"Loan Eligibility Prediction Using Machine Learning Models"

Submitted in partial fulfillment of the requirements
for the course of MA-308 (Mini Project)
in
3rd Year (6th) Semester of

Master of Science

(Five Year Integrated Program)
in Mathematics
submitted by

Sanghavi Ishika Sandeep (I20MA011)
Dhanani Jatinbhai Chimanbhai (I20MA022)
Fatema Maksood Bhatt (I20MA028)

Under the supervision of
Dr. Jayesh M. Dhodiya
Associate Professor

DEPARTMENT OF MATHEMATICS
SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY
SURAT-395007, GUJARAT, INDIA

May 2023

Department of Mathematics
Sardar Vallabhbhai National Institute of Technology
(An Institute of National Importance, NITSER Act 2007)
Surat-395007, Gujarat, India.

APPROVAL SHEET

Report entitled "**Loan Eligibility Prediction Using Machine Learning Models**" by Sanghavi Ishika Sandeep, Dhanani Jatinbhai Chimanbhai, Fatema Maksood Bhatt is approved for the completion of course MA-308 (Mini Project) for the degree of Master of Science in Mathematics.

Signature of Internal Examiner

Dr. V. H. Pradhan
(**Internal Examiner**)

Signature of Internal Examiner

Dr. Ramakanta Meher
(**Internal Examiner**)

Signature of Supervisor

Dr. Jayesh M. Dhodiya
(**Supervisor**)

Date: 09/05/2023
Place: Surat

Department of Mathematics
Sardar Vallabhbhai National Institute of Technology
(An Institute of National Importance, NITSER Act 2007)
Surat-395007, Gujarat, India.

DECLARATION

We hereby declare that the report entitled “Loan Eligibility Prediction Using Machine Learning Models" is a genuine record of work carried out by us and no part of this report has been submitted to any University or Institution for the completion of any course.

Signature:
Sanghavi Ishika Sandeep
Admission No.: I20MA011
Department of Mathematics
Sardar Vallabhbhai National Institute of Technology
Surat-395007

Signature:
Dhanani Jatinbhai Chimanbhai
Admission No.: I20MA022
Department of Mathematics
Sardar Vallabhbhai National Institute of Technology
Surat-395007

Signature:
Fatema Maksood Bhatt
Admission No.: I20MA028
Department of Mathematics
Sardar Vallabhbhai National Institute of Technology
Surat-395007

Date: 09/05/2023
Place: Surat

Department of Mathematics
Sardar Vallabhbhai National Institute of Technology
(An Institute of National Importance, NITSER Act 2007)
Surat-395007, Gujarat, India.

CERTIFICATE

This is to certify that the course's report entitled "Loan Eligibility Prediction Using Machine Learning Models" submitted by Sanghavi Ishika Sandeep, Dhanani Jatinbhai Chimanbhai and Fatema Maksood Bhatt in fulfilment for the completion of course of MA 308: Mini Project at Sardar Vallabhbhai National Institute of Technology, Surat is record of their work carried out under my supervision and guidance.

Signature:
Dr. Jayesh M. Dhodiya
Associate Professor
Department of Mathematics
Sardar Vallabhbhai National Institute of Technology
Surat-395007

Date: 09/05/2023
Place: Surat

Acknowledgment

We are very grateful as students of 5 Years Integrated M.Sc. in Mathematics at Sardar Vallabhbhai National Institute of Technology, Surat. First, we would like to express our genuine appreciation to our supervisor Dr. Jayesh M. Dhodiya for his guidance for our work. To work with him is a great opportunity and pleasure to us. We are thankful for Prof. Anupam Shukla, SVNIT, Director of SVNIT and all other Faculties, Research Scholars and Non-Teaching staff of our department for their regular inspiration and co-activity.

Last but not least we are especially grateful to our respective parents who are always there for us and support us emotionally and financially. It is all because of them that we have made it so far.

List of Figures

1.1 Flow Chart of Modeling 3
1.2 Flowchart of data mining process 4

2.1 Linear Regression 1 6
2.2 Linear Regression 2 6
2.3 Logistic Regression 7
2.4 Convex and Non-convex curves 7
2.5 Random Forest 8
2.6 Random Forest 9
2.7 Hyperplane SVM 10

3.1 Convex and Non-convex curves 14
3.2 Target Variable 14
3.3 Correlation Map 15
3.4 Income Stability Map 16

Contents

Abstract **vi**

Problem Statement **vi**

1 Introduction to Data mining **1**
- 1.1 Introduction . . . 1
- 1.2 Fundamental of Modelling . . . 2
- 1.3 Methodology . . . 3

2 Models **5**
- 2.1 Introduction . . . 5
- 2.2 Linear Regression . . . 5
- 2.3 Logistic Regression . . . 6
- 2.4 Random Forest . . . 8
- 2.5 Support Vector Machine (SVM) . . . 9
  - 2.5.1 Linear SVM . . . 9
- 2.6 Decision Tree Model . . . 10

3 Solution to the problem using Python **12**
- 3.1 Description About the Data Set . . . 12
- 3.2 Pre-processing of dataset . . . 13
  - 3.2.1 Importing libraries and loading data . . . 13
  - 3.2.2 Dealing with duplicate and missing data . . . 13
  - 3.2.3 About Target variable . . . 14
  - 3.2.4 Relational Between All Variables . . . 14
  - 3.2.5 Working on handling missing data . . . 15
  - 3.2.6 Encoding . . . 16

4 Implementation of Code in Python **17**
- 4.1 Statistical Analysis and Data Cleaning . . . 17

5 Conclusion **25**
- 5.1 Model Wise Accuracy . . . 25
- 5.2 Conclusion . . . 25
- 5.3 Reference Links . . . 27

Abstract

The Loan Eligibility Prediction project aims to develop machine learning models that can accurately predict whether a loan applicant is eligible for a loan or not based on their demographic and financial information. This project is motivated by the need to improve loan approval processes in financial institutions by making them more efficient and objective.

