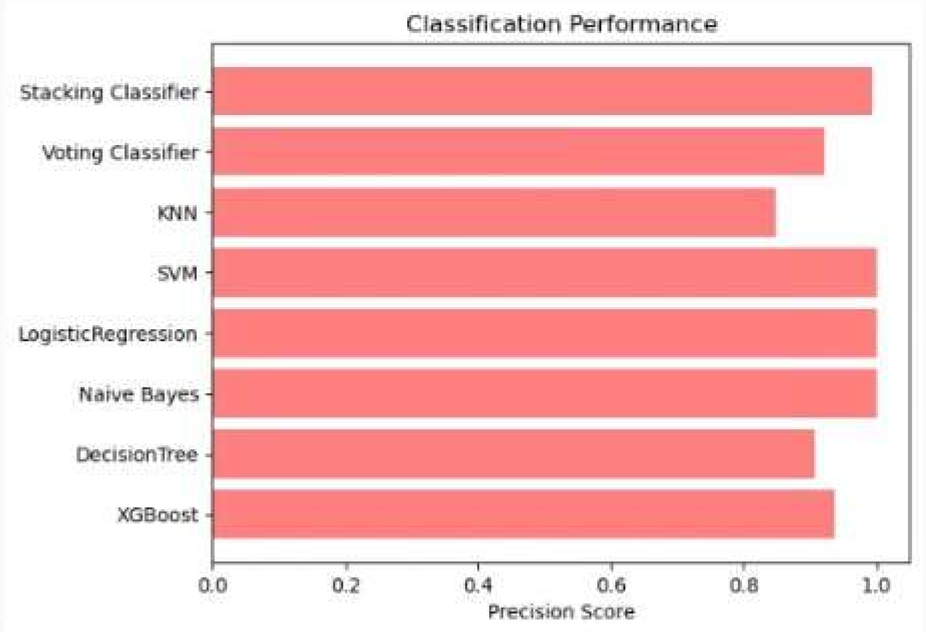
Algorithm Name	Accuracy	Precision	Recall	FSCORE
XG Boost	0.934	0.936	0.934	0.935
Decision Tree	0.906	0.907	0.906	0.907
Navie Bayes	0.588	1.000	0.588	0.741
Logistic Regression	0.588	1.000	0.588	0.741

SVM	**0.588**	**1.000**	**0.588**	**0.741**
KNN	**0.843**	**0.847**	**0.843**	**0.844**
Voting Classifier	**0.921**	**0.922**	**0.921**	**0.922**
Stacking Classifier	**0.994**	**0.994**	**0.994**	**0.994**

