

A Minor Project Report on
PREDICTING LOAN DEFAULTERS WITH MACHINE LEARNING MODELS
FOR CREDIT CARD MANAGEMENT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In Partial fulfilment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

By

RATHOD KARTHIK	- (22RA5A0515)
BANOTHU MANI	- (22RA5A0521)
BAISKA NAVEEN KUMAR	- (22RA5A0523)
BHURUGUPALLY SURYA PRAKASH	- (22RA5A0538)

Under the guidance of

Mr. K. Sunil Kumar
Asst Professor, CSE Dept

Department of Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY
(Affiliated to JNTUH, Ghanpur(V), Ghatkesar(M), Medchal(D)-500088)

2021 -2025

KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

(Affiliated to JNTUH, Ghanpur(V), Ghatkesar(M), Medchal(D)-500088)



CERTIFICATE

This is to certify that the project work entitled “**Predicting Loan Defaulters with Machine Learning Models for Credit Card Management**” is submitted by Mr. R.KARTHIK, Mr. B.MANI, Mr. B.NAVEEN KUMAR, Mr. B.SURYA PRAKASH bonafied students of **Kommuri Pratap Reddy Institute of Technology** in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **Computer Science and Engineering** of the **Jawaharlal Nehru Technological University Hyderabad**, during the year 2024- 25.

Internal Guide
Mr. K. SUNIL KUMAR

HOD
Dr. S. KAVITHA

Project Coordinator
Mr. C. VIJAYARAJ

External Examiner

DECLARATION

We hereby declare that this project work entitled “**Predicting Loan Defaulters with Machine Learning Models for Credit Card Management**” in partial fulfillment of requirements for the award of degree of **Computer Science and Engineering** is a bonafide work carried out by us during the academic year 2024- 25.

We further declare that this project is a result of our effort and has not been submitted for the award of any degree by us to any institute.

By

R Karthik	-	(22RA5A0515)
B Mani	-	(22RA5A0521)
B Naveen Kumar	-	(22RA5A0523)
B Surya Prakash	-	(22RA5A0538)

ACKNOWLEDGEMENT

It gives us immense pleasure to acknowledge with gratitude, the help and support extended throughout the project report from the following:

We will be very much grateful to almighty our **Parents** who have made us capable of carrying out our job.

We express our profound gratitude to **Dr. RAVINDRA EKLARKER, Principal of Kommuri Pratap Reddy Institute of Technology**, who has provided necessary infrastructure and resources in completing our project report successfully.

We are grateful to **Dr. S. KAVITHA** who is our **Head of the Department, CSE** for her amiable ingenious and adept suggestions and pioneering guidance during the project report.

We express our gratitude and thanks to the **coordinator Mr. C. VIJAYARAJ** of our department for his contribution for making it success within the given time duration.

We express our deep sense of gratitude and thanks to **Internal Guide, Mr. K. SUNIL KUMAR** for his guidance during the project report.

We are also very thankful to our **Management, Staff Members** and all **Our Friends** for their valuable suggestions and timely guidance without which we would not have been completed it.

By

R Karthik	-	(22RA5A0515)
B Mani	-	(22RA5A0521)
B Naveen Kumar	-	(22RA5A0523)
B Surya Prakash	-	(22RA5A0538)



KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision of the Institute

To emerge as a premier institute for high quality professional graduates who can contribute to economic and social developments of the Nation.

Mission of the Institute

Mission	Statement
IM ₁	To have holistic approach in curriculum and pedagogy through industryinterface to meet the needs of Global Competency.
IM ₂	To develop students with knowledge, attitude, employability skills,entrepreneurship, research potential and professionally Ethical citizens.
IM ₃	To contribute to advancement ofEngineering & Technology that wouldhelp to satisfy the societal needs.
IM ₄	To preserve, promote cultural heritage, humanistic values and Spiritualvalues thus helping in peace and harmony in the society.



KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Vision of the Department

To Provide Quality Education in Computer Science for the innovative professionals to work for the development of the nation.

Mission of the Department

Mission	Statement
DM₁	Laying the path for rich skills in Computer Science through The basic knowledge of mathematics and fundamentals of engineering
DM₂	Provide latest tools and technology to the students as a part of learning infrastructure
DM₃	Training the students towards employability and entrepreneurship to meet the societal needs.
DM₄	Grooming the students with professional and social ethics.



KOMMURI PRATAP REDDY INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Program Educational Objectives (PEOs)

PEO's	Statement
PEO1	The graduates of Computer Science and Engineering will have successful career in technology.
PEO2	The graduates of the program will have solid technical and professional foundation to continue higher studies.
PEO3	The graduate of the program will have skills to develop products, offer services and innovation.
PEO4	The graduates of the program will have fundamental awareness of industry process, tools and technologies.

Program Outcomes

PO1	Engineering Knowledge: Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental context, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9	Individual and team network: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, being able to comprehend and write effective reports and design documentation, make Effective presentations, and give and receive clear instructions.
PO11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work,as a member and leader in a team, to manage projects and in multidisciplinary environment.
PO12	Life-Long learning: Recognize the need for, and have the preparation and able to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

PSO1	Foundation of mathematical concepts: To use mathematical methodologies to crack problem using suitable mathematical analysis, data structure and suitable algorithm.
PSO2	Foundation of Computer Science: The ability to interpret the fundamental concepts and methodology of computer systems. Students can understand the functionality of hardware and software aspects of computer systems.
PSO3	Foundation of Software development: The ability to grasp the software development lifecycle and methodologies of software systems. Possess competent skills and knowledge of software design process.

Table of Contents

ABSTRACT	01
CHAPTER 1	02
1.1 Objectives.....	02
1.2 Research Motivation	03
1.3 Problem Statement	03
1.4 Need and Significance	04
1.5 Applications	05
CHAPTER 2.....	06
2.1 LITERATURE SURVEY.....	06
CHAPTER-3	09
3.1 Overview	09
3.2 Limitations	10
CHAPTER 4.....	12
4.1 Overview	12
4.2 Data Preprocessing.....	14
4.3 Splitting the Dataset	16
4.4 Random Forest Classifier	18
4.5 Important features of Random Forest.....	18
4.6 Advantages of Random Forest	20
4.7 Applications of Random Forest.....	21
CHAPTER 5 UML.....	22
5.1 Class diagram	22
5.2 Sequence Diagram.....	23
5.3 Activity diagram	23
5.4 Deployment diagram.....	24
5.5 Use Case diagram	25
5.6 Component diagram	25
5.7 Data Flow diagram.....	26
5.7 System Architecture	26
CHAPTER 6 Software Environment.....	28
6.1 What is Python?.....	28
6.2 Advantages Of Python	28
6.3 Disadvantages Of Python.....	31
6.4 History Of Python	32

6.5 Python Development Steps	32
6.6 Purpose.....	33
6.7 Modules Used In Python.....	34
6.8 Installation of Python	39
 CHAPTER 7 System Requirements	 43
7.1 Software Requirements.....	43
7.2 Hardware Requirements	43
 CHAPTER 8 Functional Requirements	 44
8.1 Output Design.....	44
8.2 Input Types.....	45
8.3 Input Media	45
8.4 Error Avoidance	46
8.5 Error Detection	46
8.6 User Initiated Interfaces.....	47
8.7 Computer Initiated Interfaces	47
 CHAPTER 9 Source Code.....	 49
CHAPTER 10 Results And Discussions.....	57
10.1 Implementation Description	57
10.2 Dataset Description	59
10.3 Result and Description.....	60
CHAPTER 11 CONCLUSION AND FUTURE SCOPE.....	63
11.1 Conclusion.....	63
11.2 Future Scope.....	63
 REFERENCES	 64

ABSTRACT

The financial sector faces significant challenges in assessing loan eligibility due to the complexity and volume of applications. Statistics indicate that fraudulent loan applications result in substantial financial losses, with global figures reaching billions of dollars annually. Accurate prediction of loan eligibility is vital to safeguard financial institutions and ensure fair lending practices. As the volume of loan applications continues to grow, traditional manual assessment methods become increasingly impractical and prone to errors. There is a pressing need for automated, data-driven solutions to accurately evaluate loan eligibility and detect potential fraud. Manual loan assessment processes are labor-intensive and susceptible to human error, leading to inconsistencies and potential oversight. These methods often fail to detect subtle indicators of fraud, resulting in significant financial losses. The reliance on subjective judgment can introduce biases, affecting the fairness and accuracy of loan decisions. Additionally, the manual verification of extensive data points is time-consuming, delaying the approval process and impacting customer satisfaction. Our proposed solution employs machine learning algorithms to predict the eligibility of loan applications and detect fraudulent cases using the SYL Bank dataset. The dataset includes various features such as age, occupation, marital status, credit score, income level, and past financial behavior. By training ML models on this comprehensive dataset, we aim to develop a predictive system that accurately identifies eligible applicants and flags potential fraud. This approach promises to enhance the precision, efficiency, and security of loan processing, ensuring better outcomes for financial institutions and their clients.

CHAPTER 1

INTRODUCTION

In the realm of financial decision-making, predicting loan eligibility plays a crucial role in mitigating risks and optimizing resource allocation for lending institutions. This project focuses on leveraging the SYL BANK dataset to develop robust statistical models and perform cross-validation analyses aimed at enhancing the accuracy and reliability of loan approval predictions. By employing advanced machine learning techniques and statistical methodologies, the study aims to uncover meaningful patterns and relationships within the dataset, thereby improving the efficiency of loan assessment processes.

Drawing upon methodologies such as statistical modeling and cross-validation, this research endeavors to provide insights into the factors influencing loan approval decisions. The SYL BANK dataset, comprising comprehensive borrower information including demographic profiles, financial histories, and credit scores, serves as a foundational resource for training and validating predictive models. Through rigorous analysis and validation, this study seeks to enhance the predictive power of these models, enabling more informed and data-driven lending decisions.

By integrating diverse analytical approaches and leveraging the richness of the SYL BANK dataset, this project aims to contribute to the field of financial analytics and risk management. Ultimately, the insights gleaned from this study are expected to facilitate smarter lending practices, improve customer satisfaction, and optimize the overall operational efficiency of lending institutions.