Here we have taken the dataset from Kaggle.com. The project involves collecting a dataset of loan applicants' information, including their age, gender, income, credit score, employment status, and other relevant factors. The dataset is preprocessed to handle missing values, outliers, and other data quality issues. The preprocessed data is then split into training and testing sets for model development and evaluation. Several machine learning models are trained on the training data, including logistic regression, decision tree, random forest, and gradient boosting models.

This project will provide step by step guide from the mathematics of all machine learning algorithm to analysis of accuracy obtain from different algorithm.

Problem Statement

Loan Eligibility Prediction Using Machine Learning Models

Chapter 1

Introduction to Data mining

1.1 Introduction

Modern science and engineering utilize first-principle models to describe physical systems, biological beings, and social systems surrounding us. These models, such as Maxwell's Equations and Newton's Laws of Motion, are useful but have their limitations. The rigid constraints of these principles make it difficult to find them in the real world or satisfy them entirely. To address these challenges, probability theory and statistics are used to understand the relationship between different variables. The mathematical principles of probability theory and statistics can be applied to tackle this problem of understanding complex systems without strict constraints. However, to use these principles, we require data.

"Data mining is the process of applying computational methods to large amounts of data in order to reveal new non-trivial and relevant information." (4)
Data mining involves the extraction of valuable information and knowledge from extensive datasets. By utilizing computational and statistical techniques, data mining analyzes data, identifies patterns, and uncovers previously unknown relationships and trends. The primary goal of data mining is to reveal insights and knowledge that can support informed decision-making and enhance business processes. This interdisciplinary field draws from various domains such as computer science, statistics, and artificial intelligence. Numerous industries, including finance, healthcare, retail, and telecommunications, rely on data mining to address diverse challenges such as fraud detection, customer segmentation, and predictive modeling.

In practice, data mining serves two primary objectives: prediction and description. The predictive aspect involves using data mining techniques to develop models based on known data that can accurately predict unknown or future data using mathematics and concepts from machine learning. On the other hand, in the description, we analyze patterns in the dataset to gain insights and understand the dataset, drawing valuable conclusions for the specific domain.

1. **Predictive Data-mining**
   Used to make a model from the given dataset.

2. **Descriptive Data-mining**
   To gauge non-trivial information from the given dataset.

Data mining can be used in a variety of fields. The primary tasks of data mining are given below.

1. **Regression**
   Statistical method used to discover a predictive learning function that maps data to a real-valued predictive variable.
2. **Classification**
   The Discovery of the predictive learning function is used to classify the data into multiple classes.
3. **Clustering**
   A standard descriptive task which can be done to identify finite clusters to describe the data.
4. **Summarization**
   Techniques that summarise the whole data into a compact form.
5. **Dependency modelling**
   A model which can find the dependency of the variables and can perform statistical analysis.
6. **Change and deviation detection**
   Finding outliers and serious changes in data.

1.2 Fundamental of Modelling

Data mining is an interdisciplinary field that combines mathematics, computer science, and control theory engineering. To fully understand data mining, one needs to have knowledge of how these fields interact. Since data mining does not deal with first-order principles, it uses dependent and independent variables to understand unknown mathematical systems. This process is called system identification and involves two steps: structure identification and parameter identification.

1. **Structure identification**
   To understand this step, we need some prior knowledge about the target systems and class of models within which we can find a suitable model.

   Structure identification in system identification entails identifying the appropriate class of models that best fits the system under study. This involves leveraging domain expertise to determine the parameterized function $y = f(u, t)$, where y represents the model's output, u denotes the input vector, and t represents the parameter vector. Determining the function f relies on mathematical boundaries, fundamental principles, and personal familiarity with the system.
2. **Parameter identification**
   After defining the structure model, we need optimization to determine parameters vector t such that $y^* = f(u, t^*)$ can describe the system properly.

The algorithm and flowchart of system identification is given below:

1. **Algorithm** (5)
   (a) Specify a class of formalized models and parameterize it, $y^* = f(u, t)$.
   (b) Identify the parameter and choose the parameters that will best fit the data set (the difference $y - y^*$ is minimal).
   (c) To see if the model identified response correctly to an unseen data set or not that check through validation tests.
   (d) If the results of the validation test are as expected, stop the process.

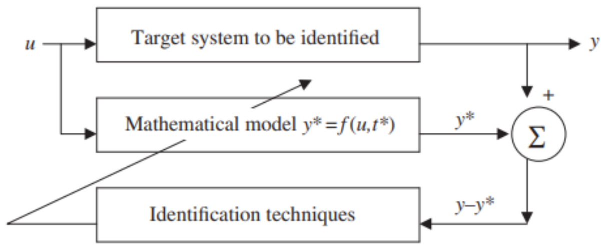

u
Target system to be identified
y
+
Mathematical model y* = f(u,t*)
y*
Σ
Identification techniques
y–y*

Figure 1.1 : Flow Chart of Modeling

1.3 Methodology

The methodology generally involves the following steps:

1. **Clear problem statement and hypothesis**
   Having a clear problem statement is essential in data mining because, without a clear problem statement, data mining will give general results which are not applied to any particular phenomena. Domain knowledge is required to extract information through a dataset.
2. **Data Collection**
   The first step in data mining is to generate or collect datasets. This can be done by conducting controlled experiments or by observation. Alternatively, free datasets are available on platforms like Kaggle that can be used to create models using data mining concepts. However, it is important to consider the quality of the data, as datasets with a high proportion of null values or that are randomly generated may not yield useful conclusions.
3. **Data pre-processing**
   When gathering datasets for data mining, it is common for the data to be incomplete and require preprocessing before mathematical calculations can be applied. This preprocessing typically involves two steps.
   (a) **Detecting outliers and removing them.**
   Outliers are the unusual data points which do not match with the overall dataset. This data point can skew the model and results. The following two strategies are applied to dealing with the outliers.
      i. First find and then remove the outliers from the dataset.
      ii. Use a model which has not very sensitive to the outliers.

