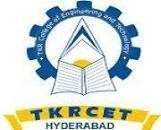
LOAN APPROVAL PREDECATION USING MACHINE LEARNING MODELS



*Submitted in partial fulfillment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

**in**

# Computer Science and Engineering

*by*

# R. GAYATHRI 20K91A05F2

NEHA 20K91A05C6

# MD. ABDUL GAFOOR 20K91A05B2

M. AJAY KUMAR 20K91A05B7

# **Under the guidance of**

## **MS. Y.LATHA**

**ASSISTANT PROFESSOR**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING TKR COLLEGE OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)

(ACCREDITED BY NBA AND NAAC WITH ’A’ GRADE)

**Medbowli, Meerpet, Saroornagar, Hyderabad-500097**

**DECLARATION BY THE CANDIDATE**

We, Ms. **R. GAYATHRI** bearing Hall Ticket Number: **20K91A05F2**, Ms. **NEHA** bearing Hall Ticket Number: **20K91A05C6**, Mr. **MD. ABDUL GAFOOR** bearing Hall Ticket Number: **20K91A05B2**, Mr. **M. AJAY KUMAR** bearing Hall Ticket Number: **20K91A05B7**, hereby declare that the main project report titled **LOAN APPROVAL PREDECATION USING MACHINE LEARNING MODELS** under the guidance of **MS. Y. LATHA**, **Assistant Professor** in Department of Computer Science and Engineering is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

By

# R. GAYATHRI 20K91A05F2

NEHA 20K91A05C6

# MD. ABDUL GAFOOR 20K91A05B2

M. AJAY KUMAR 20K91A05B7

**CERTIFICATE**

This is to certify that the main project report entitled **LOAN APPROVAL** **PREDICTION USING MACHINE LEARNING MODELS**, being submitted by **R. GAYATHRI**, bearing **ROLL.NO: 20K91A05F2**, **NEHA**, bearing **ROLL.NO: 20K91A05C6**, **MD. ABDUL GAFOOR**, bearing **ROLL.NO: 20K91A05B2**, **M. AJAY KUMAR**, bearing **ROLL.NO: 20K91A05B7** in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering, to the TKR College of Engineering and Technology is a record of bonafide work carried out by them under my guidance and supervision.

Name and Signature of the Guide Name and Signature of the HoD Ms. Y. LATHA Dr. A. Suresh Rao

Assistant Professor Professor

## **TABLE OF CONTENTS**

[**ABSTRACT**](#_bookmark0)i

[ACKNOWLEDGEMENTS](#_bookmark0) ii

[LIST OF FIGURES](#_bookmark0) iii

1. [INTRODUCTION](#_bookmark1) 1
   1. [Motivation](#_bookmark2) 1
   2. [Objectives](#_bookmark3) 1
   3. [Existing System](#_bookmark4) 2
   4. [Proposed system](#_bookmark5) 2
2. [LITERATURE REVIEW](#_bookmark6) 3
   1. [Review of Literature](#_bookmark7) 7
3. [SYSTEM ANALYSIS](#_bookmark8) 8
   1. [Feasibility Study](#_bookmark9) 8
      1. [Economical feasibility](#_bookmark10) 8
      2. [Technical feasibility](#_bookmark11) 8
      3. [Social feasibility](#_bookmark12) 8
4. [REQUIREMENT ANALYSIS](#_bookmark13) 9
   1. [Functional Requirements](#_bookmark14) 9
      1. [Hardware Requirements](#_bookmark15) 9
      2. [Software Requirements](#_bookmark16) 9
   2. [Non-Functional Requirements](#_bookmark17) 10
5. [SYSTEM DESIGN](#_bookmark18) 11
   1. [Modules In the System](#_bookmark19) 12
   2. [Flow Graph](#_bookmark20) 12
   3. [UML Diagrams](#_bookmark22) 13
      1. [Class Diagram](#_bookmark23) 14
      2. [Use Case Diagram](#_bookmark25) 15
      3. [Sequence Diagram](#_bookmark27) 16
      4. [Activity Diagram](#_bookmark29) 17
      5. [System Architecture](#_bookmark31) 18
   4. Algorithms 19
      1. Support vector machine 20
      2. Logistic Regression 21
      3. Gaussian NB 23
6. [CODING](#_bookmark33) 24
   1. Pseudo Code 29
   2. Source Code 33
   3. Training Algorithms 36
7. [IMPLEMENTATION and RESULTS](#_bookmark36) 37
   1. [Explanation of Key functions](#_bookmark37) 38
   2. [Method of Implementation](#_bookmark38) 51

7.1.1 [Output Screens](#_bookmark39) 55

* + 1. [Result Analysis](#_bookmark48) 59

1. [TESTING and VALIDATION](#_bookmark49) 60
   1. [Design of Test Cases and Scenarios](#_bookmark50) 60
2. [CONCLUSION and FUTURE SCOPE](#_bookmark53) 62
   1. [Conclusion](#_bookmark54) 62
   2. [Future Scope](#_bookmark55) 62

[REFERENCES](#_bookmark55) 63

[PLAGIARISM REPORT](#_bookmark56) 65

## **ABSTRACT**

A bank's profit or loss is heavily influenced by loans, i.e. whether consumers repay the loans or default on them. By forecasting loan defaulters, the bank can reduce non-performing assets. To demonstrate the improvements, we presented a brief contrast between the existing system and our suggested approach. The system architecture and requirements are discussed in the following slides. Kaggle data is collected for analysis and prediction. The foundation stack for the stacking algorithm in ensemble learning includes models such as logistic regression, SVM, random forest, XG Boost, and AdaBoost. The XG Boost model is used as a metamodel in prediction. The primary goal of this research is to forecast whether giving a loan to a certain person would.

.

## **ACKNOWLEDGEMENTS**

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowned my efforts with success.

We are indebted to the **Internal Guide**, **Ms. Y. Latha**, Assistant Professor, Dept. of Computer Science and Engineering, TKR College of Engineering and Technology, for her support and guidance throughout our Thesis/Dissertation.

We are also indebted to the **Head of the Department**, **Dr. A. Suresh Rao**, Professor, Computer Science and Engineering, TKR College of Engineering and Tech nology, for his support and guidance throughout our Thesis/Dissertation.

We extend my deep sense of gratitude to the **Principal**, **Dr. D. V. Ravi Shankar**, TKR College of Engineering and Technology, for permitting us to undertake this The- sis/Dissertation.

Finally, We express our thanks to one and all that have helped us in successfully completing this Thesis/Dissertation. Furthermore, we would like to thank our family and friends for their moral support and encouragement

BY

R. GAYATHRI 20K91A05F2

NEHA 20K91A05C6

MD. ABDUL GAFOOR 20K91A05B2

M. AJAY KUMAR 20K91A05B7

## **LIST OF FIGURES**

* 1. [Data Flow Diagram](#_bookmark21) 14
     1. [Class diagram](#_bookmark24) 19
     2. [Use Case diagram](#_bookmark26) 20
     3. [Sequence diagram](#_bookmark28) 21

5.3.4 [Activity diagram](#_bookmark30) 22

* + 1. [System Architecture](#_bookmark32) 24
    2. [Interface](#_bookmark40) 32

7.1.1 [Dataset Upload](#_bookmark41) 33

7.2.2 [Preprocessing](#_bookmark42) 34

7.2.3 [Model creation](#_bookmark43) 35

7.2.4 [Output of](#_bookmark44) Approval 36

7.2.5 [Output of](#_bookmark45) Rejection 37

7.2.6 [Accuracy Vs Loss Graph](#_bookmark46) 38

[8.1.1 Test Cases](#_bookmark51) 60

**1.INTRODUCTION**

In recent years, the integration of machine learning algorithms into various domains has revolutionized decision-making processes and improved the efficiency of complex tasks. One such application is the prediction of loan approval using machine learning techniques. Traditional loan approval processes often involve a time-consuming and manual evaluation of numerous factors, leading to delays and sometimes inconsistent decisions. With the advent of machine learning, financial institutions can leverage advanced algorithms to automate and enhance the loan approval process. Machine learning models, such as classifiers, can be trained on historical data to analyze patterns and make predictions about the likelihood of loan approval based on various features. These features may include the applicant's financial history, income, credit score, employment status, and other relevant factors. By incorporating machine learning, financial institutions can achieve faster, more accurate, and unbiased loan approval decisions.

**1.1** [**Motivation**](#_bookmark2)**:**

The aim of implementing machine learning for loan approval prediction is to revolutionize and optimize the traditional lending process by leveraging advanced technologies. The primary goal is to enhance decision-making in financial institutions, making it more efficient, accurate, and transparent. By employing machine learning models, the aim is to automate and expedite the loan approval process while ensuring fairness, reducing risks, and improving overall customer satisfaction.

**1.2 Objectives:**

**Efficiency Improvement:**

Develop machine learning models that streamline and automate the loan approval process, reducing the time and resources required for decision-making.

**Accuracy Enhancement:**

Train machine learning algorithms on historical loan data to create predictive models capable of accurately assessing the creditworthiness of applicants.

**Risk Mitigation:**

Identify and analyze potential risks associated with loan approvals, allowing financial institutions to make more informed decisions and minimize the likelihood of defaults.

**1.3 EXISTING SYSTEM:**

Logistic regression, SVM, Decision tree etc. are all famous classification machine learning models which have been used in the prediction of Loan approval use case in Banking. The accuracy of popular classifiers for a generic dataset for loan approval is:

Logistic Regression: 80%

Ada Boost: 78%

XG Boost: 79%

Random tree forest: 81%

Support Vector Machine: 69%

**1.4 PROPOSED SYSTEM:**

The proposed model of the system is the Ensemble Learning model which combines individual models together to improve the stability and predictive power of the model.

This learning model helps us to use the best qualities of each classifier included which works best for different aspects of the dataset. This augments the precision of the model.

Our ensemble model is divided into two:

1) Base-model stack consisting of Decision Tress, SVM, Linear Regression,

2) Meta-model: Gaussian NB

**2.** [**LITERATURE REVIEW**](#_bookmark6)

In Finance sector, the banking system, banks have a variety of products to provide, but credit lines are their primary source of revenue. As a result, they will profit from the interest earned on the loans they make. Loans, or whether customers repay or default on their loans, affect a bank's profit or loss. The bank's Non-Performing Assets will be reduced before casting loan defaulters. As a result, further investigation into this occurrence is essential. Because precise forecasts are essential for benefit maximization, it's crucial to analyze and compare the various methodologies. The logistic regression model is an important predictive analytics tool for detecting loan defaulters. In order to assess and forecast, data from kagle is acquired. Logistic Regression models were used to calculate the various performance indicators. The models are compared using performance metrics like sensitivity and specificity. In addition to checking account details (which indicate a customer's wealth), the model is significantly better because it includes variables (customer personal attributes such as age, objective, credit score, credit amount, credit period, and so on) that should be considered when correctly calculating the probability of loan default. As a result, using a logistic regression approach, the appropriate clients to target for loan issuance can be easily identified by evaluating their plausibility of loan default. The model implies that a bank should assess a creditor's other attributes, which play a critical role in credit decisions and forecasting loan defaulters, in addition to giving loans to wealthy borrowers.