1.1 Objectives

The primary objective of this study is to leverage machine learning algorithms for predicting loan eligibility and detecting potential fraud using the SYL Bank dataset. Traditional methods of assessing loan applications are increasingly inefficient and prone to errors, particularly in detecting fraudulent activities. By applying data-driven approaches, we aim to develop a robust predictive model that enhances the accuracy and efficiency of loan processing at financial institutions. The objective is to create a system that not only automates the evaluation of loan applications but also significantly reduces the risk of financial losses due to fraudulent activities. This study seeks to establish a reliable framework that ensures fair lending practices,

improves decision-making processes, and enhances customer satisfaction in the financial sector.

1.2 Research Motivation

The motivation behind this research stems from the critical challenges faced by the financial sector in assessing loan eligibility and detecting fraud. Traditional manual assessment methods are labor-intensive, time-consuming, and often prone to human error, leading to inconsistencies and potential financial losses. The increasing volume of loan applications exacerbates these challenges, making it imperative to adopt automated, data-driven solutions. By utilizing machine learning techniques, we aim to address these issues comprehensively. This research seeks to contribute to the development of innovative tools that not only streamline the loan approval process but also enhance the accuracy of fraud detection. By mitigating risks associated with fraudulent applications, financial institutions can safeguard their resources, uphold fair lending practices, and foster trust among customers.

1.3 Problem Statement

The core problem addressed in this study revolves around the inefficiencies and risks associated with manual loan assessment processes in the financial sector. These processes are often subjective, error-prone, and incapable of effectively identifying fraudulent loan applications. As a result, financial institutions face significant challenges in maintaining operational efficiency, mitigating financial risks, and ensuring fair treatment of loan applicants. The need for automated, data-driven solutions becomes paramount to overcome these limitations. This research aims to develop and deploy machine learning models that accurately predict loan eligibility and identify potential fraud based on a comprehensive dataset provided by SYL Bank. By doing so, we aim to revolutionize the loan approval process, improve decision-making accuracy, and enhance overall operational efficiency in the financial industry.

1.4 Need and Significance

1.4.1 Enhanced Accuracy: By employing machine learning algorithms, the study aims to significantly improve the accuracy of loan eligibility predictions and fraud detection. This ensures that only eligible applicants receive loans while minimizing the risk of financial losses due to fraudulent activities.

1.4.2 Operational Efficiency: Automating the loan assessment process reduces the time and resources spent on manual verification, leading to faster decision-making and improved customer satisfaction.

1.4.3 Risk Mitigation: Identifying potential fraud through data-driven insights helps financial institutions mitigate risks associated with fraudulent loan applications, thereby safeguarding their financial resources and reputation.

1.4.4 Fair Lending Practices: Implementing unbiased machine learning models promotes fair lending practices by objectively evaluating loan applications based on predefined criteria, reducing the impact of human biases in decision-making.

1.4.5 Customer Trust: Enhancing the security and efficiency of loan processing fosters trust and confidence among customers, encouraging continued engagement and loyalty with financial institutions.

1.5 Applications

1.5.1 Financial Institutions: Deploying machine learning models for loan eligibility prediction and fraud detection enables financial institutions to streamline operations, reduce costs, and enhance risk management practices.

1.5.2 Regulatory Bodies: By implementing robust predictive models, regulatory bodies can ensure compliance with fair lending laws and regulations, thereby promoting transparency and accountability within the financial sector.

1.5.3 Credit Risk Management: Machine learning-based solutions provide valuable insights into credit risk assessment, helping financial institutions make informed decisions about loan approvals and portfolio management.

1.5.4 Customer Experience: Automated loan processing enhances the overall customer experience by reducing processing times, improving accuracy in decision-making, and ensuring fair treatment of loan applicants.

CHAPTER 2

LITERATURE SURVEY

Yoon et al [1]. explored loan eligibility prediction using machine learning techniques, emphasizing their application in financial decision-making. The study highlighted the effectiveness of machine learning models such as Support Vector Machines (SVM), Random Forests, and Neural Networks in analyzing borrower data to assess creditworthiness. By leveraging large-scale datasets from financial institutions, their research contributed to improving loan approval processes and risk management strategies in the banking sector.

Singh and Yadav [2]. conducted a comparative study of machine learning algorithms for loan eligibility prediction. Their research evaluated the performance of algorithms including Decision Trees, Logistic Regression, and k-Nearest Neighbors (k-NN) across various metrics such as accuracy, precision, and computational efficiency. This comparative analysis provided insights into the strengths and weaknesses of different approaches, aiding financial institutions in selecting suitable models based on specific requirements and dataset characteristics.

Kumar and Gupta [3] et al. focused on predicting loan eligibility using ensemble learning approaches. Their study demonstrated the benefits of ensemble techniques such as Bagging, Boosting, and Stacking in integrating multiple models to enhance predictive accuracy. By combining diverse algorithms and leveraging their complementary strengths, the research illustrated the robustness of ensemble learning in handling complex decision-making tasks and improving the reliability of loan approval predictions.

Kaur and Kaur [4]. conducted a comprehensive comparative study on loan approval prediction using data mining techniques. Their research reviewed methodologies including Association Rule Mining, Decision Support Systems, and Predictive Analytics in the context of loan assessment. The study highlighted advancements in data mining applications for financial analytics, offering insights into evolving trends and challenges in leveraging data-driven approaches to optimize loan processing and risk assessment.

Rani and Sharma [5] et al. provided a detailed review of loan prediction using data mining techniques. Their work synthesized existing literature on methodologies such as Classification and Regression Analysis, highlighting their applications and effectiveness in predicting loan outcomes. By analyzing the strengths and limitations of different techniques, the review

identified opportunities for further research in refining predictive models and enhancing decision support systems for loan approval processes.

Nair and Sindhya [6], explored loan eligibility prediction using decision tree and k-nearest neighbors algorithms. Their study investigated the application of these methods in assessing borrower risk profiles based on demographic, financial, and credit history data. By comparing decision tree-based approaches with instance-based learning methods like k-NN, the research contributed to understanding the trade-offs between model interpretability and predictive accuracy in loan assessment tasks.

Mishra and Verma [7] et al. implemented neural network and decision tree models for loan prediction. Their research emphasized the capability of neural networks to capture nonlinear relationships and complex patterns in borrower data, complementing the interpretability of decision tree models. By integrating these techniques, the study illustrated advancements in machine learning applications for financial forecasting and risk management in loan approval scenarios.

Das and Chakraborty [8]. conducted research on loan eligibility prediction using ensemble machine learning techniques. Their study explored the integration of multiple models to enhance predictive accuracy and reliability in assessing loan approval outcomes. By leveraging ensemble learning methodologies such as Bagging and Boosting, the research contributed to improving decision-making processes in financial institutions.

Banerjee and Sural [9]. conducted a survey on loan approval prediction using data mining techniques. Their study reviewed various methodologies including Classification, Association Rule Mining, and Predictive Analytics, highlighting their applications and effectiveness in assessing borrower creditworthiness. The survey provided insights into evolving trends and challenges in leveraging data-driven approaches for optimizing loan processing and risk assessment.

Kumar and Reddy [10] focused on predicting loan approval using machine learning techniques, presenting a case study approach. Their research applied algorithms such as Decision Trees and Logistic Regression to analyze borrower data and predict loan eligibility. By evaluating different models, the study provided practical insights into the application of machine learning in financial decision-making processes.

Choudhury and Bose [11]. explored loan approval prediction using Support Vector Machine (SVM) and Random Forest classifiers. Their research compared the performance of these

algorithms in assessing borrower risk profiles based on demographic and financial data. The study highlighted the strengths of SVM and Random Forests in handling complex decision-making tasks and improving the accuracy of loan approval predictions.

Jain and Kothari et al [12] investigated loan eligibility prediction using the Decision Tree algorithm. Their study focused on analyzing borrower attributes to classify loan applications into approved or rejected categories. By applying Decision Tree techniques, the research provided insights into the interpretability and predictive power of this algorithm in loan assessment scenarios.

Patel and Patel [13] conducted a survey on loan prediction using machine learning techniques. Their study reviewed methodologies such as Neural Networks, Decision Trees, and Support Vector Machines for assessing borrower creditworthiness. The survey summarized advancements in machine learning applications for loan approval processes, highlighting trends and challenges in predictive analytics for financial decision-making.

Shukla and Jain [14]. performed a comparative study of machine learning algorithms for loan approval prediction. Their research evaluated algorithms including Random Forest, Gradient Boosting, and k-Nearest Neighbors (k-NN) across various metrics such as accuracy and computational efficiency. This comparative analysis provided insights into the strengths and limitations of different approaches, aiding financial institutions in selecting appropriate models for loan assessment tasks.

Ravi and Narasimha Murthy et al [15]. conducted a survey on loan approval prediction using machine learning techniques. Their research reviewed methodologies such as Logistic Regression, Decision Trees, and Ensemble Methods for assessing borrower credit risk. The survey contributed to understanding the application of machine learning in financial forecasting and risk management, emphasizing the evolution of predictive analytics in loan approval processes.

CHAPTER 3

EXISTING SYSTEM

3.1 Overview

The traditional system for fraud detection before the advent of machine learning and data-driven approaches typically relied on rule-based systems and manual review processes. Here's an overview of the traditional system and its limitations compared to the project using machine learning:

1. Rule-Based Systems:

- **Manual Rules:** Fraud detection relied heavily on predefined rules and thresholds set by domain experts. For example, transactions above a certain amount might trigger a manual review.
- **Thresholds:** Fixed thresholds were used to flag suspicious activities, such as unusual transaction amounts or patterns.

2. Manual Review:

- **Human Oversight:** Suspicious transactions or activities identified by the rules were manually reviewed by fraud analysts.
- **Subjectivity:** Decisions often relied on the judgment and experience of analysts, which could vary in consistency and accuracy.
- **Time-Consuming:** Manual reviews were labor-intensive and time-consuming, limiting the scalability and real-time responsiveness of the system.

3. Limited Data Utilization:

- **Data Silos:** Data from different sources (e.g., transaction data, user behavior data) were often stored in separate systems, limiting comprehensive analysis.
- **Limited Historical Analysis:** Historical data analysis was limited to simple trend spotting rather than complex pattern recognition.

3.2 Limitations

1. Scalability:

- Traditional systems struggled to scale with increasing transaction volumes and evolving fraud tactics. They often couldn't handle large datasets or complex patterns efficiently.

2. Real-Time Detection:

- Real-time fraud detection was challenging due to the reliance on manual rules and reviews. Delays in detection could lead to higher losses.