(b) **Encoding, scaling and selecting features**

In scaling, if one feature has a range [0,1] and another has a range [1,200] will not have the same weight on applied techniques. Their impact is not the same on the results, so we normalize all the features to the range between [0,1].

Suppose we have variable values, such as M and F or other object classification. We cannot run the model directly on this, so we first assign each value with a different numerical, e.g. M as 0 and F as 1.

(c) **Estimate the model**

process. The choice of model(s) depends on the specific dataset and problem statement. Various machine learning models can be applied to a single problem, and their performance can be compared to select the best models.

(d) **Interpret the model and make conclusions**

Data mining is used for decision-making purposes. So after selecting the estimated model, different performance matrices are used to understand the model's logic and dependability, e.g. confusion matrix, accuracy score, and f1 score.

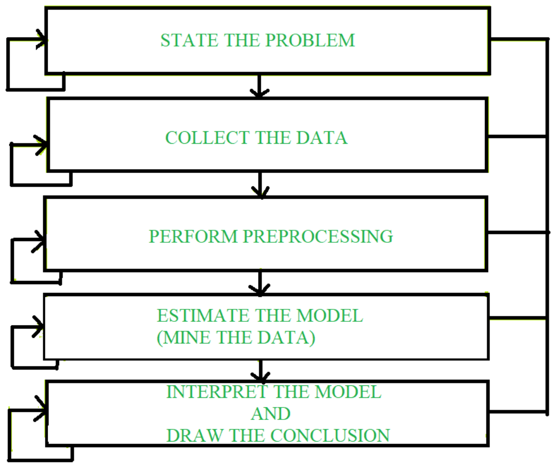

STATE THE PROBLEM
COLLECT THE DATA
PERFORM PREPROCESSING
ESTIMATE THE MODEL (MINE THE DATA)
INTERPRET THE MODEL AND DRAW THE CONCLUSION

Figure 1.2 : Data mining steps

Chapter 2

Models

2.1 Introduction

Machine learning models are computer algorithms designed to automatically learn patterns and insights from data without being explicitly programmed. These models are trained on large datasets and can make predictions or decisions based on new data. Machine learning aims to create models that can generalize well on new and unseen data. Machine learning models are widely used in various fields, including finance, healthcare, e-commerce, and many others, to make predictions, recognize patterns, and provide insights that can be used to improve business operations and decision-making. Some examples of popular machine learning models include decision trees, neural networks, support vector machines, and random forests.

2.2 Linear Regression

Linear Regression works best when the dependent and the independent variables have a linear relationship. This linear function, also called the 'hypothesis function,' is represented as $h_\theta(x) = \theta_0 + \theta_1 x = \theta^T x$, where θ_0 is the intercept along the y-axis and θ_1 is the slope. This algorithm uses the concept of drawing a 'best-fitting line' passing through all the data points such that the cost function is minimum. The cost function, also known as the mean squared error function, is represented as

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2. \tag{2.1}$$

where m is the total number of training examples, $h_\theta(xi)$ is the predicted value, and y_i is the actual value. This function is minimized using optimizing techniques such as the Gradient Descent. In the following paragraph, we will see why this algorithm might fail to predict accurate discrete values when dealing with classification problems and how Logistic Regression uses an S-shaped curve which works better than the linear function.

One of the foremost problems with using Linear Regression with classification problems is that this algorithm predicts continuous values, while we want to have discrete values as the final answer. This can still be modified to get discrete values from Linear Regression by taking a threshold value, say $y = 0.5$, to classify output. For any value of x, if $y(x)$ is greater than 0.5 we classify it as positive class (1) and below 0.5 as negative class (0) (See Fig. 2.1). At first glance, this method seems reasonable and has the potential to work, but we start witnessing problems in the presence of outliers. The best fitting line gets severely deviated, and the aforementioned strategy starts giving anomalous results (See Fig. 2.2). Another problem is that since Linear Regression gives continuous values as results, it can very possibly give answers for $h_\theta(x)$ which are less than 0 and greater than 1, the above strategy failing yet again. Hence, Linear Regression is not usually preferred.

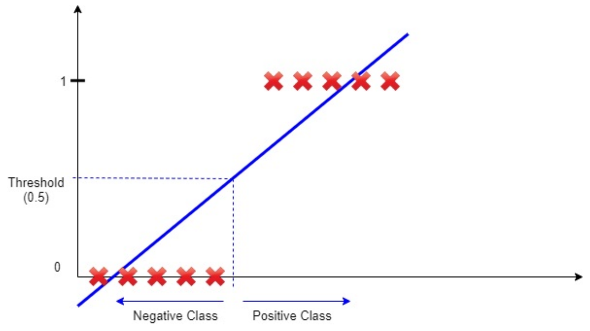


Figure 2.1 : Linear Regression 1

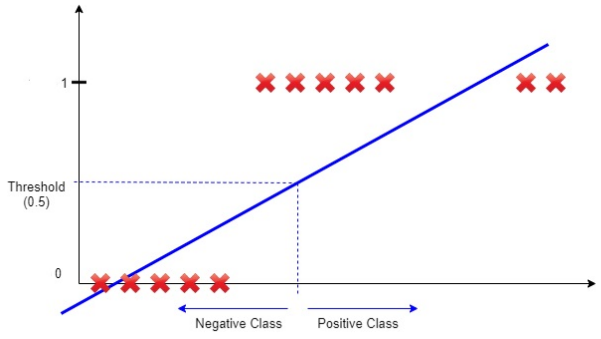


Figure 2.2 : Linear Regression 2

2.3 Logistic Regression

We aim for a function which gives $0 \leq h_\theta(x) \leq 1$ as the result. The hypothesis function $h_\theta(x) = \theta^T x$ in Linear Regression is modified by making the right side a function of g.

Hence, the hypothesis function for Logistic Regression $h_\theta(x) = g(\theta^T x)$, where $g(z) = 1/(1 + e^{-z})$ is called the sigmoid function or the logistic function. Therefore, $h_\theta(x) = 1/(1 + e^{-\theta^T x})$.