**Title: Customer Loan eligibility Prediction using ML**

**Published by: A Shaik, KS Asritha, and N Lahre**

In the contemporary banking landscape, predicting customer loan eligibility is a crucial task due to the significant contribution of loan revenue to a bank's profit. Manual assessment of credit risk is time-consuming and lacks assurance regarding borrower security. To address this challenge, this project leverages machine learning models for accurate loan eligibility predictions. The proposed methodology involves developing an automated loan prediction platform using thirteen key factors such as gender, education, income, and credit history. Five machine learning algorithms — Random Forest Classifier, Passive Aggressive Classifier, Multinomial Naïve Bayes, Support Vector Classifier, and Adaboost Classifier — are applied to the dataset. The process includes user registration, data collection, analysis, preprocessing, splitting, training, testing, and prediction. Performance metrics such as accuracy, precision, and recall are used to evaluate algorithm efficacy. Results analysis reveals the Random Forest Classifier outperforms others, achieving a 78% accuracy rate, indicating its effectiveness in classifying loan eligibility. Visualizations including line charts, bar graphs, and a pie chart compare accuracies across all algorithms. This machine learning-based approach offers efficient and accurate prediction of loan eligibility, benefiting both banks and customers by providing insights into developing robust loan prediction systems for the banking industry.

**Title: ML based Loan Approval Prediction System A Novel Approach**

**Published by: MKJ Kannan, and AR Nithej**

The paper titled "ML Based Loan Approval Prediction System: A Novel Approach" delves into the critical role of loan approval systems within banks, stressing the necessity to minimize losses and ensure loans are granted solely to eligible customers capable of repayment. While previous studies have shown promising results, the authors seek to enhance accuracy further. To achieve this, the study explores a diverse array of machine learning algorithms, including Logistic Regression, Decision Tree, Random Forest, K Nearest Neighbors, Artificial Neural Network, Naive Bayes, Adaboost, and Voting Classifier, for predicting loan approval.

The proposed algorithm utilizes a comprehensive selection of machine learning techniques tailored for loan approval prediction. These encompass Logistic Regression, Decision Tree, Random Forest Classifier, K-Nearest Neighbor, Naïve Bayes, Adaboost, Linear SVM, Polynomial SVM, and Wavelet SVM. Each algorithm is succinctly described, emphasizing its application in loan approval prediction. The methodology encompasses model training using pertinent datasets, preprocessing, and the application of these algorithms to predict loan approval.

Simulation results are presented to showcase the performance of various machine learning algorithms in loan approval prediction. The comparative analysis underscores the superior performance of the proposed study, with accuracy rates of 86%, 74%, 86%, and 86% achieved for Logistic Regression, Decision Tree, Support Vector Machine (RBF), and Naive Bayes, respectively. The study concludes by accentuating the potential efficiency and accuracy enhancements the proposed system offers to the loan approval process.

**Title: Analysis of Loan Availability using ML Techniques**

**Published by: Sharayu Dosalwar, Ketki Kinkar, and Rahul Sannat**

The project is centered on utilizing machine learning techniques to analyze loan availability, particularly within the banking system, with the overarching aim of developing a predictive model for identifying potential loan defaulters. By doing so, the project seeks to mitigate a bank's Non-Performing Assets and bolster profit margins. Central to this endeavor is the implementation of the logistic regression model as a pivotal tool for predictive analytics. The dataset utilized for analysis is obtained from Kaggle, encompassing variables such as checking account details, customer personal attributes, age, objective, credit score, credit amount, and credit period.

To assess and compare the performance of various machine learning models in predicting loan availability, the research employs a range of models including Logistic Regression, Support Vector Machine (SVM), Decision Trees, Random Forest, Linear Models (LM), XGBoost Classifier, K-Nearest Neighbors (KNN), and Naive Bayes. Each model is evaluated based on accuracy scores, and a comparative analysis is conducted. The logistic regression model emerges as particularly noteworthy, attributed to its consideration of a comprehensive set of variables, leading to enhanced accuracy in predicting loan outcomes. Furthermore, the study presents a block diagram illustrating the different phases of the prediction process, encompassing data cleaning, processing, and model testing on test data.

The experimental analysis reveals that the Logistic Regression model attains the highest accuracy (0.785) among the various machine learning models considered, indicating its effectiveness in identifying potential loan defaulters, taking into account both checking account details and customer personal attributes. The research underscores the importance for banks to not solely rely on a customer's wealth but also consider other characteristics in credit decisions and predicting loan defaulters. The model's accuracy in forecasting loan availability renders it a promising tool for the banking industry, offering a reliable and efficient means of evaluating loan eligibility.

**Title: ML Techniques For Recognizing the Loan Eligibilty**

**Published by: Mr. Abhiroop Sarkar**

Our study focuses on effectively automating loan eligibility forecasts using machine learning algorithms such as logistic regression, decision trees, and random forests. This program seeks to overcome the labor-intensive and costly nature of manual loan assessments by creating a predictive model that quickly determines loan acceptance based on key criteria such as gender, marital status, education, and income. Logistic regression has the highest mean validation accuracy (80.78%), making it ideal for binary outcome predictions such as loan acceptance. Decision trees provide useful insights into decision rules that influence loan eligibility, albeit with slightly lower accuracy (70.51%) than logistic regression. Random forest, using an ensemble of decision trees, yields a competitive mean validation accuracy of 79.79%, making it a viable option for reliable loan prediction. This project

**Title: ML Algorithm to predict Fradulent Loan Requests**

**Published by: Nazmul Hasan, Tanvir Anzum, Tareq Hasan, and Nusrat jahan**

The study's goal is to create a viable predictive model for detecting fraudulent loan requests in the banking sector using machine learning techniques. With an increasing volume of loan applications, banking institutions have a considerable issue in differentiating legitimate from fraudulent ones. This study looks at six machine learning algorithms—Decision Tree, Support Vector Machine (SVM), Random Forest, K-Nearest Neighbors (KNN), Ada-Boost, and Logistic Regression—to predict fraudulent loan applications. Data obtained from an online platform includes numerous loan request attributes, which are visualized, preprocessed, and model implemented throughout the algorithms. Model performance is assessed using evaluation criteria like as accuracy, precision, F1-score, recall, and support. Notably, the K-Nearest Neighbors (KNN) algorithm emerges as the most successful, with an accuracy of 83.75% and greater precision.

**Title: Loan Default Identification and its effect**

**Published by: Gopal Choudhary, Yash Garud, Akshil Shetty, Rumit Kadakia, and Sonali Borase**

The study "Loan Default Identification and its Effect" addresses the pressing issue of loan defaults in the banking industry by offering a machine learning method to improve loan approval processes. As the number of loan applications increases, banks must precisely identify legitimate consumers who will repay their loans. The authors use massive datasets of previous loan data to prepare for model training, with a focus on data analysis and cleansing. To enhance the dataset, they use feature selection approaches, null value imputation, and outlier treatment. The characteristics include loan amount, interest rate, credit history, and loan status. The machine learning model predicts loan defaults using logistic regression and random forest techniques, with a reported 61% accuracy. Furthermore, the return on investment (ROI).

**3. SYSTEM ANALYSIS**

**3.1 Feasibility Study**

In this phase, the analysis of the project's feasibility is conducted, and a business proposal is presented along with a basic project plan and cost estimates. During system analysis, it is crucial to assess the feasibility of the proposed system to ensure it does not burden the company. To carry out feasibility analysis, a clear understanding of the major system requirements is necessary.

**3.1.1 Economic Feasibility**

This assessment examines the economic impact of the system on the organization. Due to limited research and development funds, expenditures must be justified. The developed system must stay within budget, which was achieved by leveraging mostly freely available technologies and purchasing only customized products.

**3.1.2 Technical Feasibility**

This evaluation focuses on the technical requirements of the system. The system should not excessively strain available technical resources, which would burden the client. It is essential that the developed system has modest technical requirements, necessitating minimal or no changes for implementation.

**3.1.3 Social Feasibility**

This aspect of the study assesses the level of user acceptance of the system. It involves training users to efficiently utilize the system and ensuring they perceive it as a necessity rather than a threat. User acceptance primarily depends on the methods employed to educate and familiarize users with the system. Building user confidence is crucial, enabling them to provide constructive criticism, as they are the ultimate users of the system.

**4.REQURIMENT ANALYSIS**

**4.1 FUNCTIONAL REQURIMENT:**

4.1.1 Hardware:

1. OS – Windows 7,8 or 10 (32 or 64 bit)
2. RAM – 4GB

4.1.2 Software:

* Python
* Anaconda Navigator
* Python built-in modules
* Numpy
* Pandas
* Matplotlib
* Sklearn
* Seaborn

**4.2 NON-FUNCTIONAL REQURIMENTS:**

**Performance**: How fast the system can process loan applications and make predictions accurately.  
**Scalability**: The ability to handle increasing amounts of data and users without sacrificing performance.  
**Reliability**: Ensuring the system is dependable and operates consistently under different conditions.  
**Security**: Safeguarding sensitive data, such as personal and financial information, from unauthorized access or breaches.

**Maintainability**: How easily the system can be updated, fixed, or enhanced over time without disrupting its operations.

**5.SYSTEM DESIGN**

**5.1 MODULES**

**1. DATA COLLECTION**

**2. DATA PREPARATION**

**3. FEATURE ENGINEERING**

**4. MODEL EVALUATION**

**DATA COLLECTION**

This stage involves selecting a subset of available data for analysis. Machine learning tasks typically begin with labeled data, where the target answer is known.

**DATA PREPARATION**

Prepare the selected data by formatting, cleaning, and sampling. Common pre-processing steps include formatting the data into a suitable format, cleaning to handle missing or sensitive information, and sampling to manage large datasets.

**FEATURE ENGINEERING**

Feature extraction is a process of reducing attributes, while feature selection involves ranking existing attributes based on their predictive significance. The transformed attributes, or features, are combinations of the original attributes. The data is then used to train classifiers, such as the ones provided by the Natural Language Toolkit library in Python.

**MODEL EVALUATION**

Model evaluation is crucial for determining the best-performing model that represents the data effectively. Evaluating model performance with separate testing data helps avoid overfitting. Performance metrics, such as accuracy, are used to assess the classification models. The results are typically visualized to provide insights into the classified data.

Proposed Approach Summary:

1. Utilize historical loan approval records.

2. Filter the dataset based on analysis requirements to create a refined dataset.

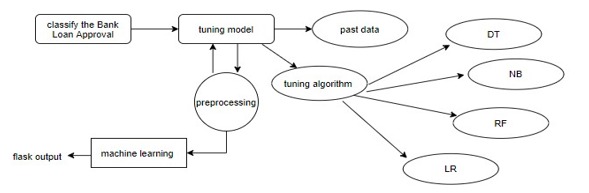
3. Perform pre-processing on the dataset.