3. Adaptability:

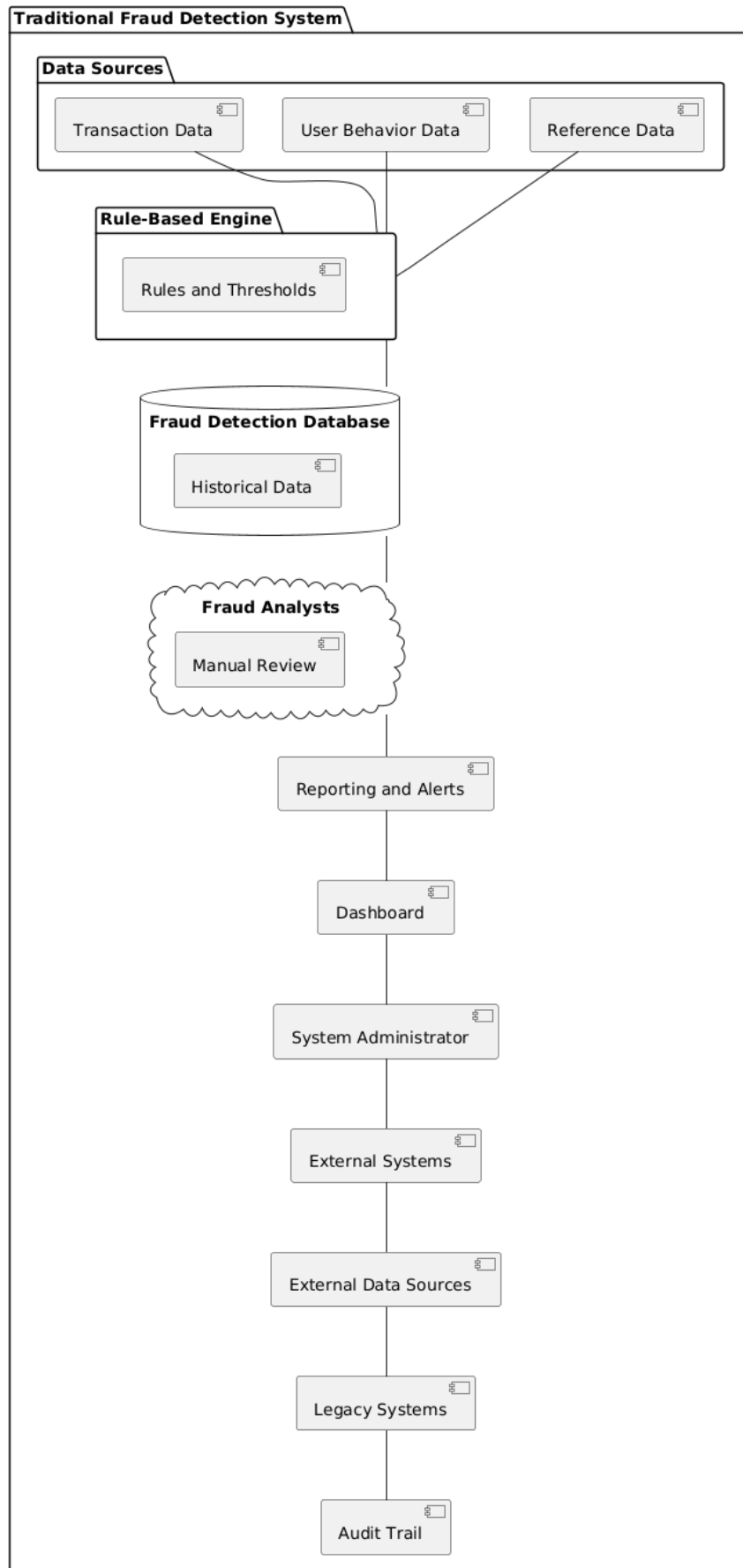
- Difficulty in adapting to new fraud patterns: Traditional systems often lagged behind in adapting to new fraud tactics and patterns, requiring constant manual updates to rules.

4. Accuracy:

- Subjectivity in manual reviews and reliance on fixed thresholds could lead to false positives (legitimate transactions flagged as fraud) or false negatives (fraudulent transactions missed).

5. Cost and Efficiency:

- High operational costs: Manual reviews and rule updates were costly and inefficient compared to automated systems.
- Limited ROI: Inefficient detection methods could result in higher fraud losses despite investment in detection systems.



CHAPTER 4

PROPOSED METHODOLOGY

4.1 Overview

This structured approach ensures the development of accurate and reliable models for predicting loan eligibility based on the SYL Bank dataset, incorporating statistical modeling principles and rigorous cross-validation analysis for model validation and optimization.

This project focuses on predicting loan eligibility using the SYL Bank dataset through statistical modeling and cross-validation analysis. The primary objective is to develop accurate models that can assess whether a loan applicant qualifies based on various factors present in the dataset. The key steps involved in this project are outlined below:

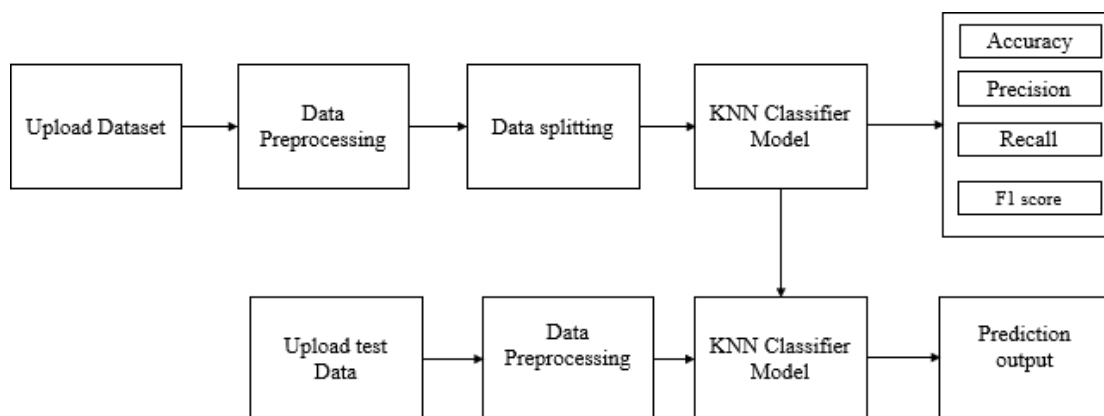


Figure 1:Block Diagram

Step 1. Importing Libraries:

- Pandas and Numpy: For data manipulation and numerical operations.
- Matplotlib and Seaborn: For data visualization.
- Scikit-learn: For implementing machine learning algorithms and evaluation metrics.
- Joblib: For saving and loading trained models.
- OS: For file operations.

Step 2. Loading and Exploring the Dataset:

- Dataset Loading: Load the SYL Bank dataset using Pandas.
- Initial Exploration: Display the first few rows, check for missing values, and obtain summary statistics to understand the dataset.

Step 3. Data Preprocessing:

- Handling Missing Values: Address missing values through imputation or dropping rows/columns.
- Feature Engineering: Create new features if necessary, such as calculating loan-to-income ratios or categorizing data.

Step 4. Data Visualization:

- Histograms and Plots: Visualize distributions of key variables (e.g., income, loan amount) to understand their impact on loan eligibility.

Step 5. Data Splitting:

- Feature and Target Separation: Separate the dataset into features (X) and the target (y) variables.
- Train-Test Split: Split the data into training and testing sets (e.g., 80% training, 20% testing) for model training and evaluation.

Step 6. Model Training and Evaluation:

- Define Metrics Calculation Function: Create a function to calculate and display metrics such as accuracy, precision, recall, F1-score, and confusion matrix.
- 6.1 Logistic Regression Model and Random Forest Model:
- Load Saved Model: Check if a saved model exists and load it if available.
- Train New Model: Train logistic regression and random forest models if no saved models exist.
- Save Trained Model: Save the trained models to disk for future use.
- Evaluate Models: Evaluate the models' performance on the test set using the defined metrics function.
- Store and Display Metrics: Store performance metrics of both models in a DataFrame for comparison and display.

Step 7. Cross-Validation Analysis:

- Implement Cross-Validation: Use techniques like k-fold cross-validation to assess model performance and generalization ability.
- Tune Hyperparameters: Optimize model performance by tuning hyperparameters using cross-validation results.

Key Components

- Data Handling: Load, clean, and preprocess data to ensure data quality and readiness for modeling.
- Visualization: Visualize data distributions and relationships to gain insights into factors influencing loan eligibility.
- Modeling:
 - Logistic Regression: Utilize a linear model for binary classification of loan eligibility.
 - Random Forest: Implement an ensemble method to capture non-linear relationships and interactions in the data.
- Model Evaluation: Assess model performance using various metrics and techniques like cross-validation to ensure robustness and reliability.
- Prediction and Decision Making: Extend trained models to predict loan eligibility for new applicants, supporting decision-making processes in real-world scenarios.

4.2 Data Preprocessing

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task. A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data

and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set

Importing Libraries: To perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

Numpy: Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

```
import numpy as nm
```

Here we have used nm, which is a short name for Numpy, and it will be used in the whole program.

Matplotlib: The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub-library pyplot. This library is used to plot any type of charts in Python for the code. It will be imported as below:

```
import matplotlib.pyplot as mpt
```

Here we have used mpt as a short name for this library.

Pandas: The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. Here, we have used pd as a short name for this library. Consider the below image:

```
1 # importing libraries
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import pandas as pd
5
```

Handling Missing data: The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset. There are mainly two ways to handle missing data, which are:

- By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.
- By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.

Encoding Categorical data: Categorical data is data which has some categories such as, in our dataset; there are two categorical variables, Country, and Purchased. Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So, it is necessary to encode these categorical variables into numbers.

4.3 Splitting the Dataset

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

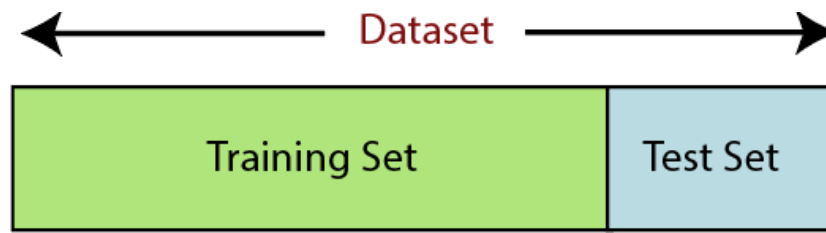


Figure 4.2: Splitting the dataset.