This sigmoid function is an S-shaped curve and has the range $(0, 1)$ (having asymptotes at 0 and 1), which now is convenient for our classification problem where $y = \{0, 1\}$. The value of $h_\theta(x)$ gives the estimated probability of $y = 1$ for any input x. E.g., for any particular value of x, $h_\theta(x) = 0.7$ means there is 70% chance of y being of positive class. We now say that for values of $h_\theta(x) \geq 0.5, y = 1$ is predicted, and for

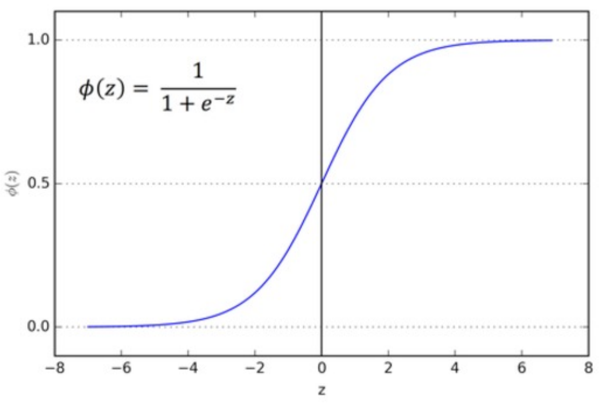


Figure 2.3 : Logistic Regression

$h_\theta(x) < 0.5, y = 0$. (See Fig. ??) Now, we rewrite the cost function for the Linear Regression as

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}[h_\theta(x^{(i)}) - y^{(i)}]^2, \tag{2.2}$$

$$Cost(h_\theta(x), y) = \frac{1}{2}[h_\theta(x^{(i)}) - y^{(i)}]^2, \tag{2.3}$$

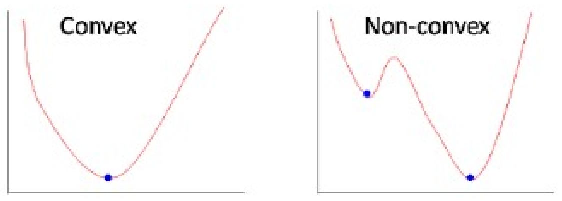


Figure 2.4 : Convex and Non-convex curves

This, however, results in a non-convex cost function (See Fig. 4). The function has various local minima, and hence applying Gradient Descend will be a huge problem as it won't be guaranteed if it will converge to the global minima. In contrast, we would want a convex cost function, a single bow shaped function, where applying Gradient Descent would guarantee a global minima. Hence we define another cost function for Logistic Regression which is convex and will be used from hereon forward.

$$Cost(h_\theta(x), y) = \left\{ \begin{array}{ll} -\log(h_\theta(x)), & \text{if } y = 1 \\ -\log(1 - h_\theta(x)), & \text{if } y = 0 \end{array} \right\}. \tag{2.4}$$

This can be written together in a compact way as

$$Cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1 - h_\theta(x)). \tag{2.5}$$

Now, putting this value of $Cost(h_\theta(x), y)$ in 2.2, we have

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)}), \\ &= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))], \\ &= -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} Cost(h_\theta(x), y). \end{aligned} \tag{2.6}$$

Now, to fit parameter theta, we apply Gradient Descent on this $J(\theta)$ to get the global minima at that particular theta and minimize the function. Henceforth, for any new value of x, the predicted value will be $h_\theta(x) = 1/(1+e^{-\theta^T x})$, where $h_\theta(x)$ is the probability of $y = 1$.

2.4 Random Forest

Random forest is a supervised learning algorithm. It is also called as random decision forests. Random Forest is a widely used machine learning model for classification and regression tasks. It is an ensemble learning method that constructs a multitude of decision trees and combines their predictions to achieve high accuracy and reduce overfitting.

The mathematics behind Random Forest involves two main concepts: decision trees and bootstrap aggregating (bagging).

A decision tree is a binary tree data structure that recursively splits the dataset into subsets based on the most significant feature at each node. This process is repeated until the final nodes, or leaves, represent the predicted class or value. A decision tree can be prone to overfitting, which means it may perform well on the training data but poorly on the unseen data.

To reduce overfitting, Random Forest employs bagging, which is a technique that involves creating multiple subsets of the original dataset by randomly sampling with replacement. Each subset is used to train a decision tree model, and the final predictions are made by aggregating the results from all trees.

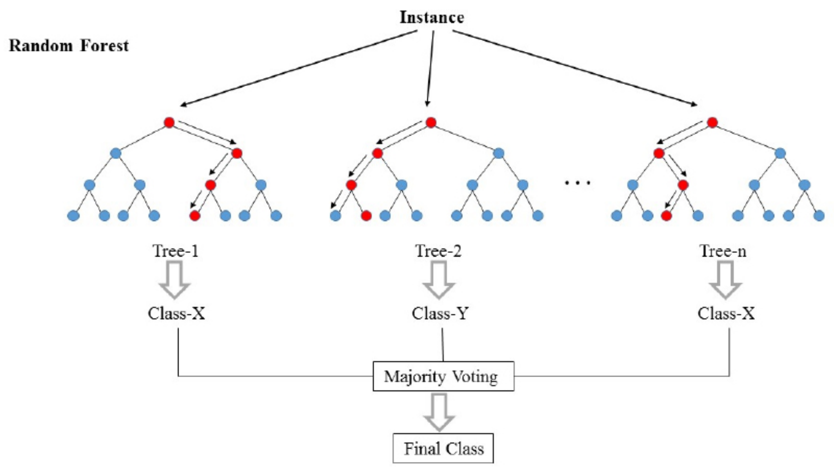


Figure 2.5 : Random Forest

In this algorithm, each decision tree is built using a random sample of the training data, with some instances being repeated (called "bootstrap sample"). One-third of the data is then set aside as test data (called "out-of-bag sample"). Another randomization step called "feature bagging" is applied to reduce the correlation between decision trees.