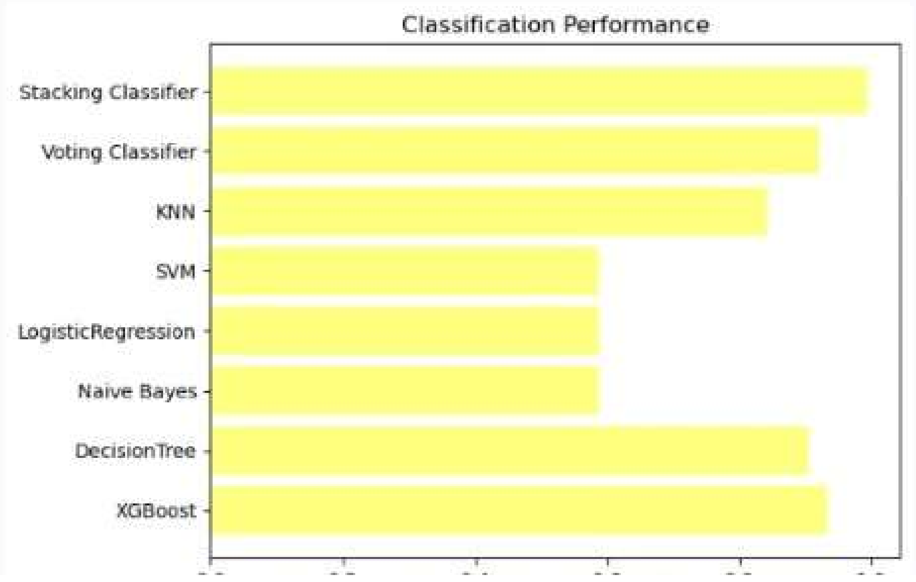
Comparison Graphs

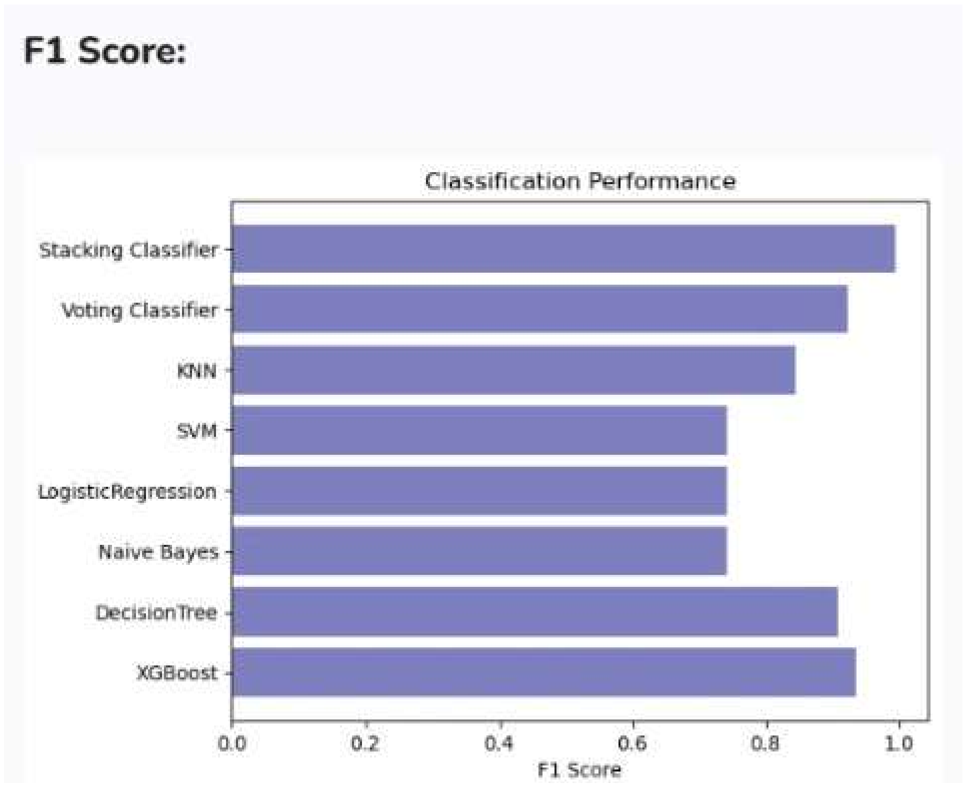


Precision:



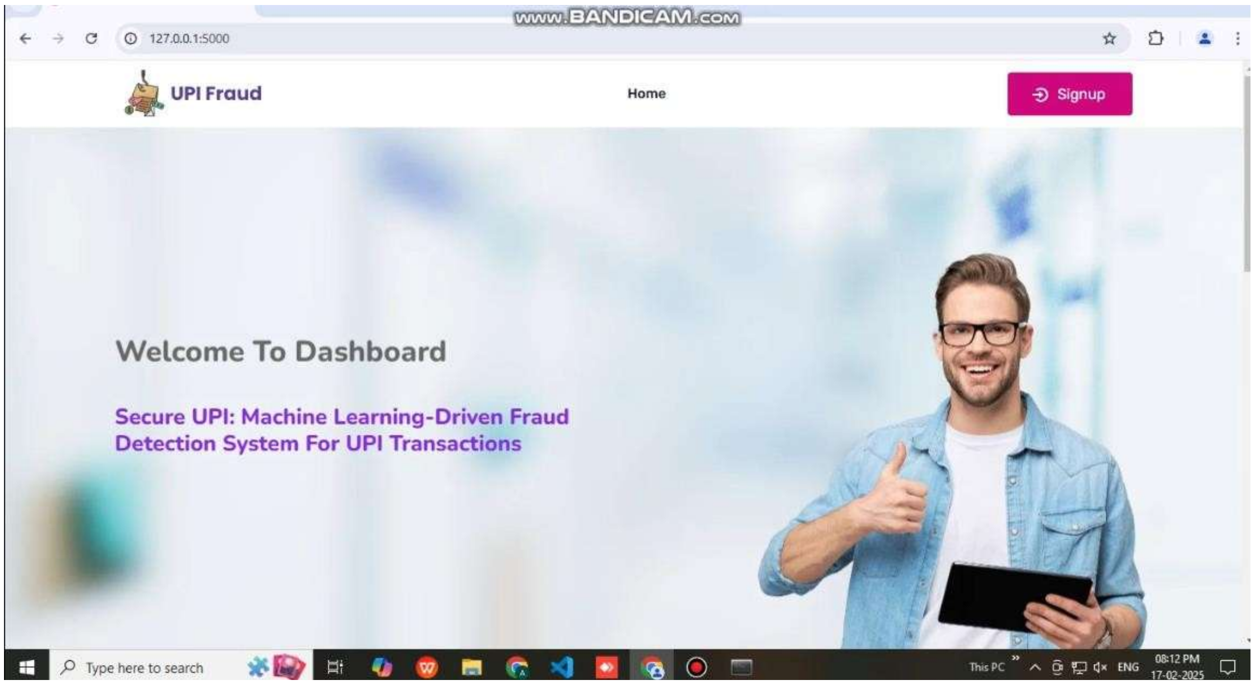
Recall Score:



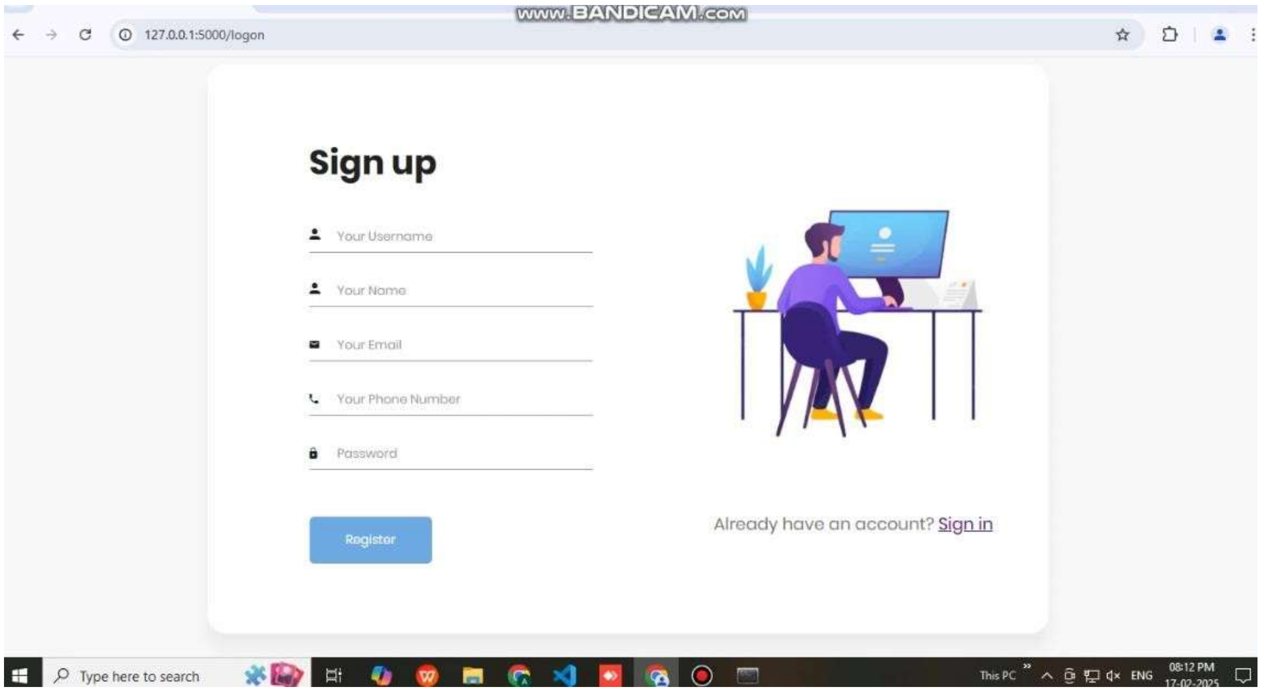


User Interface

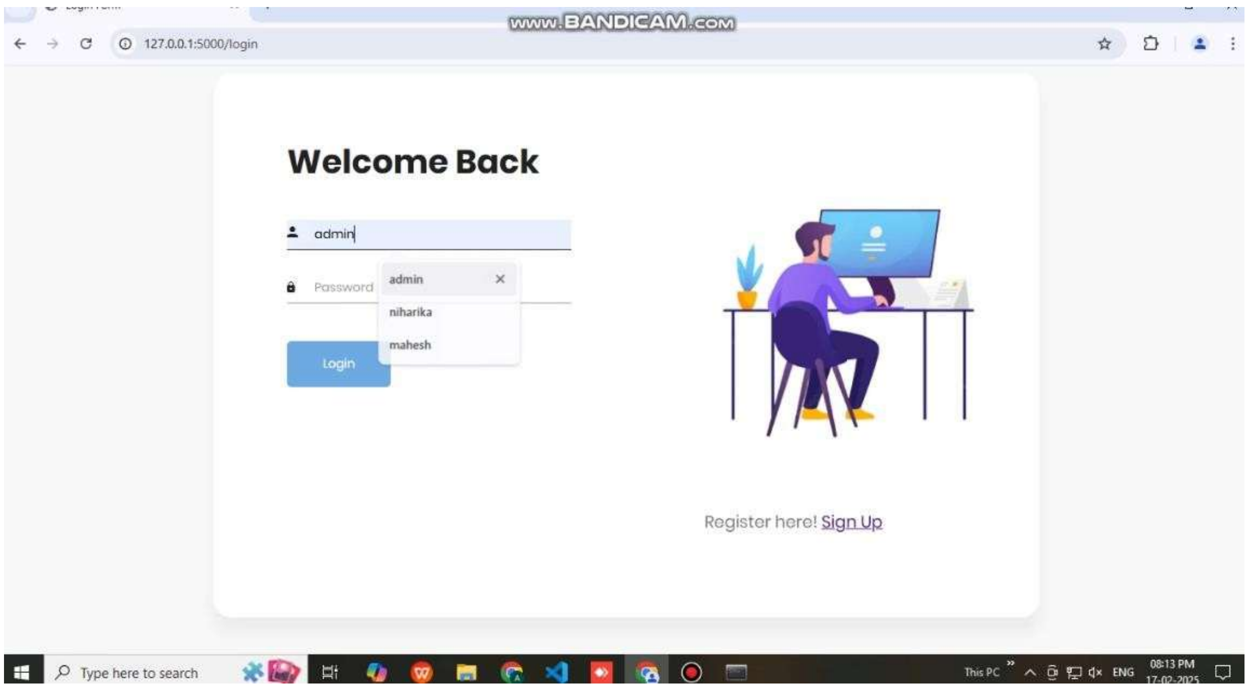
Step 1



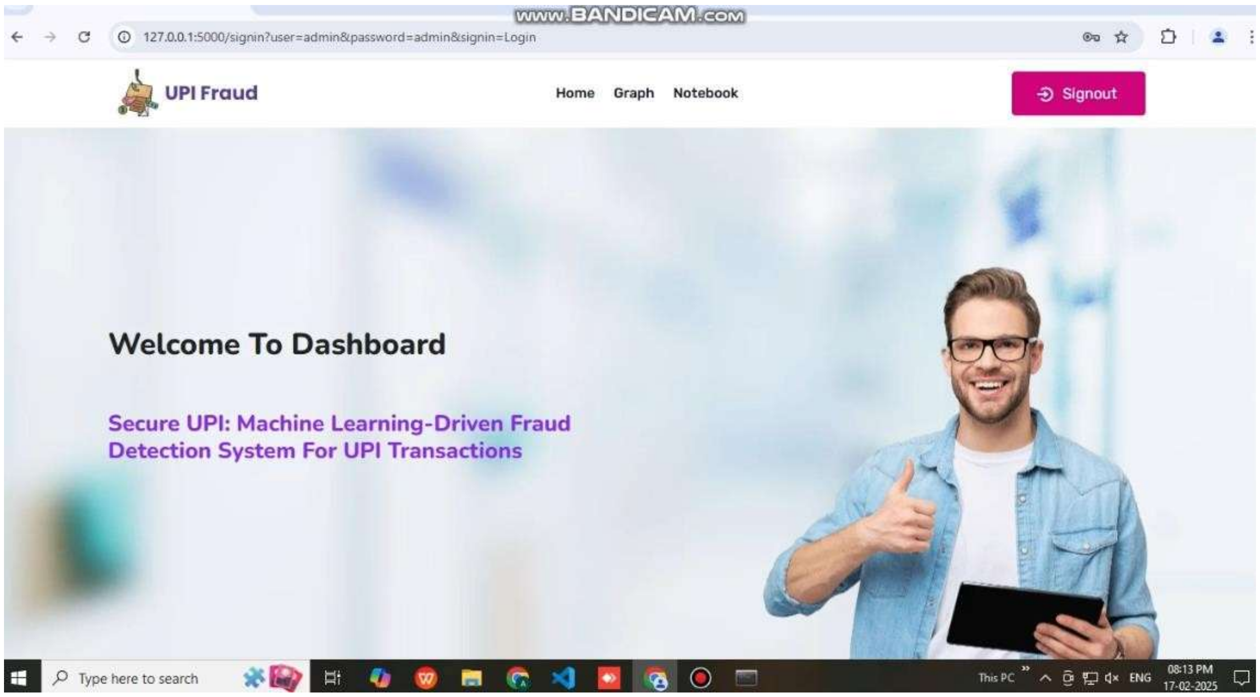
Step 2



Step 3



Step 4

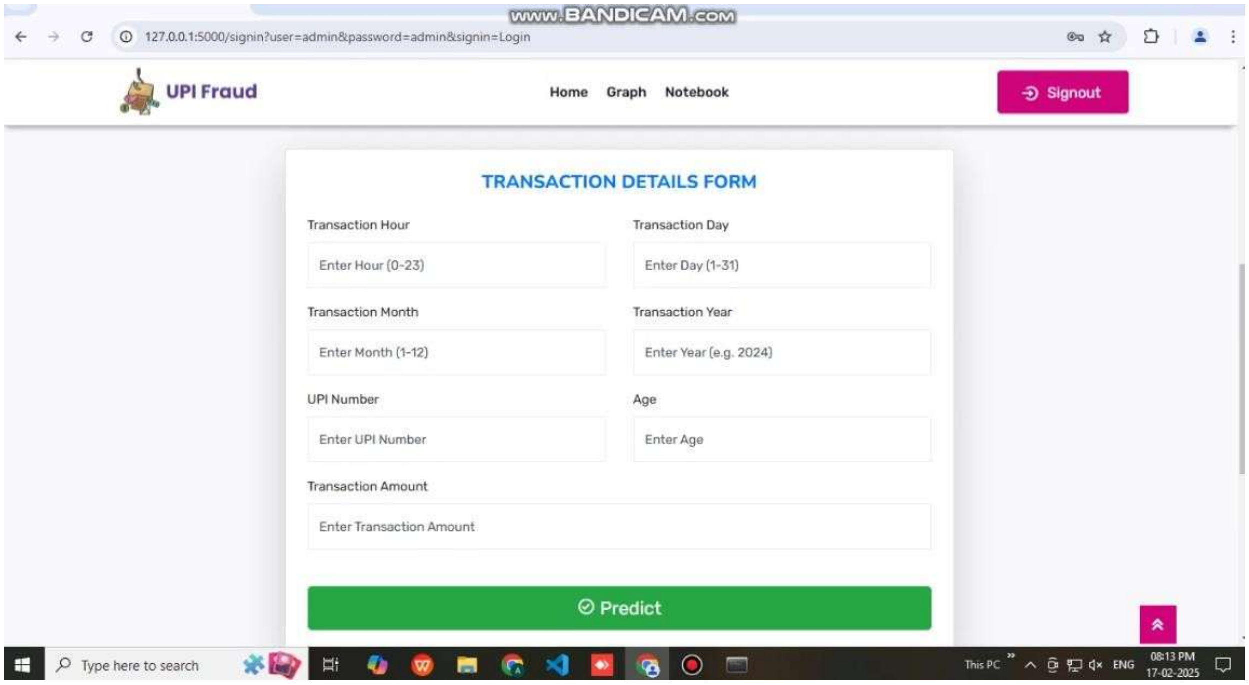