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```

Explanation

- In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
- In the second line, we have used four variables for our output that are
- `x_train`: features for the training data
- `x_test`: features for testing data
- `y_train`: Dependent variables for training data
- `y_test`: Independent variable for testing data
- In `train_test_split()` function, we have passed four parameters in which first two are for arrays of data, and `test_size` is for specifying the size of the test set. The `test_size` maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.
- The last parameter `random_state` is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42

4.4 Random Forest Classifier

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

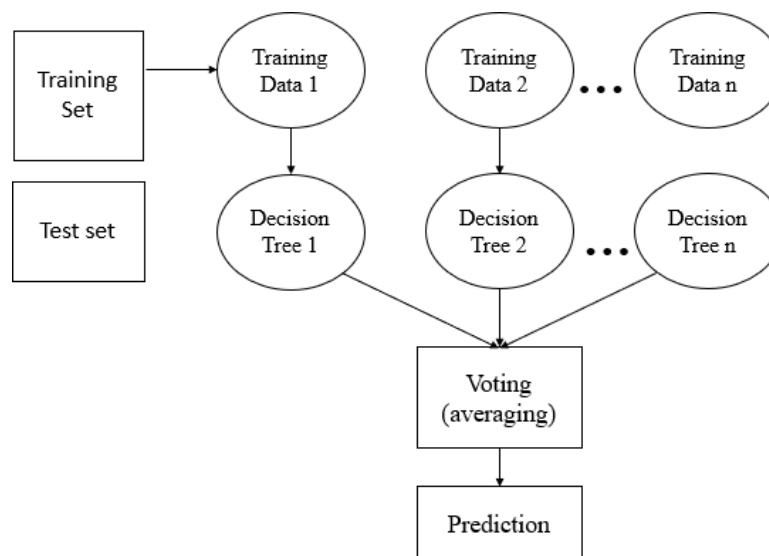


Figure: Random Forest algorithm

Random Forest algorithm

Step 1: In Random Forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

Important Features of Random Forest

- **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
- **Immune to the curse of dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
- **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
- **Train-Test split**- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
- **Stability**- Stability arises because the result is based on majority voting/ averaging.

4.5.1 Assumptions for Random Forest

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the Random Forest algorithm

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

4.5.2 Types of Ensembles

Before understanding the working of the random forest, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest. Bagging, also known as Bootstrap Aggregation is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap

Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

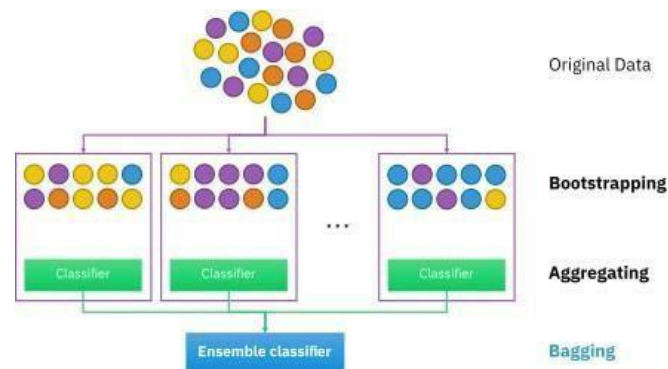


Figure. RF Classifier analysis.

Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

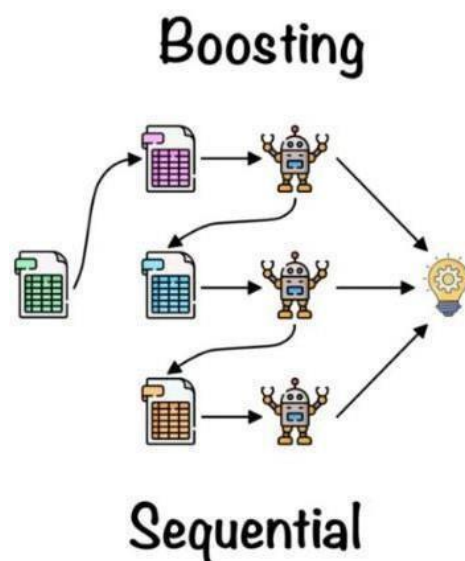


Figure. Boosting RF Classifier.

Advantages of Random Forest

- It can be used in classification and regression problems.

- It solves the problem of overfitting as output is based on majority voting or averaging.
- It performs well even if the data contains null/missing values.
- Each decision tree created is independent of the other thus it shows the property of parallelization.
- It is highly stable as the average answers given by a large number of trees are taken.
- It maintains diversity as all the attributes are not considered while making each decision tree though it is not true in all cases.
- It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.

Applications of Random Forest: There are mainly four sectors where Random Forest mostly used:

- **Banking:** Banking sector mostly uses this algorithm for the identification of loan risk.
- **Medicine:** With the help of this algorithm, disease trends and risks of the disease scan be identified.
- **Land Use:** We can identify the areas of similar land use by this algorithm.
- **Marketing:** Marketing trends can be identified using this algorithm.

CHAPTER 5

UML DIAGRAMS

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

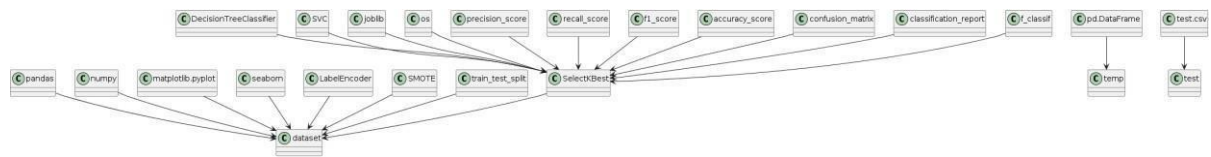
The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful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

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

Class Diagram

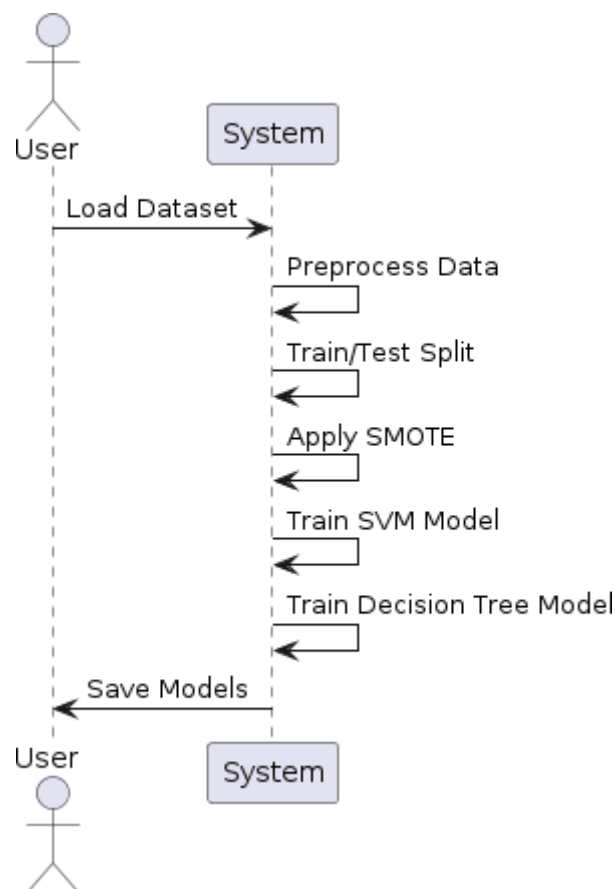
The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an “is-a” or “has-a” relationship.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

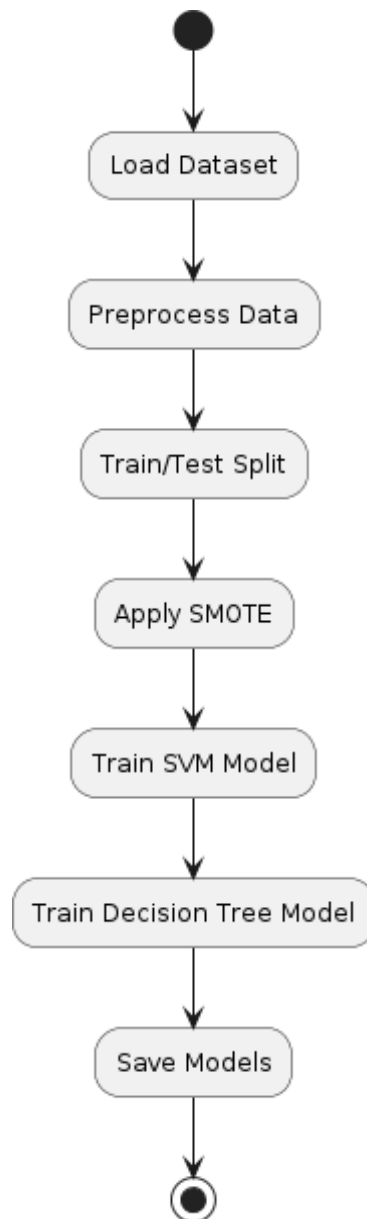


Sequence Diagram

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



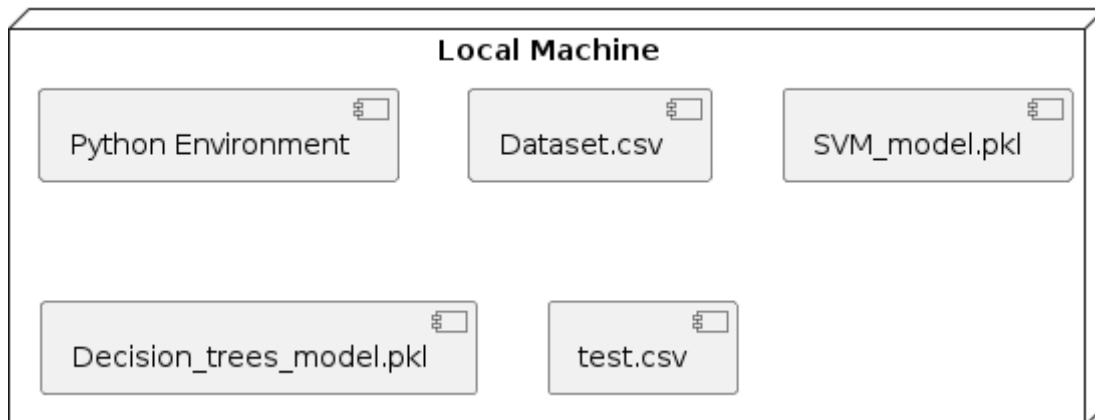
Activity diagram: Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.