The way predictions are made depends on the type of problem: for regression, the individual decision trees are averaged, while for classification, the most frequent categorical variable is used. Finally, the out-of-bag sample is used for cross-validation, which helps to determine the accuracy of the prediction.

How Random Forest Work

1. Randomly sample a subset from the original dataset, allowing for replacement.
2. Randomly choose a subset of features to be used at each node of the decision tree.
3. Construct a decision tree using the selected subset of data and features.
4. Repeat steps 1-3 iteratively to create multiple decision trees.
5. Generate predictions by combining the outcomes from all decision trees in the ensemble.

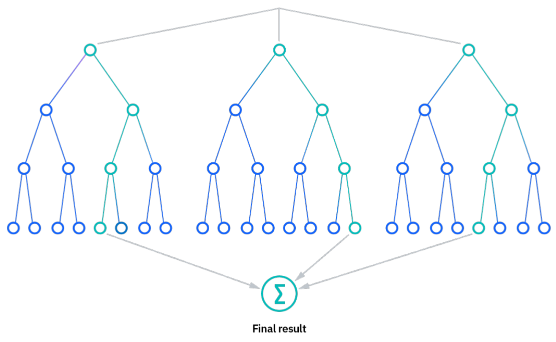


Figure 2.6 : Random Forest

2.5 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm commonly used for classification and regression analysis. SVM finds the best possible boundary or decision boundary that separates the data points into different classes, which makes it ideal for classification tasks.

The mathematical concept behind SVM involves finding the hyperplane that maximizes the margin between the two classes. In a two-dimensional space, the hyperplane is simply a line; in higher dimensions, it is a hyperplane. The margin is the distance between the hyperplane and the closest data points from each class, called support vectors.

2.5.1 Linear SVM

Let's begin by considering a training set comprising 'n' points denoted as $\{\mathbf{x}_i, y_i\}, where$ y$_i$ belongs to the class labels $\{-1, 1\}$, representing the class of the point x_i. Our goal is to discover a classifier, specifically a hyperplane, capable of mapping the points xi into a higher-dimensional space, enabling a clear separation between the two distinct classes. Additionally, we aim to maximize the margin of the hyperplane, which refers to maximizing the distance between the hyperplane and the nearest points from both groups. For linear cases, We can write hyperplane as:

$$\mathbf{x}_i \mathbf{w} + b = 0,$$

Our objective is to identify two parallel hyperplanes that effectively separate the data, and we aim to maximize the distance between these hyperplanes. Here is one way to describe them:

$$\mathbf{x}_i \mathbf{w} + b = +1,$$

and

$$\mathbf{x}_i \mathbf{w} + b = -1.$$

It can be mathematically demonstrated that the distance between the two hyperplanes is equal to $\frac{2}{||w||}$.:

$$d_+ + d_- = \frac{|1-b|}{\|w\|} + \frac{|-1-b|}{\|w\|} = \frac{2}{\|w\|}. \tag{2.7}$$

Hence, our objective is to minimize the norm of the parameter vector w since it is always positive. Additionally, we seek to ensure that for every $i \in (1, n)$, x_i and y_i satisfy the given constraints.:

$$\mathbf{x}_i \mathbf{w} + b \geq +1, \quad y_i = +1, \tag{2.8}$$
$$\mathbf{x}_i \mathbf{w} + b \leq -1, \quad y_i = -1, \tag{2.9}$$
$$\equiv \tag{2.10}$$
$$y_i(\mathbf{x}_i \mathbf{w} + b) - 1 \geq 0, \qquad \forall i. \tag{2.11}$$

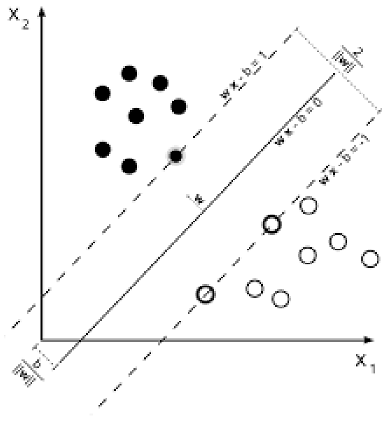


Figure 2.7 : Hyperplane SVM

2.6 Decision Tree Model

A decision tree is a supervised learning algorithm for classification and regression tasks. It builds a tree structure using the provided input data and classifies the data by traversing the tree. The tree's construction involves recursively partitioning the data into smaller subsets based on conditions, ensuring that each subset exhibits similar values for the target variable.

The mathematics behind decision trees involves finding the best split at each tree node, which is done using a measure of impurity or entropy. Entropy measures the degree of randomness or disorder in a set of data. In decision trees, entropy is used to evaluate the homogeneity of a subset of data, where a homogeneous subset has all the data points belonging to the same class.

Let X be the input feature space, Y be the output space, and D be the training dataset consisting of (x_i, y_i) pairs. The decision tree algorithm uses the following steps to construct a model:

1. Choose the best attribute a to split the data.
2. Partition the data into subsets using the chosen attribute.
3. Repeat steps 1 and 2 recursively for each subset until a stopping criterion is met.
4. Assign the most common class (for classification tasks) or the average value (for regression tasks) to each leaf node.

The splitting criterion is typically based on maximizing the information gain or minimizing the impurity of the resulting subsets. This can be computed using measures such as the Gini index, entropy, or misclassification error. For example, the Gini index is defined as:

$$Gini(D) = \sum_{k=1}^{K} p_k(1 - p_k). \tag{2.12}$$

where p_k is the proportion of samples in class k and K is the number of classes. The attribute with the highest information gain or lowest impurity is chosen for the split.

The goal is to find the split that results in the greatest reduction in entropy or the greatest information gain. Information gain is calculated by subtracting the weighted average of the entropy of the child nodes from the entropy of the parent node.