4. Split the data into training and testing sets.

5. Train the model with the training data and evaluate its performance using the testing dataset.

6. Assess accuracy metrics to measure the model's effectiveness.

**5.2 Flow chart**

****

**5.3 UML diagrams**

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated more or less formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system’s architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

Building blocks of UML:

The vocabulary of the UML encompasses three kinds of building blocks:

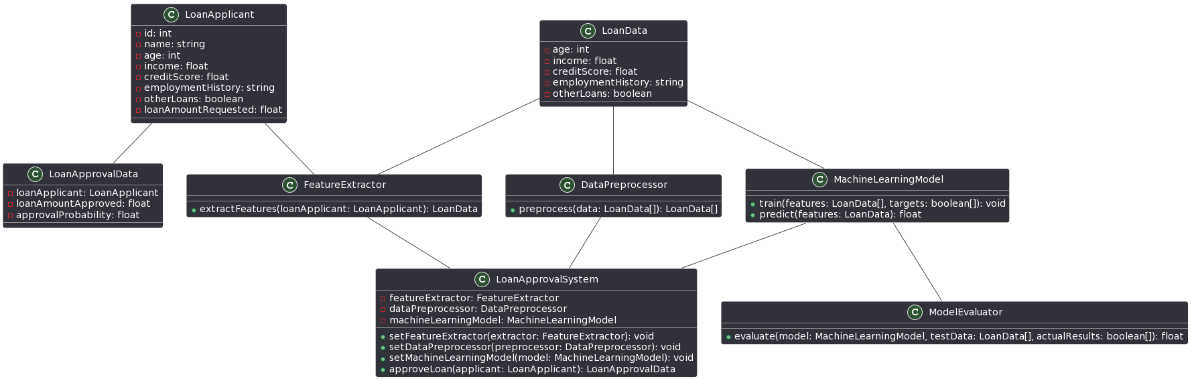
* Things
* Relationships
* Diagrams

Things in UML:

There are four kinds of things in the UML:

* Structural things
* Behavioral things
* Grouping things

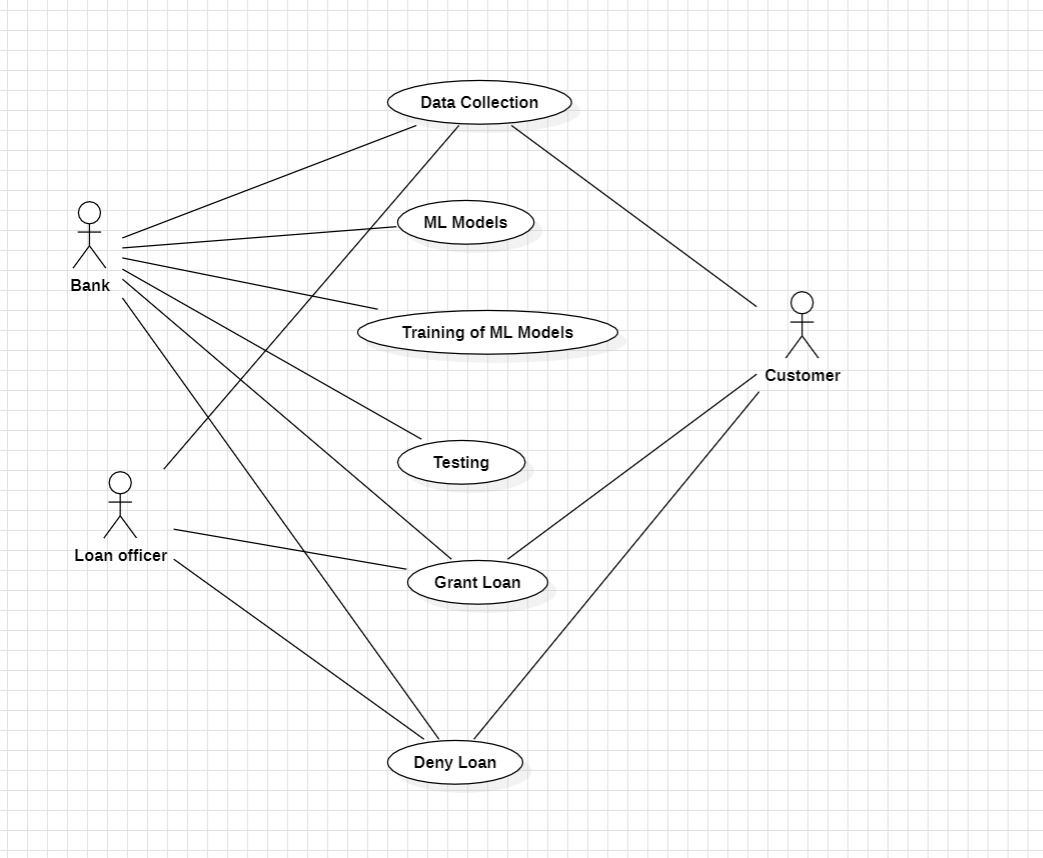
**5.3.1 Class Diagram:**



***Fig 5.3.1*** Class Diagram

A class diagram in UML is an important tool for representing the structure of object-oriented systems. It represents classes as rectangles, with their characteristics and operations contained within compartments. Relationships between classes, such as affiliations, aggregations, compositions, and inheritances, are represented by lines that connect them, indicating how classes interact. Multiplicity specifies the cardinality of associations, which indicates the number of instances involved. Dependencies show how changes in one class might affect another. Class diagrams serve as visual blueprints that help developers comprehend, communicate, and create software systems successfully. They provide a clear visual representation of class hierarchies, relationships, and behaviors, directing the implementation process. Class diagrams, which provide a full understanding of the system's design, improve communication among the team's developers and ensure the successful production.

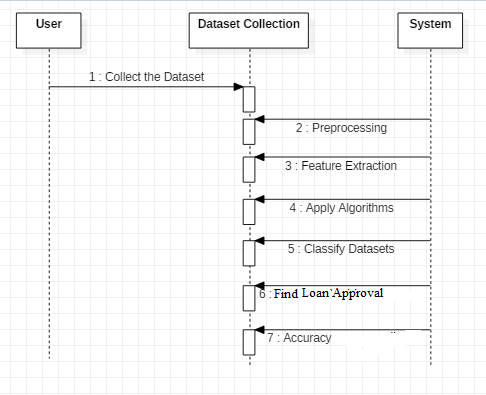
**5.3.2 Use Case Diagram:**



***Fig 5.3.2*** Use Case Diagram

A use case diagram in Unified Modelling Language (UML) is a visual representation of the interactions between users or systems and a system under different scenarios. It helps in capturing and communicating the functional requirements of a system. For example, in a banking system scenario, actors like customers, bank tellers, and administrators interact with use cases like account login, balance inquiries, fund transfers, and account management. The diagram's system boundary encloses all actors and use cases, providing a clear delineation of the system's scope. Use case diagrams aid in understanding user-system interactions, facilitating requirement analysis, system design, and communication among stakeholders and development teams.

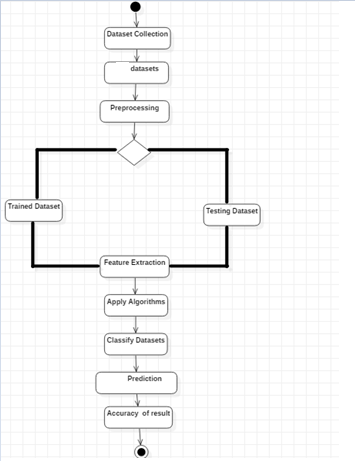
**5.3.3 Sequence Diagram:**



***Fig 5.3.3*** Sequence Diagram

Unified Modeling Language (UML) sequence diagrams show how messages move and interact with one another over time amongst objects or components in a system. It shows the communications that were sent back and forth between various entities in order to accomplish a specific function. For instance, a sequence diagram could show how messages are sent and received between users, servers, and the database in a messaging application scenario. Horizontal arrows between each vertical line, which represents an object or system participant, indicate messages and exchanges. The precise procedures or actions being called as well as their sequential order are described in these exchanges. It is also possible to indicate timing restrictions, such how long each message exchange lasts. Sequence diagrams are very helpful in comprehending how a system behaves dynamically.

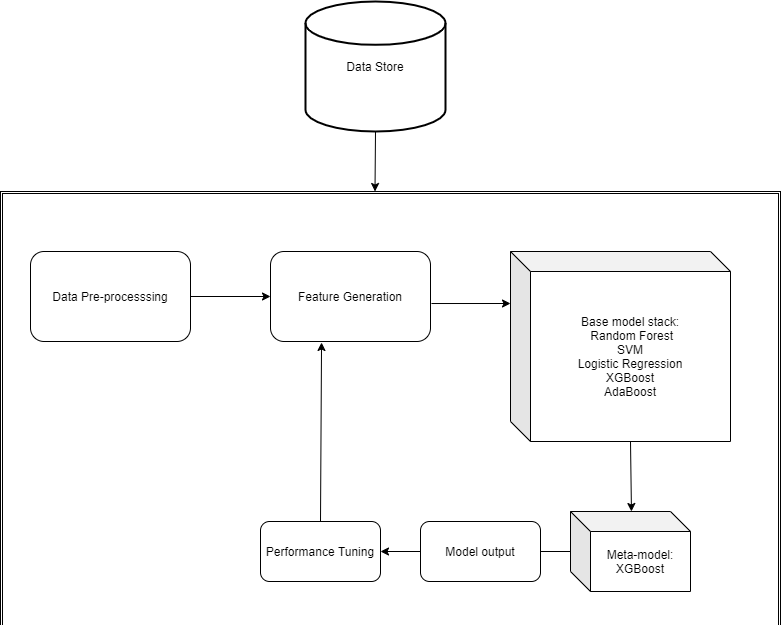
**5.3.4 Activity Diagram:**



***Fig 5.3.4*** Activity Diagram

Unified Modeling Language (UML) activity diagrams highlight the order of actions and decision points while graphically depicting the control flow inside a system or process. Modeling workflows, business processes, and system behaviors are some of its most helpful applications. An activity diagram might, for example, show the stages involved in making a purchase in an online shopping scenario. These phases could include product browsing, adding things to the basket, checking out, and finishing the transaction. A rounded rectangle representing each activity is connected by arrows that show how control moves between them. Diamond-shaped decision points divide the flow according to parameters or requirements. Joins and forks show synchronization points and concurrent activity, respectively. Activity diagrams assist stakeholders in comprehending the behavior of the system, locating possible bottlenecks or inefficiencies, and improving

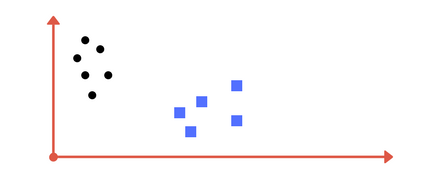
**5.3.5 System Architecture:**



***Fig 5.3.5*** System Architecture

**5.4 ALGORITHMS**

**5.4.1 SUPPORT VECTOR MACHINE**

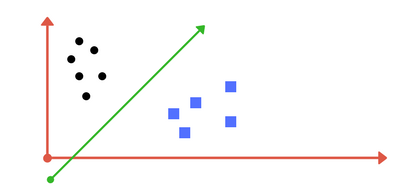


***Fig 5.4.1*** SUPPORT VECTOR MACHINE

A Support Vector Machine (SVM) is a type of discriminative classifier that is formally defined by a separating hyperplane. In simpler terms, when provided with labeled training data (a form of supervised learning), SVM outputs an optimal hyperplane that effectively divides new examples into distinct categories. In a two-dimensional space, this hyperplane manifests as a line that splits the plane into two parts, with each class positioned on either side.