Deployment diagram

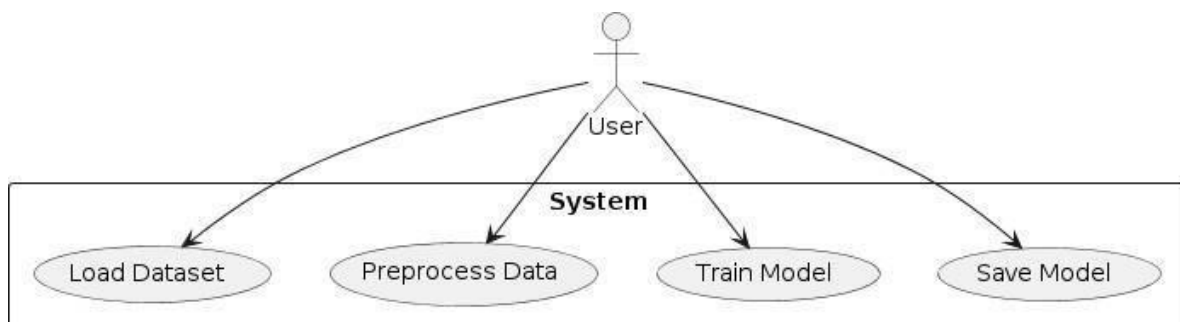
A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components exist (e.g., a web server, an application server, and a database), what software components (“artifacts”) run on each node (e.g., web application, database), and how the different pieces are connected (e.g., JDBC, REST, RMI). The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

Deployment diagram

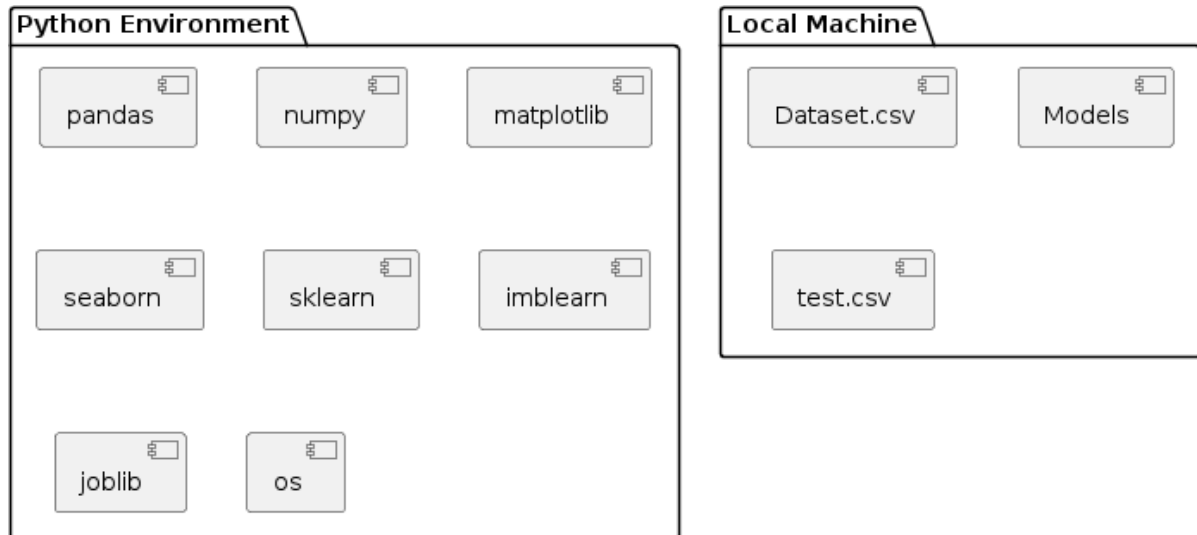


Use case diagram

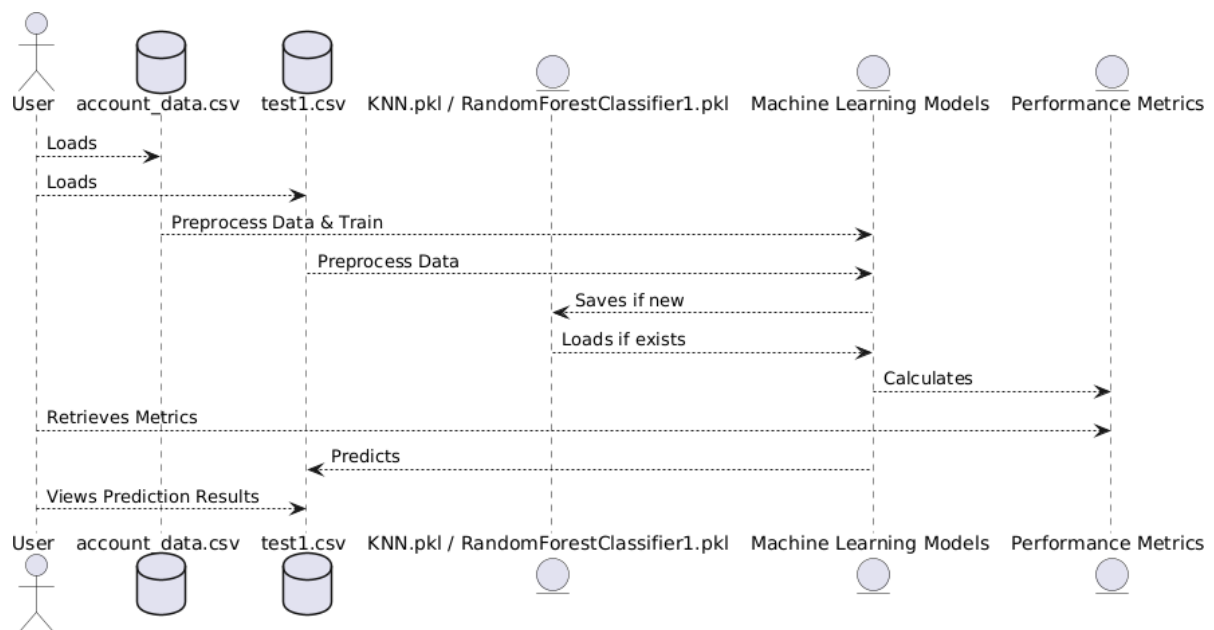
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies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



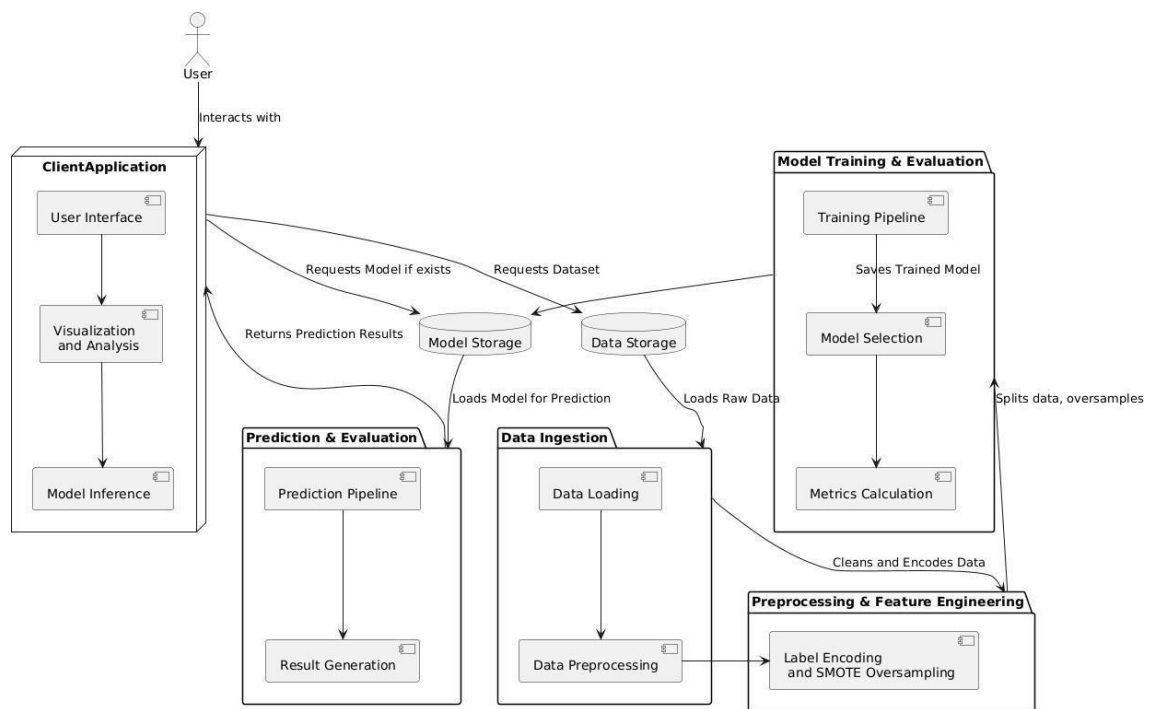
Component diagram: Component diagram describes the organization and wiring of the physical components in a system.



Dataflow Diagram:



System Architecture:



CHAPTER 6

SOFTWARE ENVIRONMENT

What is Python?

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

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

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

Advantages of Python

1. Extensive Libraries

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

2. Extensible

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

3. Embeddable

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

4. Improved Productivity

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

5. IOT Opportunities

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

6. Simple and Easy

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

7. Readable

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

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

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

10. Portable

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

11. Interpreted

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

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

Advantages of Python Over Other Languages

1. Less Coding

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

2. Affordable

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

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

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build

web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

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

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

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

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

3. Design Restrictions

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

4. Underdeveloped Database Access Layers

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

5. Simple

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

This was all about the Advantages and Disadvantages of Python Programming Language.

History of Python

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners¹, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

Python Development Steps

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido

Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it." Some changes in Python 3:

Print is now a function.

- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

Purpose

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

Python

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

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

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

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

Modules Used in Project

NumPy

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

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

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

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

Pandas

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

Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

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

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

Scikit – learn

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

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

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

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

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code.

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

Install Python Step-by-Step in Windows and Mac

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

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

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet [here](#). The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system

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



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

Step 2: Click on the Download Tab.



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

Looking for a specific release?