The equation for information gain is given by:

$$\text{Information Gain } = I(p) - \sum_{j=1}^{m} \frac{N_j}{N} I(p_j). \tag{2.13}$$

where A is the attribute being split on, $values(A)$ are the possible values of attribute A, and S_v is the subset of data where attribute A has value v.

The resulting decision tree can be prone to overfitting, which can be mitigated by pruning techniques such as reduced error pruning or cost complexity pruning. Additionally, ensembling methods such as random forests or gradient boosting can be used to improve the accuracy and robustness of the model.

Chapter 3

Solution to the problem using Python

The steps to solution of problem include:

1. Loading Necessary Python libraries and Data set.
2. Data Cleaning.
3. Optimization.
4. Exploratory Data Analysis.
5. Application of Statistical models.

3.1 Description About the Data Set

Size of the data set is (30000, 23)

Features of the data set are :

Customer ID	object
Name	object
Gender	object
Age	int64
Income (USD)	float64
Income Stability	object
Profession	object
Type of Employment	object
Location	object
Loan Amount Request (USD)	float64
Current Loan Expenses (USD)	float64
Expense Type 1	object
Expense Type 2	object
Dependents	float64
Credit Score	float64
No. of Defaults	int64
Has Active Credit Card	object
Property ID	int64
Property Age	float64
Property Type	int64
Property Location	object
Co-Applicant	int64
Property Price	float64
Loan Sanction Amount (USD)	float64

3.2 Pre-processing of dataset

reprocess the data by cleaning it, filling in missing values, transforming categorical variables into numerical ones, and normalizing or scaling the data to ensure consistency across all features.

3.2.1 Importing libraries and loading data

1. **Libraries**
   First import the necessary libraries like **NumPy**, **Pandas**, **Matplotlib** etc.. And Use Pandas to load the data from its location address.
   Use Pandas to load the data from its location address.

   Use necessary function to know more about the data

2. **df.shape()**
   gives the number of rows and columns in the data set which is **(30000, 24)**
3. **df.head()**
   It gives you a first 5 raws of data. To know how each features take values
4. **df.info()**
   Provides the name of each column and their respective data types
5. **df.describe()**
   Gives statistical details of each column such as total count, mean value and other statistical information.

3.2.2 Dealing with duplicate and missing data

Duplicate values

1. **df.duplicated().sum()**
   will give the total number of duplicate data in the data set. In this particular data set, there is no duplicate data hence returns 0.

Steps for working with missing data

1. Identify missing data
   (a) **df.isnull().sum()**
   Gives total null entries in each column.
   Identify the Total Missing Value Count and Percentage
2. Identify feature wise missing value Percentage
3. Correct data format of all features
4. Plot Heat Map to identify the plotting of missig values

   There are around 3.4804

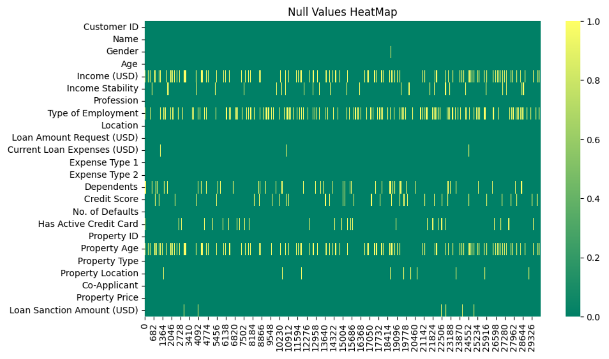
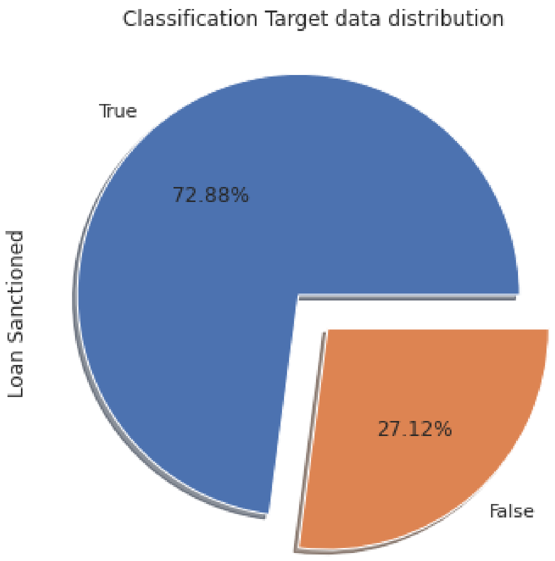


Figure 3.1 : Heat Map of Missing Values

3.2.3 About Target variable

Loan Sanction Amount

1. A lot of the target values are Zero which indicates that they were not given any loan.
2. The target show some skewness and peakedness and does not follow normal distribution
3. There are 2 unusally high values in target features



Figur 3.2 : Target Variable

3.2.4 Relational Between All Variables

1. Income is highly correlated with Property age
2. Loan Amount Request is highly correlated with Property Price
3. Loan Amount Request is highly correlated with Loan Sanction Amount
4. Current Loan Expenses is correlated with Property Price
5. Loan Sanction Amount is highly correlated with Property Price

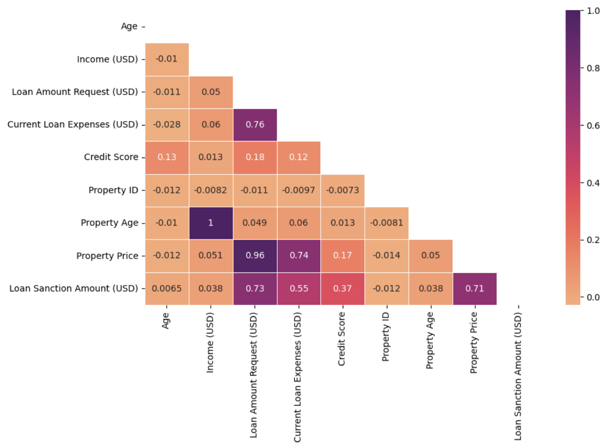


Figure 3.3 : Correlation Map of Variables

3.2.5 Working on handling missing data

Working on each data type separately

Objects

1. **df.select dtypes('object').columns**
   Separating Object data type.
2. **df[object cols].isnull().sum()**
   Searching for null values in these columns.(only 2 columns have null values in this case.)
3. **print(col, df[col].nunique())**
   To find number of unique entries in every column using for col in object cols. So later label encoding can be done (converting string entries to numbers i.e 1,2,3..etc).