www.BANDICAM.com
127.0.0.1:5000/signin?user=admin&password=admin&signin=Login
UPI Fraud
Home
Graph
Notebook
Signout

Welcome To Dashboard

Secure UPI: Machine Learning-Driven Fraud Detection System For UPI Transactions

Step - 5

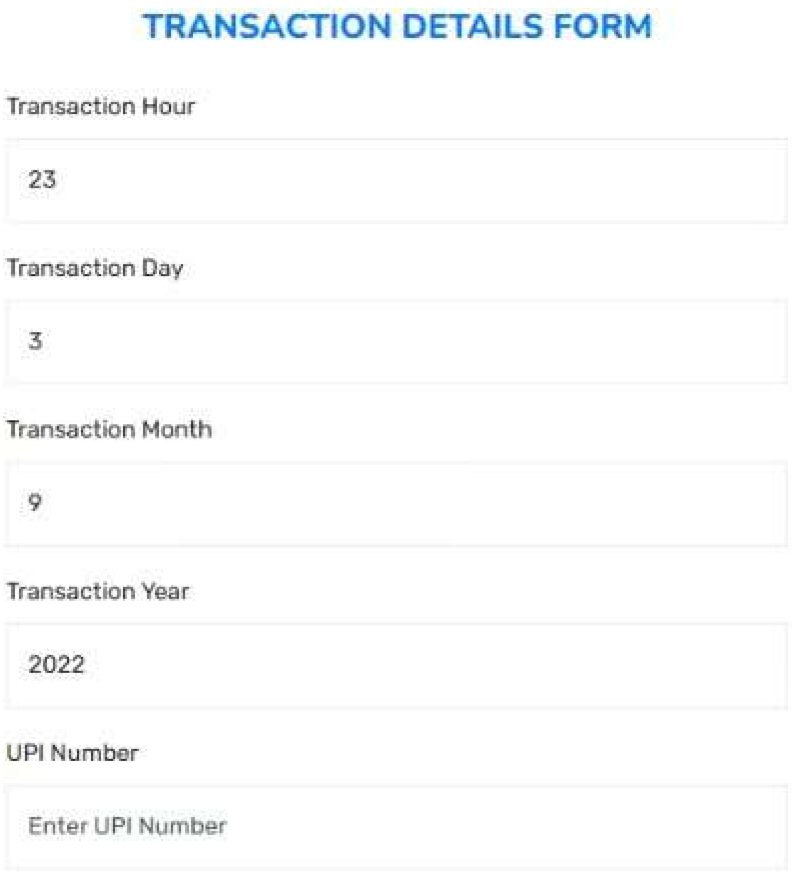
Step - 6


www.BANDICAM.com
127.0.0.1:5000/signin?user=admin&password=admin&signin=Login
UPI Fraud
Home
Graph
Notebook
Signout
Predict