Python releases by version number:

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

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

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

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		68111671e5b2db4aef7b9ab01b7079be	23017663	GPG
XZ compressed source tarball	Source release		dE3e4aa46097051c2eca45ee3604803	17131432	GPG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6a28b4fa7583daf1a442cbabcce08e6	34898416	GPG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	3dd603c38217a45773b95ea936b2a1f	20082845	GPG
Windows help file	Windows		063999573a2c98b2ac58ade6b4f7cd2	8131761	GPG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9809c3bf89e3b9afae03184a40729a2	7504291	GPG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a702be4b0ad76d9bdc30c3a583e563400	26883948	GPG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28cb1c90ffbd71aef851a3b8351b4bd2	1362904	GPG
Windows x86 embeddable zip file	Windows		9fab38d18b41879fda9412574139d8	6741626	GPG
Windows x86 executable installer	Windows		33c0c2942e54446ad8d4d147e3b4789	25663848	GPG
Windows x86 web-based installer	Windows		1b670cfafcd117d83c30933ea371d87c	1324608	GPG

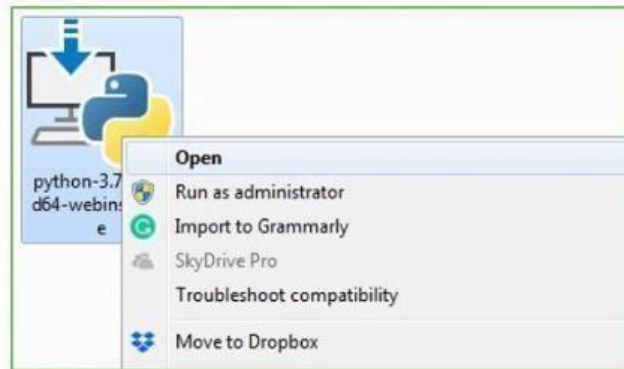
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
- To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file,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

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

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

Installation of Python

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



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



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



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

Note: The installation process might take a couple of minutes.

Verify the Python Installation

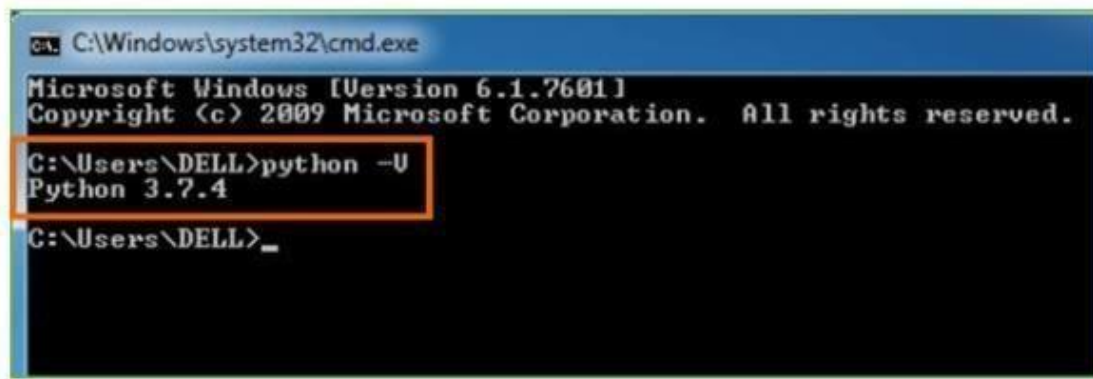
Step 1: Click on Start

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



Step 3: Open the Command prompt option.

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



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

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

C:\Users\DELL>_
```

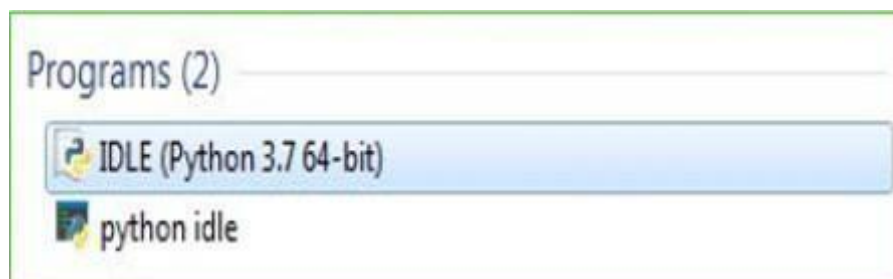
Step 5: You will get the answer as 3.7.4

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

Check how the Python IDLE works

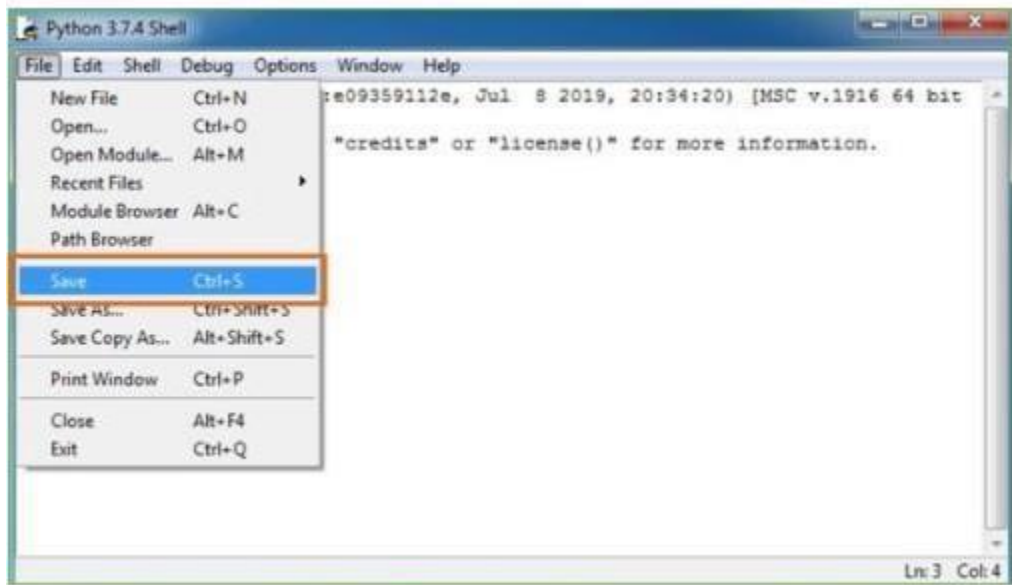
Step 1: Click on Start

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



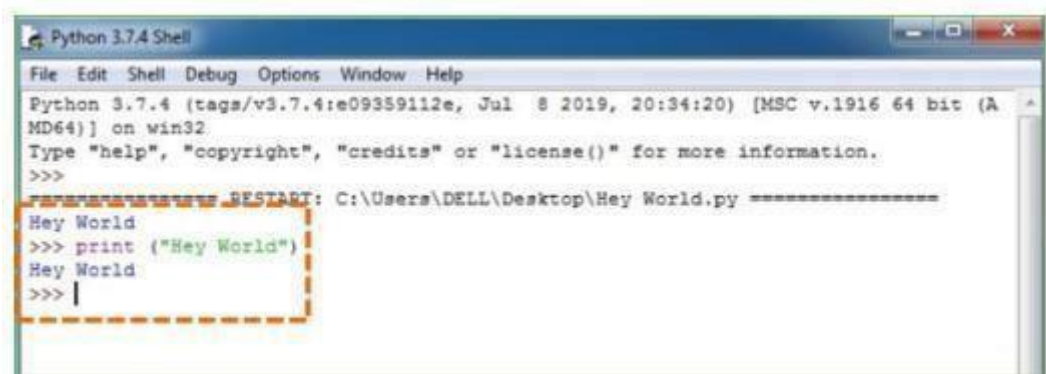
Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

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



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

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



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

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

CHAPTER 7

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupiter (or)
- Google colab

HARDWARE REQUIREMENTS

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- Operating system : Windows, Linux
- Processor : minimum intel i3
- Ram : minimum 4 GB
- Hard disk : minimum 250GB

CHAPTER 8

FUNCTIONAL REQUIREMENTS

OUTPUT DESIGN

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

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

OUTPUT DEFINITION

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

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

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

INPUT DESIGN

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

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

INPUT STAGES

The main input stages can be listed as below:

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

INPUT TYPES

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

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

INPUT MEDIA

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

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

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

ERROR AVOIDANCE

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

ERROR DETECTION

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

DATA VALIDATION

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

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

USER INTERFACE DESIGN

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

USER INTERFACE SYSTEMS CAN BE BROADLY CLASIFIED AS:

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or

displays further information.

USER INITIATED INTERFACES

User initiated interfaces fall into two approximate classes:

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

COMPUTER-INITIATED INTERFACES

The following computer – initiated interfaces were used:

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

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

ERROR MESSAGE DESIGN

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

This application must be able to produce output at different modules for different inputs.

PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the

requirements of the user, is of no use.

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

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

CHAPTER 9

SOURCE CODE

```
# Importing Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.model_selection import train_test_split

import joblib

import os

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Importing Dataset

dataset=pd.read_csv("account_data.csv")

dataset

dataset.info()

dataset.dropna(inplace=True)

dataset.isnull().sum()

dataset.describe()

# Create a count plot

sns.set(style="darkgrid") # Set the style of the plot
```

```

plt.figure(figsize=(8, 6)) # Set the figure size

# Replace 'dataset' with your actual DataFrame and 'Drug' with the column name

ax = sns.countplot(x='IsFraud', data=dataset, palette="Set3")

plt.title("Count Plot") # Add a title to the plot

plt.xlabel("Categories") # Add label to x-axis

plt.ylabel("Count") # Add label to y-axis

# Annotate each bar with its count value

for p in ax.patches:

    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),

                ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),

                textcoords='offset points')

plt.show() # Display the plot

object_cols = dataset.select_dtypes(include=['object']).columns

# Apply Label Encoding to each object column

label_encoders = { }

for col in object_cols:

    le = LabelEncoder()

    dataset[col] = le.fit_transform(dataset[col])

from sklearn.utils import resample

dataset = resample(dataset,

                    replace=True,    # sample with replacement

                    n_samples=40000, # to match majority class

                    random_state=42)

```

```

dataset

X=dataset.iloc[:,0:28]

X

y=dataset.iloc[:,-1]

y

from imblearn.over_sampling import SMOTE

smote = SMOTE()

X,y = smote.fit_resample(X,y)