Imagine you're presented with a plot displaying two distinct classes labeled on a graph (as shown in image A). Your task is to determine a separating line for these classes. You might arrive at a solution resembling the illustration in image B, where the line effectively divides the two classes. Any point to the left of the line belongs to the black circle class, while any point to the right belongs to the blue square class. This delineation of classes is precisely what SVM accomplishes—it identifies a line or hyperplane (in higher-dimensional spaces) that distinctly separates the classes. Later, we'll delve into why the concept extends to multidimensional spaces.

**5.4.2 LOGISTIC REGRESSION:**



***Fig 5.4.2*** LOGISTIC REGRESSION

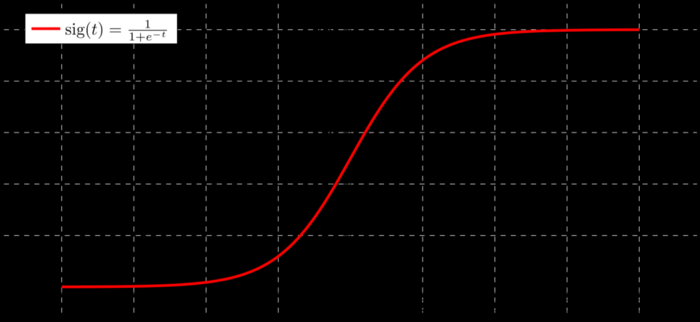
Logistic Regression has its origins in early twentieth-century biological sciences before being applied in various social science domains. It is particularly useful when the dependent variable, also known as the target, is categorical. For instance, it is commonly employed in scenarios such as predicting whether an email is spam (1) or not (0), or determining if a tumor is malignant (1) or benign (0).

In a classification problem, linear regression isn't ideal because it requires setting a threshold for classification. For example, if a tumor is actually malignant but the predicted continuous value is 0.4, and the threshold value is set at 0.5, the data point may be incorrectly classified as not malignant, potentially leading to serious consequences.

This highlights the inadequacy of linear regression for classification tasks, as it is unbounded. Logistic regression, on the other hand, is bounded and ensures that the predicted values strictly fall within the range of 0 to 1.

In Simple Logistic Regression, the model outputs either 0 or 1, and the hypothesis function is defined using the sigmoid function, ensuring that the output remains within the range of 0 to 1.

**5.4.3 Gaussian NB:**

**

***Fig 5.4.3*** Gaussian NB

Gaussian Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem, assuming that features are independent and follow a Gaussian (normal) distribution. It is particularly well-suited for classification tasks where the features are continuous and can be modeled using a normal distribution**.**

**Advantages:**

Simple and Fast: Gaussian NB is a simple and computationally efficient algorithm. It is easy to implement and well-suited for large datasets with a high dimensionality.

Good for High-Dimensional Data: Performs well in high-dimensional spaces, making it suitable for applications with a large number of features.

Stable Performance: Often exhibits stable performance across a variety of datasets, especially when the assumption of feature independence is reasonable.

Handles Missing Data: Can handle missing values in the data without the need for imputation, as it estimates the parameters of the Gaussian distribution independently for each feature.

Low Training Time: Requires minimal training time compared to more complex algorithms. It is a good choice for quick prototyping and baseline classification.

Effective in Text Classification: Gaussian NB is particularly effective in text classification tasks, such as spam filtering and document categorization.

**Disadvantages:**

Assumption of Feature Independence:

One of the main limitations is the assumption of feature independence, which may not hold true in all real-world scenarios. This could impact the model's accuracy, especially when features are correlated.

Sensitivity to Outliers:

Gaussian NB assumes that features follow a Gaussian distribution, making it sensitive to outliers. Outliers can significantly affect the estimated mean and standard deviation, impacting the model's performance.

Limited Expressiveness: Due to its simplicity and assumption of feature independence, Gaussian NB may not capture complex relationships within the data, limiting its expressiveness compared to more sophisticated models.

Difficulty with Categorical Data: Gaussian NB is designed for continuous features and may not perform well with categorical features. It may require additional preprocessing

**6.CODING**

**6.1 Pseudo Code:**

1. Import necessary libraries:

- Import libraries such as pandas, numpy, and scikit-learn for data manipulation, numerical operations, and machine learning algorithms, respectively.

2. Load the loan dataset:

- Load the dataset containing information about loan applicants, including features such as income, credit score, loan amount, employment history, etc., and the target labels indicating whether the loan was approved or not.

3. Preprocess the data:

- Handle missing values:

- Impute missing values in numerical features with the mean or median value.

- Encode categorical variables:

- Convert categorical variables into numerical format using techniques such as one-hot encoding or label encoding.

- Normalize or scale numerical features:

- Scale numerical features to have a similar scale using techniques like Standard Scaler to prevent features with larger magnitudes from dominating the learning process.

4. Split the dataset into training and testing sets:

- Split the dataset into training and testing sets to evaluate the performance of the model.

- Typically, you might use an 80-20 or 70-30 split, where 80% or 70% of the data is used for training and the remaining for testing.

5. Choose a machine learning algorithm for classification:

- Choose a suitable classification algorithm such as Logistic Regression, Decision Trees, Random Forest, or Gradient Boosting Machines (GBM) for predicting loan approval status based on applicant information.

6. Train the classification model:

- Fit the chosen classification algorithm to the training data to learn the patterns and relationships between the features and the loan approval status.

7. Evaluate the model:

- Predict loan approval status on the testing set using the trained model.

- Calculate evaluation metrics such as accuracy, precision, recall, F1-score, and/or area under the ROC curve (AUC-ROC) to assess the performance of the model.

- These metrics provide insights into how well the model is performing in terms of correctly predicting loan approvals and rejections.

8. Optionally, fine-tune hyperparameters:

- Fine-tune hyperparameters of the classification algorithm to optimize model performance.

- Use techniques like grid search or random search to search for the best combination of hyperparameters.

9. Deploy the loan approval prediction model:

- Once satisfied with the performance, deploy the trained model to predict loan approvals for new loan applications.

- Integrate the model into the loan approval system to automate the decision-making process.

10. Continuously update and improve the model:

- Monitor the model's performance over time and update it as necessary with new data or retraining to maintain its accuracy and reliability.

- Incorporate feedback from loan decisions and adjust the model accordingly to adapt to changing trends and patterns in loan applications.

**Example:**

# Step 1: Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

# Step 2: Load the loan dataset

loan\_data = pd.read\_csv('loan\_dataset.csv')

# Step 3: Preprocess the data

# Handle missing values

loan\_data.fillna(0, inplace=True) # Filling missing values with 0 for simplicity

# Encode categorical variables (if any)

loan\_data = pd.get\_dummies(loan\_data)

# Normalize or scale numerical features

scaler = StandardScaler()

loan\_data[['income', 'loan\_amount', 'credit\_score']] = scaler.fit\_transform(loan\_data[['income', 'loan\_amount', 'credit\_score']])

# Step 4: Split the dataset into training and testing sets

X = loan\_data.drop('approval\_status', axis=1)

y = loan\_data['approval\_status']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Step 5: Choose a machine learning algorithm for classification

model = LogisticRegression()

# Step 6: Train the classification model

model.fit(X\_train, y\_train)

# Step 7: Evaluate the model

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred))

# Step 8: Optionally, fine-tune hyperparameters

# (not implemented in this example, but you can use techniques like grid search or random search)

# Step 9: Deploy the loan approval prediction model

# Once satisfied with the performance, deploy the model to predict loan approvals for new loan applications

# Example: new\_loan\_application = pd.read\_csv('new\_loan\_application.csv')

# prediction = model.predict(new\_loan\_application)

# Step 10: Continuously update and improve the model

# Monitor model performance and update it as necessary with new data or retraining to enhance prediction accuracy and reliability

**6.2 Source Code**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

**### Using Pandas Loading a Dataset**

df=pd.read\_csv("train\_loanPrediction.csv")

df.head()

**### Pre Processing**

type(df)

df.shape

df.describe()

df.isnull().sum()

df1=df.dropna()

df1.isnull().sum()

type(df1)

val=df1.index

df1['ApplicantIncome'].hist(bins=10)

plt.title("Application Income")

plt.xlabel('Amount')

plt.ylabel('No.of App')

df['CoapplicantIncome'].hist(bins=10)

plt.title("CoApplication Income")

plt.xlabel('Amount')

plt.ylabel('No.of CoApp')

df['LoanAmount'].hist(bins=10)

plt.title("Loan Amount")

plt.xlabel('Amount')

plt.ylabel('No.of CoApp')

**#combining the above 2 plots in a stacked chart**

stack\_chart = pd.crosstab(df['Credit\_History'], df['Loan\_Status'])

stack\_chart.plot(kind='bar', stacked=True, color=['green','red'], grid=False)

stack\_chart = pd.crosstab(df['Gender'], df['Loan\_Status'])

stack\_chart.plot(kind='bar', stacked=True, color=['green','red'], grid=False)

df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']

df['TotalIncome\_log'] = np.log(df['TotalIncome'])

df['LoanAmount'].hist(bins=10)

plt.xlabel('Amount')

df2=df1.drop(["Loan\_ID"],axis=1)

**### LabelEncoder**

from sklearn import preprocessing

le=preprocessing.LabelEncoder()

df2['LGender'] = le.fit\_transform(df2['Gender'].values.reshape(-1,1).ravel())

df2.head()

df2['LMarried']=le.fit\_transform(df2['Married'].values.reshape(-1,1).ravel())

df2.head()

df2['LSelf\_Employed']=le.fit\_transform(df2["Self\_Employed"].values.reshape(-1,1).ravel())

df2['LLoan\_Status']=le.fit\_transform(df2["Loan\_Status"].values.reshape(-1,1).ravel())

df2['LEducation']=le.fit\_transform(df2["Education"].values.reshape(-1,1).ravel())

df2['LProperty\_Area']=le.fit\_transform(df2["Property\_Area"].values.reshape(-1,1).ravel())

df2.head()

df2=df2.drop(["Self\_Employed"],axis=1)

df2=df2.drop(["Loan\_Status"],axis=1)

df2=df2.drop(["Property\_Area"],axis=1)

df2=df2.drop(["Married"],axis=1)

df2=df2.drop(["Gender"],axis=1)

df2=df2.drop(["Education"],axis=1)

df2.head()

**### Classification**

x=df2.iloc[:,df2.columns !='LLoan\_Status']

y=df2.iloc[:,df2.columns =='LLoan\_Status']

x.head()

y.head()

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test=train\_test\_split(x,y,test\_size=0.3)

x\_train.shape

x\_train

y\_train

y\_test.shape

np1=df2.values

#plt.subplot(1,2,1)

sns.countplot(x="Gender", order=['Male','Female'], data=df, palette="Set1")

plt.show()

#plt.subplot(1,2,1)

sns.countplot(x="Married", order=['Yes','No'], data=df, palette="Set1")

plt.show()

#plt.subplot(1,2,1)

sns.countplot(x="Property\_Area", order=['Urban','Rural','Semiurban'], data=df, palette="Set1")

plt.show()

#plt.subplot(1,2,1)

sns.countplot(x="Education", order=['Graduate','Not Graduate'], data=df, palette="Set1")

plt.show()