TRANSACTION DETAILS FORM

Transaction Hour: Enter Hour (0-23)	Transaction Day: Enter Day (1-31)
Transaction Month: Enter Month (1-12)	Transaction Year: Enter Year (e.g. 2024)
UPI Number: Enter UPI Number	Age: Enter Age
Transaction Amount: Enter Transaction Amount	

Step - 7



TRANSACTION DETAILS FORM

Transaction Hour

23

Transaction Day

3

Transaction Month

9

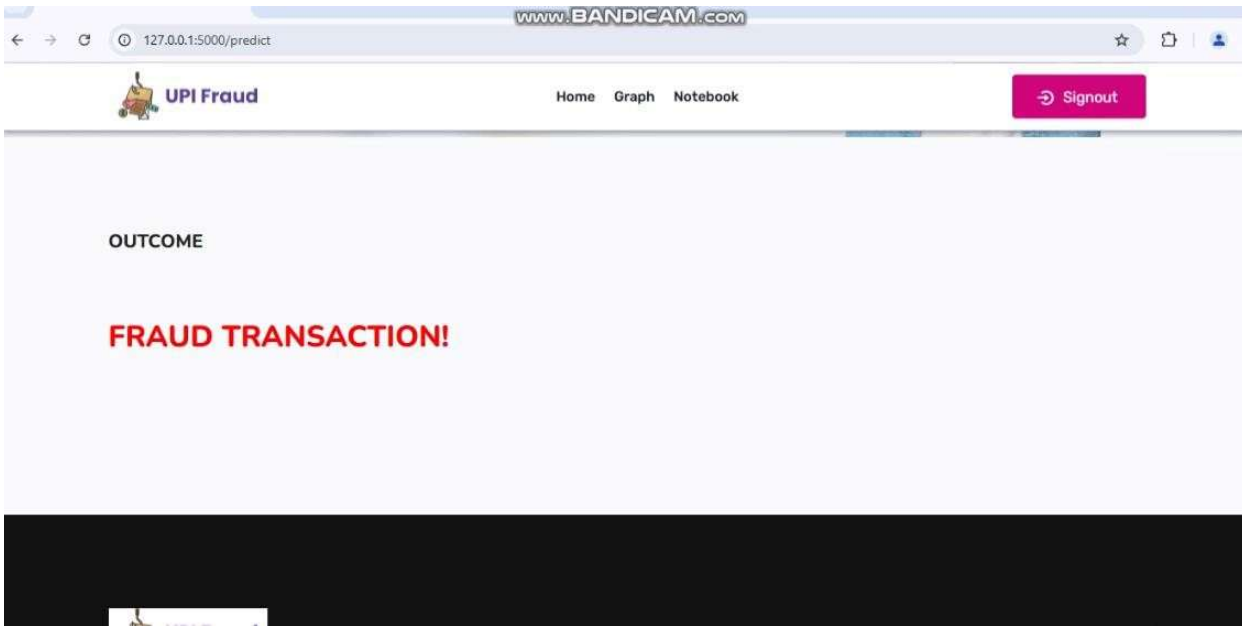
Transaction Year

2022

UPI Number

Enter UPI Number

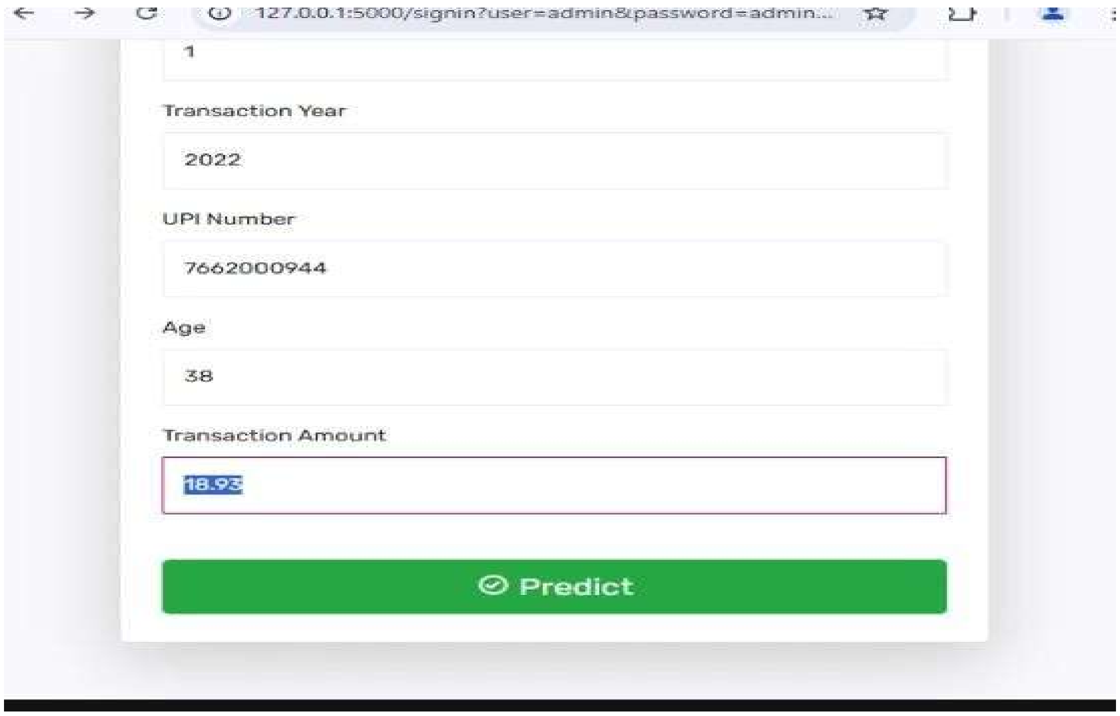
Step - 8



OUTCOME

FRAUD TRANSACTION!

Step – 9


127.0.0.1:5000/signin?user=admin&password=admin...

1

Transaction Year

2022

UPI Number

7662000944

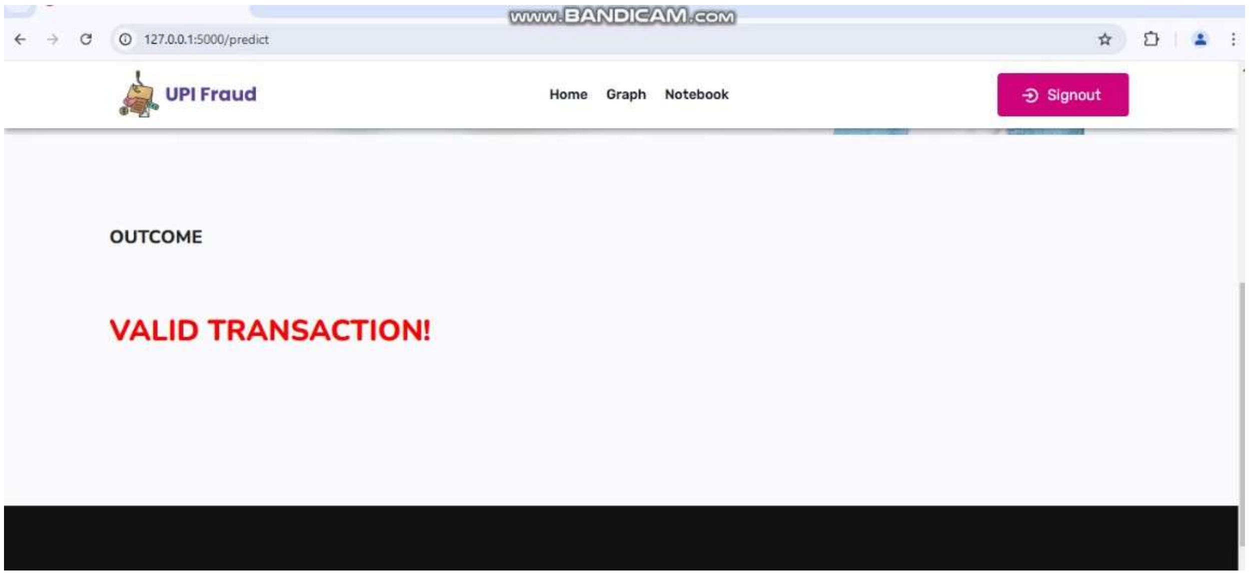
Age

38

Transaction Amount

18.93

Step – 10



OUTCOME

VALID TRANSACTION!

10. CONCLUSION

In conclusion, the machine learning-based fraud detection system for UPI transactions effectively enhances digital banking security. By employing multiple classification models, including Logistic Regression, Decision Tree, XGBoost, SVM, KNN, and Naïve Bayes, and utilizing ensemble techniques like Voting and Stacking Classifiers, the system provides high accuracy in fraud detection. The results show that XGBoost performs well with 98.2% accuracy, but the Stacking Classifier further improves this to 99.4%. The system's integration with a Flask-based frontend and SQLite-based user authentication allows for real-time fraud detection, ensuring robust and scalable protection for digital transactions. This project highlights the potential of machine learning and ensemble learning in building efficient, reliable, and secure fraud detection solutions for digital payment systems.

The future scope of this fraud detection system includes expanding the dataset to include more diverse transaction patterns, improving model generalization with additional feature engineering, and exploring deep learning techniques like neural networks for enhanced performance. Implementing real-time anomaly detection with continuous model updates could further improve accuracy. Additionally, integrating the system with other payment platforms and adopting advanced encryption for security would strengthen its robustness and scalability, making it adaptable for wider financial applications.

11. REFERENCES

[1] M. Adekunle and P. Ozoh, "Fraud detection model for illegitimate transactions", Kabale University Interdisciplinary Research Journal, vol. 2, no. 2, pp. 21-37, 2023.