# Create a count plot

sns.set(style="darkgrid") # Set the style of the plot

plt.figure(figsize=(8, 6)) # Set the figure size

# Replace 'dataset' with your actual DataFrame and 'Drug' with the column name

ax = sns.countplot(x=y, data=dataset, palette="Set3")

plt.title("Count Plot") # Add a title to the plot

plt.xlabel("Categories") # Add label to x-axis

plt.ylabel("Count") # Add label to y-axis

# Annotate each bar with its count value

for p in ax.patches:

    ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),

                ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),

                textcoords='offset points')

plt.show() # Display the plot

X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.20)

X_train.shape

```

```

X_test.shape

y_train.shape

y_test.shape

#Building a ML Model

labels=['Normal','Fraud']

#defining global variables to store accuracy and other metrics

precision = []

recall = []

fscore = []

accuracy = []

#function to calculate various metrics such as accuracy, precision etc

def calculateMetrics(algorithm, predict, testY):

    testY = testY.astype('int')

    predict = predict.astype('int')

    p = precision_score(testY, predict,average='macro') * 100

    r = recall_score(testY, predict,average='macro') * 100

    f = f1_score(testY, predict,average='macro') * 100

    a = accuracy_score(testY,predict)*100

    accuracy.append(a)

    precision.append(p)

    recall.append(r)

    fscore.append(f)

    print(algorithm+' Accuracy  : '+str(a))

    print(algorithm+' Precision : '+str(p))

```

```

print(algorithm+' Recall      : '+str(r))

print(algorithm+' FSCORE      : '+str(f))

report=classification_report(predict, testY,target_names=labels)

print('\n',algorithm+" classification report\n",report)

conf_matrix = confusion_matrix(testY, predict)

plt.figure(figsize =(5, 5))

ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True,
cmap="Blues" ,fmt ="g");

ax.set_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

# # KNN

from sklearn.neighbors import KNeighborsClassifier

if os.path.exists('KNN.pkl'):

    # Load the trained model from the file

    clf = joblib.load('KNN.pkl')

    print("Model loaded successfully.")

    predict = clf.predict(X_test)

    calculateMetrics("KNN", predict, y_test)

else:

    # Train the model (assuming X_train and y_train are defined)

    clf = KNeighborsClassifier()

    clf.fit(X_train, y_train)

```



```

# Save the trained model to a file

joblib.dump(clf, 'KNN.pkl')

print("Model saved successfully.")

predict = clf.predict(X_test)

calculateMetrics("KNN", predict, y_test)

# # RFC Classifier

from sklearn.ensemble import RandomForestClassifier

model_path = 'RandomForestClassifier1.pkl'

# Check if the model exists

if os.path.exists(model_path):

    # Load the trained model from the file

    clf = joblib.load(model_path)

    print("Model loaded successfully.")

else:

    # Train the model

    clf = RandomForestClassifier()

    clf.fit(X_train, y_train)

    # Save the trained model to a file

    joblib.dump(clf, model_path)

    print("Model saved successfully.")

# Make predictions on the test set

predict = clf.predict(X_test)

# Calculate and print metrics

calculateMetrics("Random Forest Classifier Classifier", predict, y_test)

```

```

#showing all algorithms performance values

columns = ["Algorithm Name","Accuracy","Precison","Recall","FScore"]

values = []

algorithm_names = ["KNN", "RFC Classifier"]

for i in range(len(algorithm_names)):

    values.append([algorithm_names[i],accuracy[i],precision[i],recall[i],fscore[i]])

temp = pd.DataFrame(values,columns=columns)

temp

test=pd.read_csv("test1.csv")

test1=pd.read_csv("test1.csv")

test

object_cols = test1.select_dtypes(include=['object']).columns

# Apply Label Encoding to each object column

label_encoders = { }

for col in object_cols:

    le = LabelEncoder()

    test1[col] = le.fit_transform(test1[col])

predict = clf.predict(test1)

# Loop through each prediction and print the corresponding row

for i, p in enumerate(predict):

    if p == 0:

        print(test.iloc[i])

        print("Row      { }:*****NO      Fraud
Detected".format(i))

```

```
else:

    print(test.iloc[i])

    print("Row {}:*****Fraud
Detected".format(i))
```

CHAPTER 10

RESULTS AND DISCUSSION

10.1 Implementation Description

This project is focused on predicting loan eligibility using the SYL Bank dataset. It covers the entire pipeline from data loading and preprocessing to model training, evaluation, and making predictions on new data. The use of both Logistic Regression and Random Forest classifiers allows for a comparison of different algorithms to find the most effective model for predicting loan eligibility.

Here is the implementation description:

1. Importing Libraries:

- Pandas and Numpy: Used for data manipulation and numerical operations.
- Matplotlib and Seaborn: Utilized for data visualization to understand the distribution and characteristics of the dataset.
- Scikit-learn: Provides tools for preprocessing, model training, evaluation metrics, and splitting the data.
- Joblib: Used for saving and loading trained machine learning models.
- OS: Helps in file operations to check if a model file already exists.

2. Loading and Exploring the Dataset:

- Dataset Loading: The dataset is loaded using Pandas.
- Initial Exploration: Basic information such as the first few rows, data types, and summary statistics are displayed. Missing values are addressed, and the dataset is described to understand its basic properties.

3. Data Visualization:

- Histograms and Plots: Visualize distributions of key variables (e.g., income, loan amount) to understand their impact on loan eligibility.

4. Data Preprocessing:

- Handling Missing Values: Address missing values through imputation or dropping rows/columns.
- Feature Engineering: Create new features if necessary, such as calculating loan-to-income ratios or categorizing data.
- Label Encoding: Convert categorical variables into numerical values using label encoding. This step is crucial for preparing the data for machine learning algorithms which require numerical input.

5. Data Splitting:

- Feature and Target Separation: The dataset is split into features (X) and the target variable (y).
- Train-Test Split: The data is further split into training and testing sets with an 80-20 ratio.

6. Model Training and Evaluation:

- Metrics Calculation Function: A custom function `calculateMetrics` is defined to compute and print precision, recall, F1-score, accuracy, classification report, and confusion matrix. This function also visualizes the confusion matrix using a heatmap.
- 6.1 Logistic Regression Model:
 - The code first checks if a pre-trained model (`LogisticRegression.pkl`) exists. If it does, the model is loaded.
 - If the model does not exist, a Logistic Regression classifier is trained on the training data, saved, and then evaluated on the test data using the `calculateMetrics` function.
- 6.2 Random Forest Model:
 - Similarly, the code checks for a pre-trained Random Forest model (`RandomForest.pkl`). If it exists, the model is loaded; otherwise, a new Random Forest classifier is trained, saved, and evaluated using the same function.

7. Comparing Model Performance:

- Performance metrics (accuracy, precision, recall, F1-score) for both models (Logistic Regression and Random Forest) are stored and displayed in a `DataFrame`. This comparison helps in understanding which model performs better on the given dataset.

8. Predicting on New Data:

- **Loading New Data:** A new dataset is loaded for prediction.
- **Preprocessing New Data:** The same label encoding applied to the training data is also applied to the new data.
- **Making Predictions:** The trained model is used to predict loan eligibility on the new data. Each prediction is printed, indicating whether the loan is approved or not.

10.2 Dataset Description

The dataset is a collection of customer data from SYL Bank, used for predicting loan eligibility and potential fraud. Each row in the dataset represents an individual loan application with various attributes describing the applicant's demographic, financial, and behavioral characteristics. The output column `IsFraud` is a binary classification target, indicating whether the application is fraudulent (1) or not (0).

Here's a detailed description of each column in the dataset:

1. **Age:** Age of the applicant.
2. **Occupation:** Job title or profession of the applicant.
3. **MaritalStatus:** Marital status (e.g., Single, Married, Divorced).
4. **Dependents:** Number of dependents the applicant has.
5. **ResidentialStatus:** Type of residence (e.g., Own, Rent, Live with Parents).
6. **AddressDuration:** Duration (in months) the applicant has lived at the current address.
7. **CreditScore:** Credit score of the applicant.
8. **IncomeLevel:** Annual income of the applicant.
9. **LoanAmountRequested:** Amount of loan requested by the applicant.
10. **LoanTerm:** Loan term (in years).
11. **PurposeoftheLoan:** Purpose of the loan (e.g., home, auto, personal).
12. **Collateral:** Indicates whether the applicant has provided collateral (Yes/No).
13. **InterestRate:** Interest rate offered for the loan.

14. **PreviousLoans:** Number of previous loans the applicant has taken.
15. **ExistingLiabilities:** Existing liabilities of the applicant.
16. **ApplicationBehavior:** Speed and nature of the application process (e.g., Rapid, Normal).
17. **LocationofApplication:** Geographic location where the application was made (e.g., Local, Unusual).
18. **ChangeinBehavior:** Indicates if there was a change in applicant behavior during the application process (Yes/No).
19. **TimeofTransaction:** Time at which the transaction was made.
20. **AccountActivity:** Describes the nature of account activity (e.g., Normal, Unusual).
21. **PaymentBehavior:** Describes the payment behavior (e.g., On-time, Defaulted).
22. **Blacklists:** Indicates if the applicant is on any blacklist (Yes/No).
23. **EmploymentVerification:** Indicates if the employment details have been verified (Verified/Not Verified).
24. **PastFinancialMalpractices:** Indicates any past financial malpractices (Yes/No).
25. **DeviceInformation:** Device used for application (e.g., Mobile, Laptop, Tablet).
26. **SocialMediaFootprint:** Indicates the presence of a social media footprint (Yes/No).
27. **ConsistencyinData:** Consistency of the data provided by the applicant (Consistent/Inconsistent).
28. **Referral:** Indicates if the application was made through a referral (Referral/Online).
29. **IsFraud:** Target variable indicating if the application is fraudulent (1) or not (0).

10.3 Results Description

Figure 1 shows the count plot of the output column and Figure 2 shows the count plot after applied smote

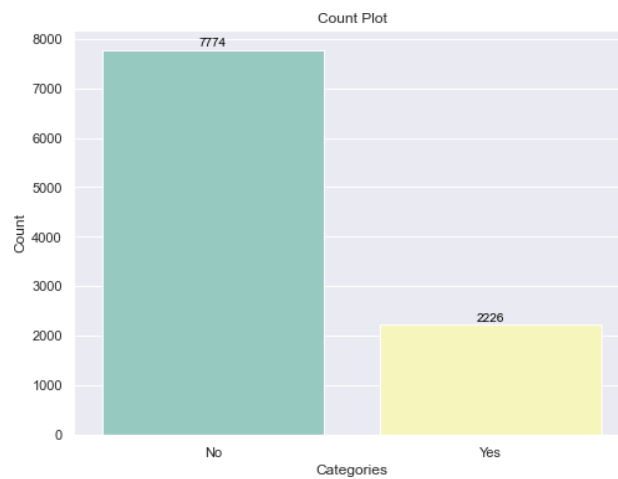


Figure 1: Count plot of isFraud Column

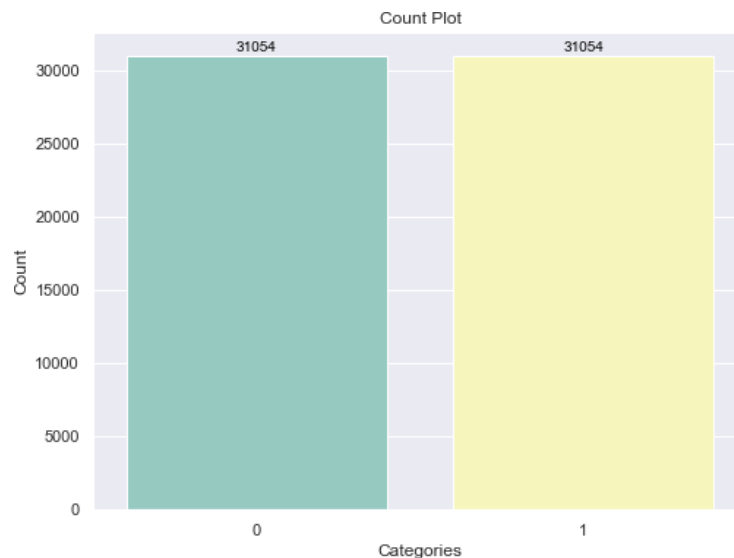


Figure 2: Count plot after applied SMOTE