**6.3 Training the Algorithams**

**#### 1.SVM-Support Vector Machine**

from sklearn import svm

from sklearn import metrics

model1=svm.SVC()

model1.fit(x\_train,y\_train)

xpredict=model1.predict(x\_test)

svm=metrics.accuracy\_score(y\_test,xpredict)

print(svm)

svm=model1.score(x\_test,y\_test)

svm

**### 2.LogisticRegression**

from sklearn.linear\_model import LogisticRegression

model2=LogisticRegression()

model2.fit(x\_train,y\_train)

xpredict=model2.predict(x\_test)

lr=metrics.accuracy\_score(y\_test,xpredict)

print(lr)

**### 3.Decision Tree algorithm**

from sklearn.tree import DecisionTreeClassifier

model3=DecisionTreeClassifier()

model3.fit(x\_train,y\_train)

xpredict=model3.predict(x\_test)

dt=metrics.accuracy\_score(y\_test,xpredict)

print(dt)

**# Naive Bayes**

from sklearn.naive\_bayes import GaussianNB

model4=GaussianNB()

model4.fit(x\_train,y\_train)

xpredict=model4.predict(x\_test)

nv=metrics.accuracy\_score(y\_test,xpredict)

print(dt)

**# Graph\_Comparision**

import matplotlib.pyplot as plt; plt.rcdefaults()

objects = ('Support Vector',' LogisticRegression','Decision Tree' , 'Naive Bayes')

y\_pos = np.arange(len(objects))

performance = [svm,lr,dt,nv]

plt.bar(y\_pos, performance, align='center', alpha=0.5)

plt.xticks(y\_pos, objects)

plt.ylabel('Accuracy')

plt.title('SVM vs LogisticRegression vs Decision Tree vs Naive Bayes')

plt.show()

import pickle

pickle.dump(model2, open('model.pkl','wb'))

**7 IMPLEMENTATION AND RESULT**

**7.1** [**Explanation of Key functions**](#_bookmark37)

**MACHINE LEARNING:**

Machine Learning refers to a system capable of learning from examples without explicit programming by developers, leading to self-improvement. This innovation lies in the concept that a machine can autonomously learn from data to generate accurate outcomes.

By combining data with statistical techniques, machine learning predicts outputs used by businesses to derive actionable insights. It shares connections with data mining and Bayesian predictive modeling, with machines processing input data through algorithms to generate responses.

A common machine learning application involves providing recommendations, as seen in platforms like Netflix, where movie or series suggestions are based on users' viewing histories. Tech companies leverage unsupervised learning to enhance user experiences by tailoring recommendations.

Beyond recommendations, machine learning finds utility in various tasks such as fraud detection, predictive maintenance, portfolio optimization, and task automation.

Contrasting with traditional programming, where programmers code all rules based on logical foundations, machine learning eliminates the need for explicit rule-writing. In traditional programming, as systems grow complex, more rules must be added, making maintenance increasingly challenging.

**COMPUTER**

DATA RULES

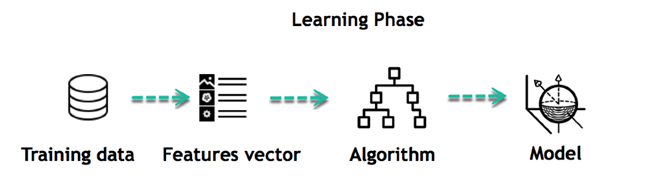
***Fig 6.2.1***

**How does Machine learning work?**

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the learning and inference. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the data. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a feature vector. You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a model. Therefore, the learning stage is used to describe the data and summarize it into a model.

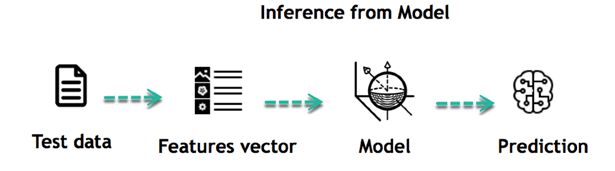


***Fig 6.2.2*** Learning phase

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

#### Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.



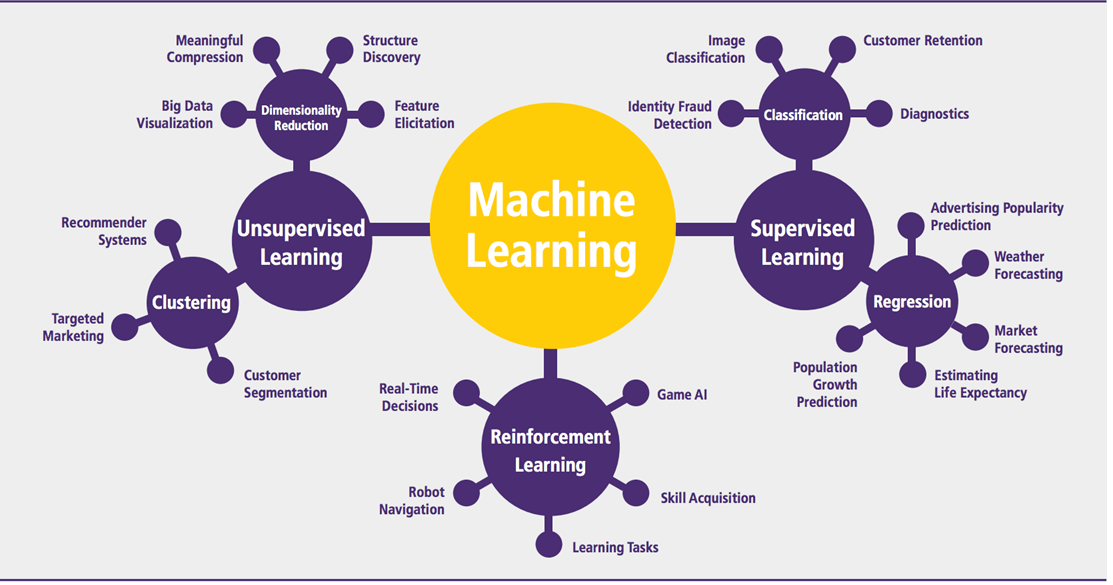
***Fig 6.2.3*** Inference from model

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

## Machine learning Algorithms and where they are used?



***Fig 6.2.4***

Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

#### Supervised learning

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

* Classification task
* Regression task

#### Classification

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customers, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learns how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

#### Regression

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of features like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

|  |  |  |
| --- | --- | --- |
| Algorithm Name | Description | Type |
| Linear regression | Finds a way to correlate each feature to the output to help predict future values. | Regression |
| Logistic regression | Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors) | Classification |
| Decision tree | Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a final decision output is made | Regression Classification |
| Naive Bayes | The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event. | Regression Classification |
| Support vector machine | Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divides the classes. It is best used with a non-linear solver. | Regression (not very common) Classification |
| Random forest | The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction. | Regression Classification |
| AdaBoost | Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome | Regression Classification |
| Gradient-boosting trees | Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous trees and tries to correct it. | Regression Classification |

#### Unsupervised learning

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

|  |  |  |
| --- | --- | --- |
| Algorithm | Description | Type |
| K-means clustering | Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans) | Clustering |
| Gaussian mixture model | A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters | Clustering |
| Hierarchical clustering | Splits clusters along a hierarchical tree to form a classification system.  Can be used for Cluster loyalty-card customer | Clustering |
| Recommender system | Help to define the relevant data for making a recommendation. | Clustering |
| PCA/T-SNE | Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances. | Dimension Reduction |

**Application of Machine learning**

**Augmentation:**

* Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

**Automation:**

* Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

**Finance Industry**

* Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

**Government organization**

* The government makes use of ML to manage public safety and utilities. Take the example of China with its massive face recognition. The government uses Artificial intelligence to prevent jaywalkers.

**Healthcare industry**

* Healthcare was one of the first industries to use machine learning with image detection.

**Marketing**

Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers developed advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, the marketing department relies on AI to optimize the customer relationship and marketing campaign.

Example of application of Machine Learning in Supply Chain

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In the past year the stock manager relied extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In terms of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

Example of Machine Learning Google Car

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

**ANACONDA NAVIGATOR**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, Mac OS and Linux.

## **Why use Navigator?**

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

## **WHAT APPLICATIONS CAN I ACCESS USING NAVIGATOR?**

The following applications are available by default in Navigator:

* JupyterLab
* Jupyter Notebook
* QTConsole
* Spyder
* VSCode
* Glueviz
* Orange 3 App
* Rodeo
* RStudio

Advanced conda users can also build your own Navigator applications

## How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

## What’s new in 1.9?

* Add support for Offline Mode for all environment related actions.
* Add support for custom configuration of main windows links.
* Numerous bug fixes and performance enhancements.

**PYTHON OVERVIEW**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

* 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 is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include:

* Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* Easy-to-read: Python code is more clearly defined and visible to the eyes.
* Easy-to-maintain: Python's source code is fairly easy-to-maintain.
* A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* Databases: Python provides interfaces to all major commercial databases.
* GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

* IT supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Python’s standard library

* Pandas
* Numpy
* Sklearn
* seaborn
* matplotlib
* Importing Datasets

**PANDAS**

Pandas is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in [data munging/wrangling](https://en.wikipedia.org/wiki/Data_wrangling) if not THE most used one. Pandas is an open source

What’s cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). This is so much easier to work with in comparison to working with lists and/or dictionaries through for loops or list comprehension.

**Installation and Getting Started**

In order to “get” Pandas you would need to install it. You would also need to have Python 2.7 and above as a pre-requirement for installation. It is also dependent on other libraries (like [NumPy](http://www.numpy.org/)) and has optional dependencies (like Matplotlib for plotting). Therefore, I think that the easiest way to get Pandas set up is to install it through a package like the [Anaconda distribution](https://www.continuum.io/downloads), “a cross platform distribution for data analysis and scientific computing.”

In order to use Pandas in your Python IDE ([Integrated Development Environment](https://en.wikipedia.org/wiki/Integrated_development_environment)) like [Jupyter Notebook](http://jupyter.org/) or [Spyder](https://pythonhosted.org/spyder/) (both of them come with Anaconda by default), you need to import the Pandas library first. Importing a library means loading it into the memory and then it’s there for you to work with. In order to import Pandas all you have to do is run the following code:

* import pandas as pd
* import numpy as np

Usually you would add the second part (‘as pd’) so you can access Pandas with ‘pd.command’ instead of needing to write ‘pandas.command’ every time you need to use it. Also, you could import numpy as well, because it is a very useful library for scientific computing with Python. Now Pandas is ready for use! Remember, you would need to do it every time you start a new Jupyter Notebook, Spyder file etc.

Working with Pandas

Loading and Saving Data with Pandas

When you want to use Pandas for data analysis, you’ll usually use it in one of three different ways:

* Convert a Python’s list, dictionary or Numpy array to a Pandas data frame
* Open a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc
* Open a remote file or database like a CSV or a JSONon a website through a URL or read from a SQL table/database

There are different commands to each of these options, but when you open a file, they would look like this:

* pd.read\_filetype()

As I mentioned before, there are different file types Pandas can work with, so you would replace “filetype” with the actual, well, filetype (like CSV). You would give the path, filename etc inside the parenthesis. Inside the parenthesis you can also pass different arguments that relate to how to open the file. There are numerous arguments and in order to know all of them, you would have to read the documentation (for example, the [documentation for pd.read\_csv()](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)would contain all the arguments you can pass in this Pandas command).