[2] P. Boulieris, J. Pavlopoulos, A. Xenos and V. Vassalos, "Fraud detection with natural language processing", Machine Learning, vol. 1, pp. 22, 2023.

[3] B. Mytnyk, O. Tkachyk, N. Shakhovska, S. Fedushko and Y. Syerov, "Application of Artificial Intelligence for Fraudulent Banking Operations Recognition", Big Data and Cognitive-Computing, vol. 7, no. 2, pp. 93, 2023.

[4] R. Ridwan, S. Abdullah and F. Yusmita, "IMPLEMENTATION OF CASHLESS POLICY STRATEGIES TO MINIMIZE FRAUD IN THE GOVERNMENTSECTOR: SYSTEMIC REVIEW", Jurnal Akuntansi, vol. 12, no. 3, pp. 181-201, 2022.

[5] V. Chang, A. Di Stefano, Z. Sun and G. Fortino, "Digital payment fraud detection methods in digital ages and Industry 4.0", Computers and Electrical Engineering, vol. 100, pp. 107734, 2022.

[6] S. K. Bandyopadhyay and S. Dutta, "Detection of fraud transactions using recurrent neural network during COVID-19: fraud transaction during COVID-19", Journal of Advanced Research in Medical Science & Technology, vol. 7, no. 3, pp. 16-21, 2020, ISSN 2394-6539.

[7] S. Manocha, R. Kejriwal and D. A. Upadhyaya, "The impact of demonetization on digital payment transactions: a statistical study", Proceedings of International Conference on Advancements in Computing & Management (ICACM), September 2019.

[8] Diadiushkin, K. Sandkuhl and A. Maiatin, "Fraud detection in payments transactions: Overview of existing approaches and usage for instant payments", Complex Systems Informatics and Modeling Quarterly, no. 20, pp. 72-88, 2019.

[9] B. Baesens, S. Höppner and T. Verdonck, "Data engineering for fraud detection", Decision Support Systems, vol. 150, pp. 113492, 2021.

[10] M. Carminati, A. Baggio, F. Maggi, U. Spagnolini and S. Zanero, FraudBuster: temporal analysis and detection of advanced financial frauds. In Detection of Intrusions and Malware and Vulnerability Assessment:

15th International Conference DIMVA 2018 Saclay France June 28–29 2018 Proceedings 15, pp. 211-233, 2018.

[11] S. Rastogi, A. Sharma, C. Panse and V. M. Bhimavarapu, "Unified Payment Interface (UPI): A digital innovation and its impact on financial inclusion and economic development", Universal Journal of Accounting and Finance, vol. 9, no. 3, pp. 518-530, 2021.

[12] P. Gupta, A. Varshney, M. R. Khan, R. Ahmed, M. Shuaib and S. Alam, "Unbalanced Credit Card Fraud Detection Data: A Machine Learning-Oriented Comparative Study of Balancing Techniques", Procedia Computer-Science, vol. 218, pp. 2575-2584, 2023.

[13] J. Hariharakrishnan, S. Mohanavalli and K. S. Kumar, "Survey of pre-processing techniques for mining big data", 2017 international conference on computer communication and signal processing (ICCCSP), pp. 1-5, January 2017.

[14] S. Bhattacharyya, S. Jha, K. Tharakunnel and J. C. Westland, "Data mining for credit card fraud: A comparative study", Decision support systems, vol. 50, no. 3, pp. 602-613, 2011.

[15] B. Branco, P. Abreu, A. S. Gomes, M. S. Almeida, J. T. Ascensão and P. Bizarro, "Interleaved sequence rnns for fraud detection", Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 3101-3109, August 2020.

[16] B. Zhu, Z. Gao, J. Zhao and S. K. Vanden Broucke, "IRIC: An R library for binary imbalanced classification", -SoftwareX, vol. 10, pp. 100341, 2019.

[17] M. R. Dileep, A. V. Navaneeth and M. Abhishek, "A novel approach for credit card fraud detection using decision tree and random forest algorithms", 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), pp. 1025-1028, February 2021.

[18] M. Ahmed, A. N. Mahmood and M. R. Islam, "A survey of anomaly detection techniques in financial domain", Future Generation Computer Systems, vol. 55, pp. 278-288, 2016.

[19] M. A. Lavadkar, P. K. Thorat, A. R. Kasliwal, J. S. Gadekar and D. P. Deshmukh, "Fingerprint Biometric Based Online Cashless Payment

System", IOSR Journal of Computer Engineering (IOSR-JCE), ISSN 2278-0661.

[20] S. Purnama, C.S. Bangun and S. A. Faaroek, "The Effect of Transaction Experience Using Digital Wallets on User Satisfaction in Millennial Generation", Aptisi Transactions on Management (ATM), vol. 5, no. 2, pp. 161-168, 2021.