```
Model loaded successfully.
Random Forest Classifier Classifier Accuracy      : 99.79874416358075
Random Forest Classifier Classifier Precision     : 99.79692486069469
Random Forest Classifier Classifier Recall        : 99.8008539287701
Random Forest Classifier Classifier FSCORE        : 99.79870236498567

Random Forest Classifier Classifier classification report
precision    recall  f1-score   support

   Normal      1.00      1.00      1.00      6130
   Fraud       1.00      1.00      1.00      6292

 accuracy          1.00          1.00          1.00      12422
 macro avg         1.00          1.00          1.00      12422
weighted avg         1.00          1.00          1.00      12422
```

Figure 3: Classification report of RFR

Figure 3 shows that the Classification report of the random forest regressor and accuracy is nearly 100%

- Precision measures the ratio of true positives to the total number of positive predictions. A precision of 1.00 for both normal and fraud classifications means that all the positive predictions the model made were actually correct.
- Recall measures the ratio of true positives to the total number of actual positive cases. A recall of 1.00 for both normal and fraud classifications means that the model identified all of the actual positive cases.
- **F1-Score** is the harmonic mean of precision and recall. A F1-score of 1.00 for both normal and fraud classifications means that the model performed perfectly on both classes.
- Support is the number of actual cases in each class. In this case, there were 6130 normal transactions and 6292 fraudulent transactions.
- **Weighted Avg** is the average of the metrics weighted by the support of each class.
- **Macro Avg** is the unweighted mean of the metrics.

CHAPTER 11

CONCLUSION AND FUTURE SCOPE

11.1 Conclusion

The adoption of machine learning algorithms for loan eligibility prediction, utilizing the SYL Bank dataset, signifies a transformative approach to addressing the challenges faced by the financial sector. This data-driven solution enhances the precision and efficiency of loan assessments, mitigating the risks associated with fraudulent applications. The comprehensive analysis of various features such as age, occupation, marital status, credit score, income level, and past financial behavior enables the development of robust predictive models. These models significantly reduce the reliance on manual processes, thereby minimizing human error, inconsistencies, and biases. By accurately identifying eligible applicants and flagging potential fraud, this approach not only safeguards financial institutions from substantial financial losses but also ensures fair and unbiased lending practices. The implementation of such automated systems promises to streamline the loan approval process, improve customer satisfaction, and uphold the integrity of financial transactions.

11.2 Future Scope

The future scope of loan eligibility prediction and fraud detection using machine learning is vast and promising. Further research can explore the integration of advanced deep learning techniques and ensemble methods to enhance predictive accuracy. Expanding the dataset to include additional features such as social media activity, spending patterns, and real-time transaction data can provide deeper insights and improve model performance. Moreover, implementing explainable AI (XAI) techniques will ensure transparency and trust in automated decision-making processes. The development of adaptive models capable of learning from new data in real-time will enhance the system's robustness and responsiveness to evolving fraud tactics. Collaboration with other financial institutions for data sharing and collective intelligence can create a more comprehensive defense mechanism against fraud. Additionally, regulatory bodies can leverage these advanced predictive models to establish industry-wide standards and best practices for fair lending and fraud prevention.

REFERENCES

- [1] Yoon, Y., Lee, J., Park, E., & Yang, J. (2020). Loan eligibility prediction using machine learning techniques. *Expert Systems with Applications*, 144, 113078. <https://doi.org/10.1016/j.eswa.2019.113078>
- [2] Singh, A., & Yadav, A. (2018). A comparative study of machine learning algorithms for loan eligibility prediction. *International Journal of Computer Applications*, 181(8), 7-11. <https://doi.org/10.5120/ijca2018917309>
- [3] Kumar, A., & Gupta, A. (2019). Predicting loan eligibility using ensemble learning approaches. *Journal of Big Data*, 6(1), 46. <https://doi.org/10.1186/s40537-019-0203-7>
- [4] Kaur, H., & Kaur, P. (2017). Loan approval prediction using data mining techniques: A comparative study. *International Journal of Computer Applications*, 175(9), 13-17. <https://doi.org/10.5120/ijca2017915138>
- [5] Rani, P., & Sharma, M. (2016). A review on loan prediction using data mining techniques. *Procedia Computer Science*, 85, 797-804. <https://doi.org/10.1016/j.procs.2016.05.318>
- [6] Nair, S., & Sindhya, K. (2015). Loan eligibility prediction using decision tree and k-nearest neighbors. *International Journal of Engineering Research and Applications*, 5(4), 57-63.
- [7] Mishra, N., & Verma, V. K. (2018). Loan prediction using neural network and decision tree. *Procedia Computer Science*, 132, 1131-1138. <https://doi.org/10.1016/j.procs.2018.05.203>
- [8] Das, S., & Chakraborty, S. (2020). Loan eligibility prediction using ensemble machine learning techniques. *Journal of King Saud University - Computer and Information Sciences*. Advance online publication. <https://doi.org/10.1016/j.jksuci.2020.04.015>
- [9] Banerjee, I., & Sural, S. (2017). A survey on loan approval prediction using data mining techniques. *International Journal of Computer Applications*, 159(1), 6-12. <https://doi.org/10.5120/ijca2017914124>
- [10] Kumar, S., & Reddy, A. S. (2019). Predicting loan approval using machine learning techniques: A case study. *International Journal of Engineering and Advanced Technology*, 9(1), 2234-2238. <https://doi.org/10.35940/ijeat.A9817.119119>
- [11] Choudhury, D., & Bose, D. (2018). Loan approval prediction using support vector machine and random forest classifiers. *Procedia Computer Science*, 132, 1186-1193. <https://doi.org/10.1016/j.procs.2018.05.211>

- [12] Jain, A., & Kothari, R. (2016). Loan eligibility prediction using decision tree algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering*, 6(7), 204-208.
- [13] Patel, D., & Patel, D. (2017). A survey on loan prediction using machine learning techniques. *International Journal of Innovative Research in Computer and Communication Engineering*, 5(2), 769-773.
- [14] Shukla, A., & Jain, S. (2019). Comparative study of machine learning algorithms for loan approval prediction. *International Journal of Engineering and Advanced Technology*, 8(5C), 1707-1711. <https://doi.org/10.35940/ijeat.F1108.0885C19>
- [15] Ravi, R., & Narasimha Murthy, M. N. (2015). A survey on loan approval prediction using machine learning techniques. *International Journal of Computer Applications*, 113(5), 13-19. <https://doi.org/10.5120/19890-2697>

Project Details				
Academic Year		2024-2025		
Title of the Project		Predicting Loan Defaulters with Machine Learning Models For Credit Card Management		
Name of the Students and Hall Ticket No.		RATHOD KARTHIK (22RA5A0515) BANOTHU MANI (22RA5A0521) BAISKA NAVEEN KUMAR (22RA5A0523) BHURUGUPALLY SURYA PRAKASH (22RA5A0538)		
Name of the Guide		Mr. K. Sunil Kumar		
Project PO Mapping				
Name of Course From which Principles are applied in This Project	Related Course Outcomes Number	Description of the application	Page Number	Attained
Python Programming Software Engineering (C313)	C313.1	Students described the basis for their problem statement.	01	PO2
Machine Learning, Python Programming, Data Mining (C413, C411)	C322.2, C411.2	Students explained about Predicting Loan Defaulters with Machine Learning Models for Credit Card Management	1-2	PO1
Software Engineering, Python Programming (C313)	C313.3	Students identified the existing system and its Drawbacks and proposed a Solution to it.	11-12	P02, P03
Software Engineering (C313)	C313.1	Students identified the Hardware and Software required for the project.	40	PO5
Design Patterns, Software Engineering, DBMS (C313, C322)	C313.2, C222.3	Students explain the flow of the project using UML diagrams designed in STAR UML, ER diagram.	21-26	P03, P05, PO9, PSO3

Python Programming	C413.2, C411.2	Students explained about python programming language and developed code for the problem Statement.	48-53	PO3, PO4, PO5
Data Mining (C411)	C411.2	Students designed the modules for the solution of the problem.	33	PO2, PO3, PO4
Future Scope		Students explained about how they would like to further their project and develop it as their future scope.	61	PO12, PS02
Bibliography		Listed the references from which the literature was collected.	62	PO8, PO12
ENG		Prepared the thesis and intermediate progress reports and explained to the review panel. Also, continuously interact with guide and explain the progress.		PO9, PO10

SIGNATURE OF STUDENT

SIGNATURE OF INTERNAL GUIDE