   Now after this process there is no null entry left in any of the object type columns.

We will start filling in missing values using known feature which has all value and a relation with null feature.

1. Property ID the Property Location is always the same. We will fill in missing values in Property Location using Property ID
2. We can see that for each Property ID the Property Location is always the same. We will fill in missing values in Property Location using Property ID

   Realation between Income Stability and Profession
3. For every Profession Income Stability was same indicating that the data is consistent and clean
4. Credit Score Using 'Profession' and 'Income Stability' to fill in nulls
5. Income (USD) Using 'Profession', 'Type of Employment' and 'Location' to fill in nulls
6. Dependents, Has Active Credit Card, Gender, Current Loan Expenses, Property Price, Co-Applicant

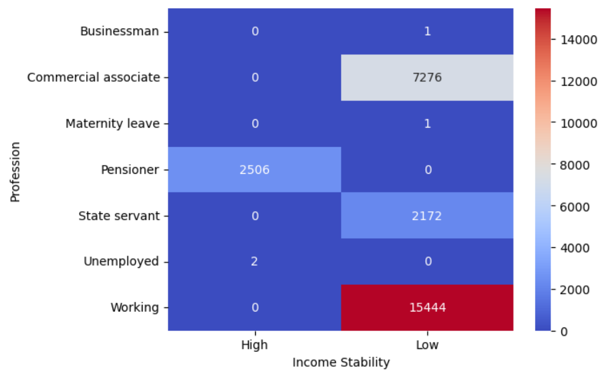


Figure 3.4 : Realation between Income Stability and Profession

7. Property Price and Co - Applicants will filled using mode

3.2.6 Encoding

For this data set we will use a OneHotEncoding technique for the purpose of increasing efficiency and accuracy oc the model. One hot encoding is a technique used to represent categorical variables as numerical variables in machine learning models. Each category is converted into a binary vector in one hot encoding where only one bit is set to 1, and all the others are set to 0. This is done to avoid the model assuming a false order or hierarchy between the categories.

One hot encoding is essential for machine learning models because most algorithms are designed to work with numerical data. By converting categorical variables into numerical variables, we can include them in the model and allow it to learn from them.

1. import pandas as pd
   cat_cols = [’Gender’, ’Income Stability’, ’Profession’, ’Location’, ’Expense Type 1’, ’Expense Type 2’, ’Has Active Credit Card’, ’Property Location’]
   encoded_data = pd.get_dummies(data, columns=cat_cols)
   merged_data = pd.concat([data, encoded_data], axis=1)
   print(merged_data.shape)

Chapter 4

Implementation of Code in Python

4.1 Statistical Analysis and Data Cleaning

Here after understanding all the data mining steps we have applied those concepts in Python, find below a important part of the code which is model building in notebook form.

```
[ ]: loan = pd.read_csv("loan.csv")
```

```
[ ]: Categorical_Cols = ['Gender', 'Income Stability', 'Profession', 'Location', 'Expense␣
      ↪Type 1', 'Expense Type 2', 'Has Active Credit Card', 'Property Location']
```

```
[ ]: cat_columns = ['Gender', 'Income Stability', 'Profession', 'Location', 'Expense Type␣
      ↪1', 'Expense Type 2', 'Has Active Credit Card', 'Property Location']
```

```
[ ]: loan = loan.drop('Customer ID', axis=1)
     loan = loan.drop('Name', axis=1)
```

```
[ ]: data = loan.copy()
```

```
[ ]: #@title
     import pandas as pd
     # Select the categorical columns to be encoded
     cat_cols = ['Gender', 'Income Stability', 'Profession', 'Location', 'Expense Type␣
      ↪1', 'Expense Type 2', 'Has Active Credit Card', 'Property Location']

     # Use one-hot encoding to encode the categorical columns
     encoded_data = pd.get_dummies(data, columns=cat_cols)

     # Merge the encoded data with the original data
     merged_data = pd.concat([data, encoded_data], axis=1)

     # Print the shape of the merged data
     print(merged_data.shape)
```

```
(28674, 62)
```

```
[ ]: import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import LabelEncoder
     from sklearn.ensemble import RandomForestRegressor
     from xgboost import XGBRegressor
     from sklearn.ensemble import GradientBoostingRegressor
     from sklearn.linear_model import LinearRegression


     # Split data into features and target
     X = encoded_data.drop('Loan Sanction Amount (USD)', axis=1)
     y = encoded_data['Loan Sanction Amount (USD)']

     # Split data into train and test sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Train random forest model
rf_model = RandomForestRegressor(n_estimators=100)
rf_model.fit(X_train, y_train)
rf_acc = rf_model.score(X_test, y_test)

# Train linear regression model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_acc = lr_model.score(X_test, y_test)

# Print model accuracies
print('Random Forest Accuracy: ', rf_acc)
print('Linear Regression Accuracy: ', lr_acc)
```

```
Random Forest Accuracy:  0.7677275511957014
XGBoost Accuracy:  0.7735067286142192
Gradient Boosting Accuracy:  0.7731421425838305
Linear Regression Accuracy:  0.6508105908487869
```

```
[ ]: data_encode = encoded_data.copy()
```

```
[ ]: col_to_log = ['Income (USD)', 'Loan Amount Request (USD)', 'Current Loan Expenses␣
      ↪(USD)', 'Property Price']
```

```
[ ]: for col in col_to_log:
         data_encode[col] = np.log(data_encode[col])
```

```
[ ]: from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LinearRegression
     from sklearn.metrics import mean_squared_error

     # Separate the features and target variable
     X = data_encode.drop(['Loan Sanction Amount (USD)'], axis=1)
     y = data_encode['Loan Sanction Amount (USD)']

     # Split data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
      ↪random_state=42)

     # Create a linear regression model object
     model = LinearRegression()
```

```
# Fit the model on the training set
model.fit(X_train, y_train)

# Predict on the testing set
y_pred = model.predict(X_test)

# Evaluate the model's performance
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

# Print the model's performance metrics
print("RMSE:", rmse)
```

```
RMSE: 32031.03149587768
```