In order to convert a certain Python object (dictionary, lists etc) the basic command is:

* pd.DataFrame()

Inside the parenthesis you would specify the object(s) you’re creating the data frame from. This command also has [different arguments](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html).

You can also save a data frame you’re working with/on to different kinds of files (like CSV, Excel, JSON and SQL tables). The general code for that is:

* df.to\_filetype(filename)

Viewing and Inspecting Data

Now that you’ve loaded your data, it’s time to take a look. How does the data frame look? Running the name of the data frame would give you the entire table, but you can also get the first n rows with df.head(n) or the last n rows with df.tail(n). df.shape would give you the number of rows and columns. df.info() would give you the index, data type and memory information. The command s.value\_counts(dropna=False) would allow you to view unique values and counts for a series (like a column or a few columns). A very useful command is df.describe() which inputs summary statistics for numerical columns. It is also possible to get statistics on the entire data frame or a series (a column etc):

* df.mean() Returns the mean of all columns
* df.corr() Returns the correlation between columns in a data frame
* df.count() Returns the number of non-null values in each data frame column
* df.max()Returns the highest value in each column
* df.min()Returns the lowest value in each column
* df.median()Returns the median of each column
* df.std()Returns the standard deviation of each column

Selection of Data

One of the things that is so much easier in Pandas is selecting the data you want in comparison to selecting a value from a list or a dictionary. You can select a column (df[col]) and return column with label col as Series or a few columns (df[[col1, col2]]) and returns columns as a new DataFrame. You can select by position (s.iloc[0]), or by index (s.loc['index\_one']) . In order to select the first row you can use df.iloc[0,:] and in order to select the first element of the first column you would run df.iloc[0,0] . These can also be used in different combinations, so I hope it gives you an idea of the different selection and indexing you can perform in Pandas.

Filter, Sort and Groupby

You can use different conditions to filter columns. For example, df[df[year] > 1984] would give you only the column year is greater than 1984. You can use & (and) or | (or) to add different conditions to your filtering. This is also called boolean filtering.

It is possible to sort values in a certain column in an ascending order using df.sort\_values(col1) ; and also in a descending order using df.sort\_values(col2,ascending=False). Furthermore, it’s possible to sort values by col1 in ascending order then col2 in descending order by using df.sort\_values([col1,col2],ascending=[True,False]).

The last command in this section is groupby. It involves splitting the data into groups based on some criteria, applying a function to each group independently and combining the results into a data structure. df.groupby(col) returns a groupby object for values from one column while df.groupby([col1,col2]) returns a groupby object for values from multiple columns.

Data Cleaning

Data cleaning is a very important step in data analysis. For example, we always check for missing values in the data by running pd.is null() which checks for null Values, and returns a boolean array (an array of true for missing values and false for non-missing values). In order to get a sum of null/missing values, run pd. Is null().sum(). Pd .not null() is the opposite of pd. Is null(). After you get a list of missing values you can get rid of them, or drop them by using df. Drop na() to drop the rows or df. drop na(axis=1) to drop the columns. A different approach would be to fill the missing values with other values by using df. Fill na(x) which fills the missing values with x (you can put there whatever you want) or s .fill na(s.mean()) to replace all null values with the mean (mean can be replaced with almost any function from the statistics section).

It is sometimes necessary to replace values with different values. For example, s. replace(1,'one') would replace all values equal to 1 with 'one'. It’s possible to do it for multiple values: s. replace([1,3],['one', 'three'])would replace all 1 with 'one' and 3 with 'three'. You can also rename specific columns by running:  df. rename(columns={'old\_name': 'new\_ name'})or use df. set\_ index('column\_one') to change the index of the data frame.

Join/Combine

The last set of basic Pandas commands are for joining or combining data frames or rows/columns. The three commands are:

* df1.append(df2)— add the rows in df1 to the end of df2 (columns should be identical)
* df. concat([df1, df2],axis=1) — add the columns in df1 to the end of df2 (rows should be identical)
* df1.join(df2,on=col1,how='inner') — SQL-style join the columns in df1with the columns on df2 where the rows for col have identical values. how can be equal to one of: 'left', 'right', 'outer', 'inner'

**NUMPY**

Numpy is one such powerful library for array processing along with a large collection of high-level mathematical functions to operate on these arrays. These functions fall into categories like Linear Algebra, Trigonometry, Statistics, Matrix manipulation, etc.

Getting NumPy

NumPy’s main object is a homogeneous multidimensional array. Unlike python’s array class which only handles one-dimensional array, NumPy’s nd array class can handle multidimensional array and provides more functionality. NumPy’s dimensions are known as axes. For example, the array below has 2 dimensions or 2 axes namely rows and columns. Sometimes dimension is also known as a rank of that particular array or matrix.

#### Importing NumPy

NumPy is imported using the following command. Note here np is the convention followed for the alias so that we don't need to write numpy every time.

* import numpy as np

NumPy is the basic library for scientific computations in Python and this article illustrates some of its most frequently used functions. Understanding NumPy is the first major step in the journey of machine learning and deep learning.

**Sk learn**

In python, scikit-learn library has a pre-built functionality under sk learn. Pre-processing.

Next thing is to do feature extraction Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally our models are trained using the Classifier algorithm.. We use nltk . classify module on Natural Language Toolkit library on Python. We use the labeled dataset gathered . The rest of our labeled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre processed data. The chosen classifiers were Decision tree , Support Vector Machines and Random forest. These algorithms are very popular in text classification tasks.

**SEABORN**

# Data Visualization in Python

Data visualization is the discipline of trying to understand data by placing it in a visual context, so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

To get a little overview here are a few popular plotting libraries:

* [Matplotlib:](https://matplotlib.org/) low level, provides lots of freedom
* [Pandas Visualization:](https://pandas.pydata.org/pandas-docs/stable/visualization.html) easy to use interface, built on Matplotlib
* [Seaborn:](https://seaborn.pydata.org/) high-level interface, great default styles
* [ggplot:](http://ggplot.yhathq.com/) based on R’s ggplot2, uses [Grammar of Graphics](https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448)
* [Plotly:](https://plot.ly/python/) can create interactive plots

In this article, we will learn how to create basic plots using Matplotlib, Pandas visualization and Seaborn as well as how to use some specific features of each library. This article will focus on the syntax and not on interpreting the graphs.

Matplotlib

Matplotlib is the most popular python plotting library. It is a low level library with a Matlab like interface which offers lots of freedom at the cost of having to write more code.

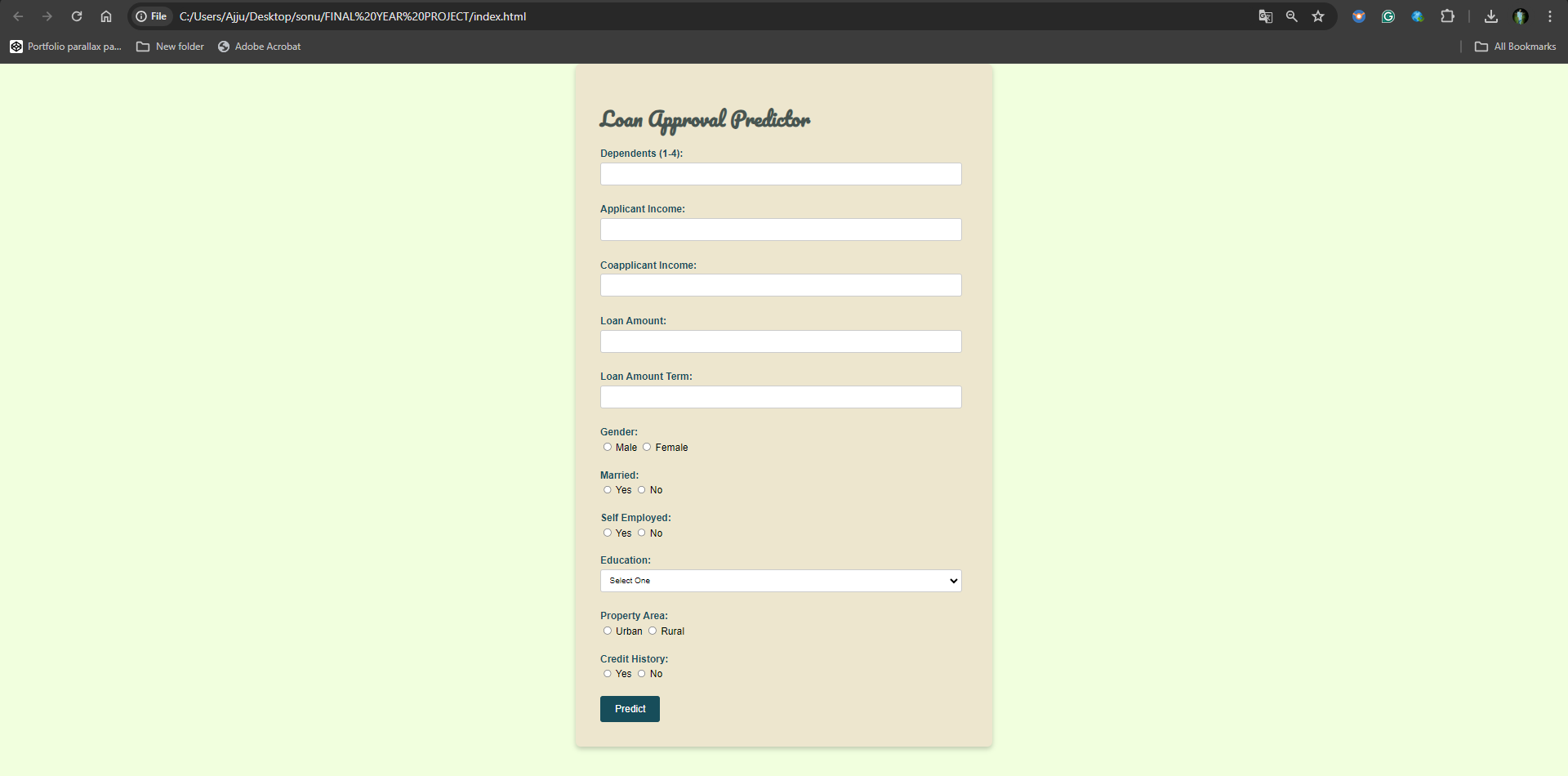
1. To install Matplotlib, pip anaconda can be used.
2. pip install matplotlib
3. conda install matplotlib

Matplotlib is specifically good for creating basic graphs like line charts, bar charts, histograms and many more. It can be imported by typing:

* import matplotlib.pyplot as plt

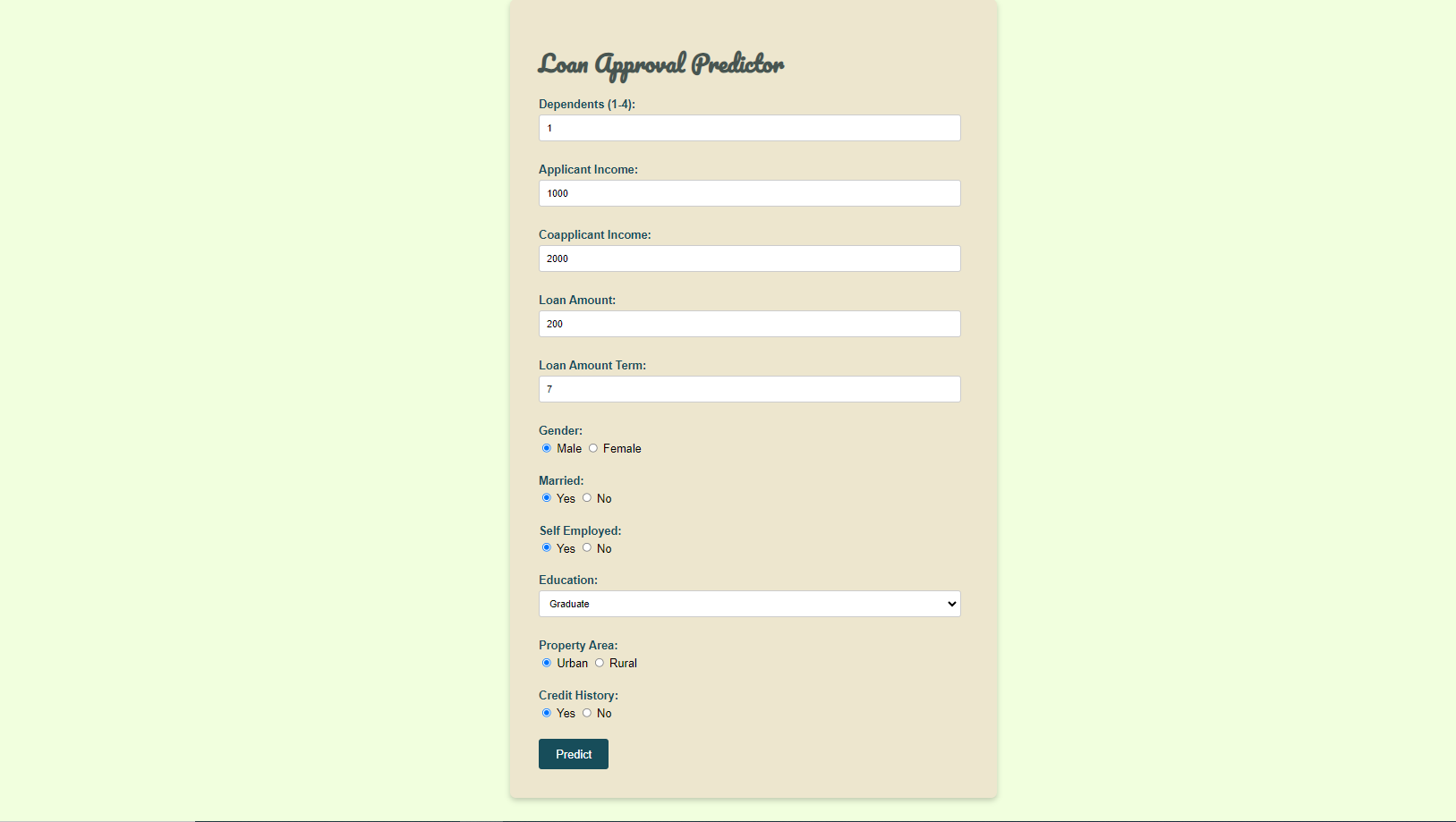
**7.2 IMPLEMENTATION AND RESULT**

**Web page outlook:**



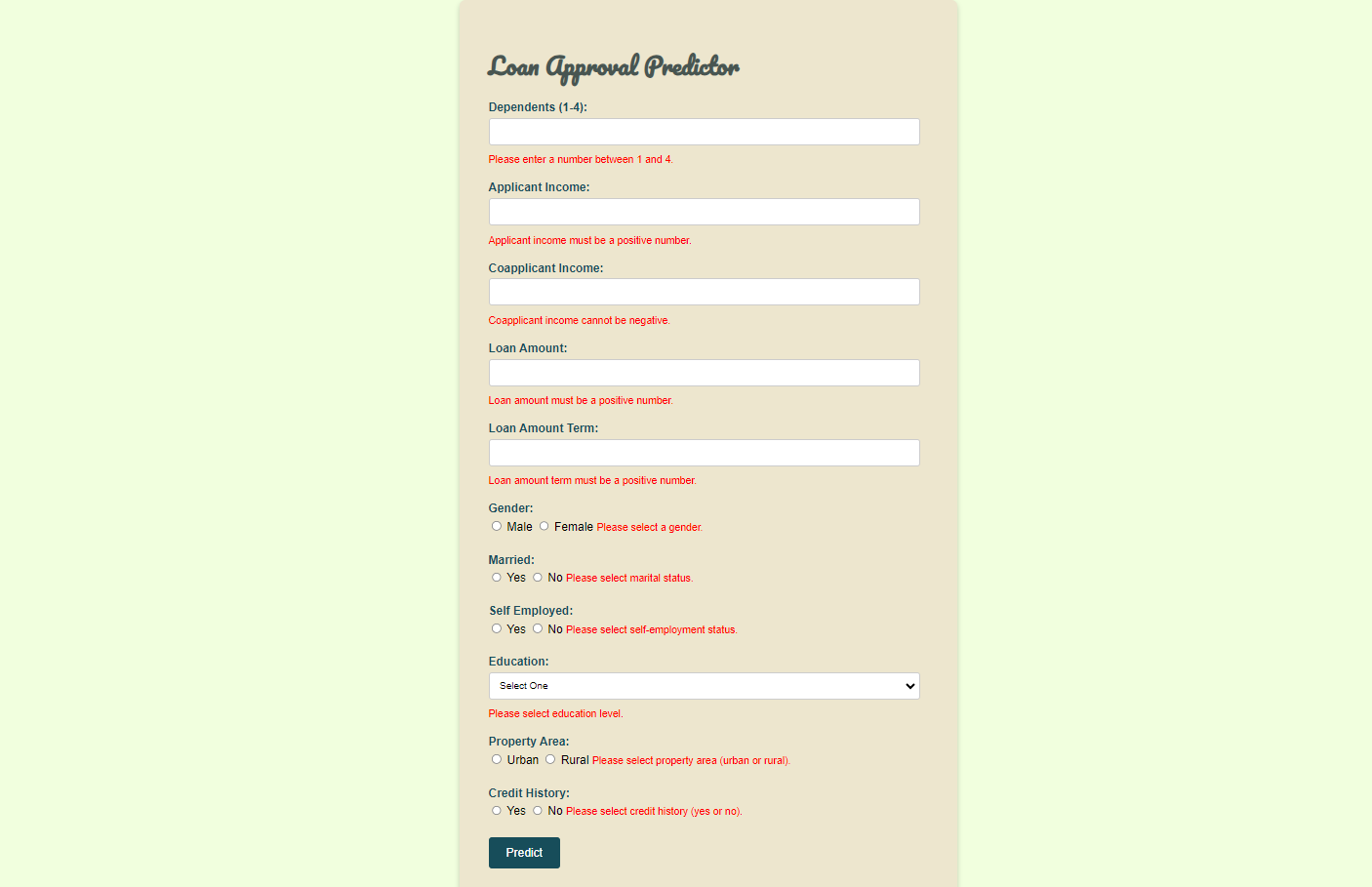
***Fig 7.2.1*** Web page outlook

**After entering data:**



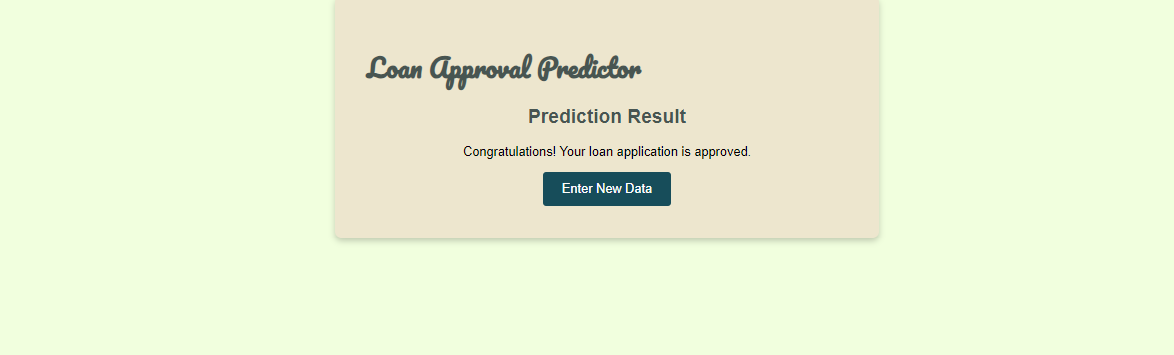
***Fig 7.2.2*** After entering data

**Invalid data:**



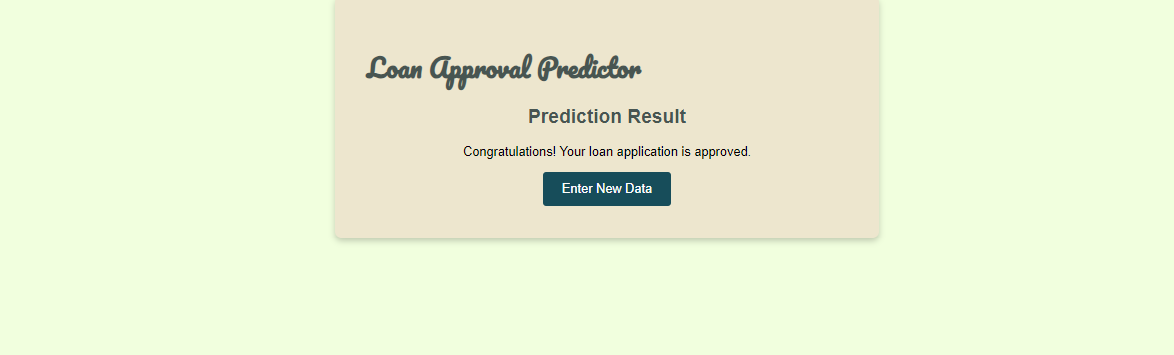
***Fig 7.2.3*** Invalid data

**Loan approved:**



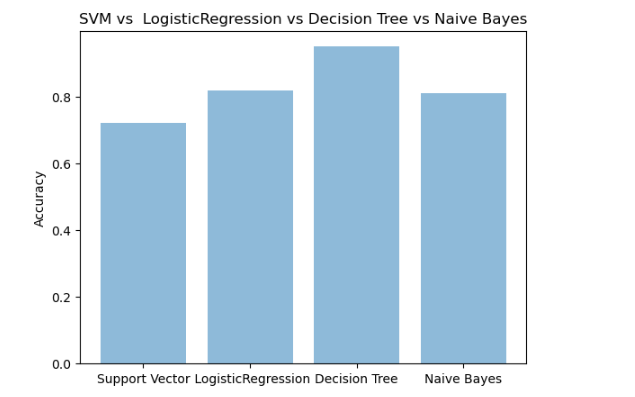
***Fig 7.2.4*** Loan approved

**Loan not approved:**



***Fig 7.2.5*** Loan not approved

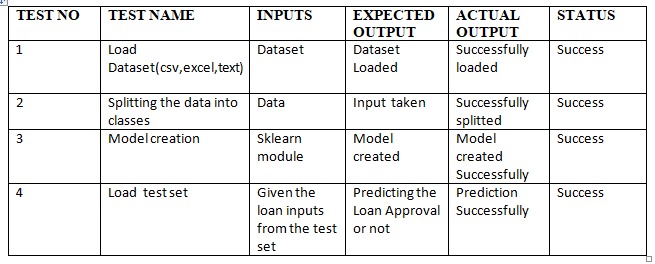
**7.2 Result Analysis**



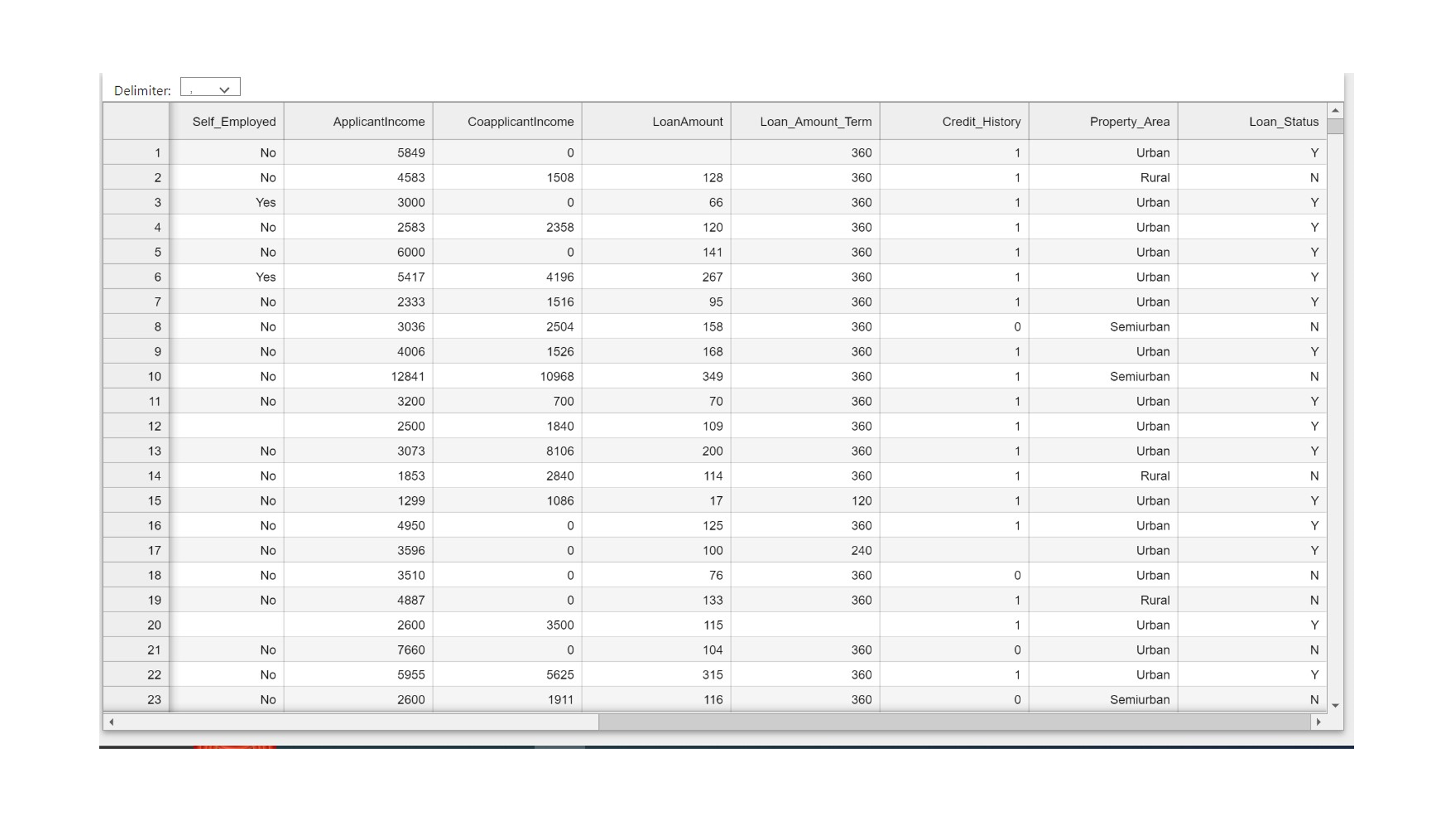
***Fig 7.2.6***

**8.** [**Design of Test Cases and Scenarios**](#_bookmark50)**:**

**8.1 TEST CASES**

****

*Fig 8.1.1*



*Fig 8.1.2*

**9.Conclusion and Future Scope**

**9.1 Conclusion:**

The analytical process started from data cleaning and processing, Missing value imputation with packages, then exploratory analysis and finally model building and evaluation. The best accuracy on a public test set is 0.841 achieved the the Stacking Classifier using Ensemble Learning. This brings some of the following insights about approval. Applicants with Credit history not passing fails to get approved, Probably because they have a probability of not paying back. Most of the Time, Applicants with high income sanctioning low amounts are more likely to get approved which make sense, more likely to pay back their loans. Some basic characteristic gender and marital status seems not to be taken into consideration by the company.

**9.2 Future Enhancement:**

The future scope for loan approval prediction using machine learning is promising, and several trends and advancements are anticipated to shape this field. Here are some key areas of future scope:

**Enhanced Predictive Models:**

Continuous improvements in machine learning algorithms and techniques will lead to more sophisticated predictive models. Advanced models, such as deep learning architectures, may be explored to extract intricate patterns from large and complex datasets.

**Explainability and Interpretability:**

The need for transparent and interpretable machine learning models is gaining importance, especially in financial decision-making. Future research may focus on developing models that provide clear explanations for their predictions, ensuring regulatory compliance and user trust.

**Integration of Alternative Data:**

Inclusion of non-traditional data sources, such as social media activity, online behavior, and other alternative data points, to enhance the accuracy of credit risk assessment. Integrating a broader range of features can provide a more comprehensive view of an individual's financial health.

**References:**

[1] Ethem Alpaydin, “Introduction to Machine Learning 3e (Adaptive Computation and Machine Learning Series)”, Third Edition, MIT Press, 2014 .

[2] Stephen Marsland, “Machine Learning – An Algorithmic Perspective”, Second Edition, Chapman and Hall/CRC Machine Learning and Pattern Recognition Series, 2014

[3] Tom M Mitchell, “Machine Learning”, First Edition, McGraw Hill Education, 2013.

[4] GUO, G., WANG, H., BELL, D., BI, Y. & GREER, K., “KNN model-based approach in classification”. On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. Springer.

[5] Min-Chun Yang, Chiun-Sheng Huang, “Whole breast lesion detection using naive bayes classifier for portable ultrasound”, elsevier, computer science section, vol. 38, No.11, pp. 1870-1880, 2012.

[5] Abedi, R., Costache, R., Shafizadeh-Moghadam, H., et al. (2022). Flash-flood susceptibility mapping based on XGBoost, random forest and boosted regression trees. \*Geocarto Int.\*, 37(19), 5479–5496.

[7] Butaru, F., Chen, Q., Clark, B., et al. (2016). Risk and risk management in the credit card industry. \*J. Bank. Finance\*, 72(Nov.), 218–239.

[8] Chen, Z., Jin, S., Liu, R., et al. (2021). A deep non-negative matrix factorization model for big data representation learning. \*Front. Neurorob.\*, 15(Jul.), 701194.

[9] Dalmau, R., Ballerini, F., Naessens, H., et al. (2021). An explainable machine learning approach to improve take-off time predictions. \*J. Air Transport. Manag.\*, 95(Aug.), 102090.

[10] Deng, T. (2019). Study of the prediction of micro-loan default based on Logit model. In: \*2019 International Conference on Economic Management and Model Engineering (ICEMME)\*. IEEE, pp. 260–264.

[11] Emekter, R., Tu, Y., Jirasakuldech, B., et al. (2015). Evaluating credit risk and loan performance in online Peer-to-Peer (P2P) lending. \*Appl. Econ.\*, 47(1), 54–70.

[12] Fitzpatrick, T., Mues, C. (2016). An empirical comparison of classification algorithms for mortgage default prediction: evidence from a distressed mortgage market. \*Eur. J. Oper. Res.\*, 249(2), 427–439.

[13] Fuster, A., Goldsmith-Pinkham, P., Ramadorai, T., et al. (2022). Predictably unequal? The effects of machine learning on credit markets. \*J. Finance\*, 77(1), 5–47.

[14] Hao, X., Zhang, Z., Xu, Q., et al. (2022). Prediction of f-CaO content in cement clinker: a novel prediction method based on LightGBM and Bayesian optimization. \*Chemometr. Intell. Lab. Syst.\*, 220(Jan.), 104461.

[15] Huang, Y., Rameezdeen, R., Chow, C.W., et al. (2022). Monitoring the health status of water mains using a scorecard modelling approach. \*Water Supply\*, 22(3), 3114–3124.

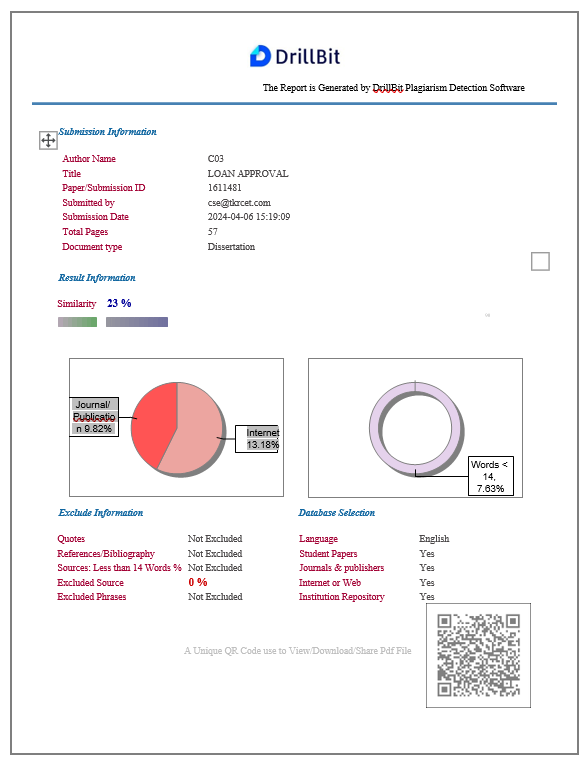
[16] Kim, A., Cho, S.B. (2019). An ensemble semi-supervised learning method for predicting defaults in social lending. \*Eng. Appl. Artif. Intell.\*, 81(May), 193–199.

[17] Li, M., Yan, C., Liu, W. (2021). The network loan risk prediction model based on Convolutional neural network and Stacking fusion model. \*Appl. Soft Comput.\*, 113(Dec.), 107961.

[18] Lim, B., Arık, S.O., Loeff, N., et al. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. \*Int. J. Forecast.\*, 37(4), 1748–1764.

[19] Liu, J., Gao, Y., Hu, F. (2021). A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. \*Comput. Secur.\*, 106(Jul.), 102289.

[20] Nguyen, V.Q., Tran, V.L., Nguyen, D.D., et al. (2022). Novel hybrid MFO-XGBoost model for predicting the racking ratio of the rectangular tunnels subjected to seismic loading. \*Transp. Geotech.\*, 37(Nov.), 100878.



**PLAGIARISM REPORT**