[ ]:
```
num_encode = encoded_data.copy()
```

[ ]:
```
cols_to_drop_reg = ["Property Price", 'No. of Defaults', 'Property Type']
```

[ ]:
```
num_code = num_encode.drop(cols_to_drop_reg, axis=1)
```

[ ]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Separate the numerical and categorical features
numerical_features = ['Income (USD)', 'Loan Amount Request (USD)', 'Current Loan␣
 ↪Expenses (USD)', 'Credit Score']
categorical_features = ['Gender', 'Income Stability', 'Profession', 'Type of␣
 ↪Employment', 'Location', 'Expense Type 1', 'Expense Type 2', 'Has Active Credit␣
 ↪Card', 'Property Location']

# Apply min-max scaling to the numerical features
scaler = MinMaxScaler()
num_encode[numerical_features] = scaler.fit_transform(data[numerical_features])

# Separate the features and target variable
X = num_encode.drop(['Loan_Approved'], axis=1)
y = num_encode['Loan_Approved']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

```
model = LogisticRegression()

# Fit the model on the training set
model.fit(X_train, y_train)

# Predict on the testing set
y_pred = model.predict(X_test)

# Evaluate the model’s performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the model’s performance metrics
print("Accuracy:", accuracy)
print("Confusion matrix:\n", conf_matrix)
```

```
Accuracy: 0.7260680034873583
Confusion matrix:
 [[   0 1571]
 [   0 4164]]
```

[ ]:
```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Set the loan sanction amount threshold
threshold = 5000

X = data_encode.drop([’Loan Sanction Amount (USD)’], axis=1)
y = data_encode[’Loan Sanction Amount (USD)’]

# Convert the loan sanction amount into binary target variable
y_binary = (data_encode[’Loan Sanction Amount (USD)’] > threshold).astype(int)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.2,␣
 ↪random_state=42)

# Initialize the SVM model
svm = SVC(kernel=’linear’, C=1)

# Fit the SVM model on the training data
svm.fit(X_train, y_train)
```

```
# Predict the target variable for the test data
y_pred_svm = svm.predict(X_test)

# Calculate the accuracy score for the SVM model
accuracy_svm = accuracy_score(y_test, y_pred_svm)

print("Accuracy of SVM model: {:.2f}%".format(accuracy_svm*100))
```

```
Accuracy of SVM model: 83.23%
```

[ ]:
```
from sklearn.tree import DecisionTreeClassifier

X = data_encode.drop(['Loan Sanction Amount (USD)'], axis=1)
y = data_encode['Loan Sanction Amount (USD)']

# Set the loan sanction amount threshold
threshold = 5000

# Convert the loan sanction amount into binary target variable
y_binary = (data_encode['Loan Sanction Amount (USD)'] > threshold).astype(int)

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.2,␣
↪random_state=42)

# Initialize the Decision Tree model
dt = DecisionTreeClassifier()

# Fit the Decision Tree model on the training data
dt.fit(X_train, y_train)

# Predict the target variable for the test data
y_pred_dt = dt.predict(X_test)

# Calculate the accuracy score for the Decision Tree model
accuracy_dt = accuracy_score(y_test, y_pred_dt)

print("Accuracy of Decision Tree model: {:.2f}%".format(accuracy_dt*100))
```

```
Accuracy of Decision Tree model: 85.16%
```

[ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
```

```
X = data_encode.drop(['Loan Sanction Amount (USD)'], axis=1)
y = data_encode['Loan Sanction Amount (USD)']

# Set the loan sanction amount threshold
threshold = 5000

# Convert the loan sanction amount into binary target variable
y_binary = (data_encode['Loan Sanction Amount (USD)'] > threshold).astype(int)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.2,␣
 ↪random_state=42)

# Create a Decision Tree model object
tree_model = DecisionTreeClassifier()

# Fit the model on the training set
tree_model.fit(X_train, y_train)

# Predict on the testing set
y_pred = tree_model.predict(X_test)

# Create a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the confusion matrix
print("Confusion matrix:\n", conf_matrix)
```

```
Confusion matrix:
 [[1192  379]
 [ 486 3678]]
```

[ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix

# Set the loan sanction amount threshold
threshold = 5000

X = data_encode.drop(['Loan Sanction Amount (USD)'], axis=1)
y = data_encode['Loan Sanction Amount (USD)']
```

```
# Convert the loan sanction amount into binary target variable
y_binary = (data_encode['Loan Sanction Amount (USD)'] > threshold).astype(int)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_binary, test_size=0.2,␣
 ↪random_state=42)

# Create a SVM model object
svm_model = SVC()

# Fit the model on the training set
svm_model.fit(X_train, y_train)

# Predict on the testing set
y_pred = svm_model.predict(X_test)

# Create a confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Print the confusion matrix
print("Confusion matrix:\n", conf_matrix)
```

```
Confusion matrix:
 [[   0 1571]
 [   0 4164]]
```

```
[ ]:
```

Chapter 5

Conclusion

5.1 Model Wise Accuracy

In the Loan Eligibility Prediction process, we have trained several machine learning algorithms on the preprocessed dataset. Initially, we used the Support Vector Machine, which provided excellent accuracy about 83.23% ; apart from this, We have used the Random Forest algorithm, which provided promising results with an accuracy of 76.77% . We have also performed Decision Tree Algorithm, providing an accuracy of 85.16% respectively. Respectively we have also performed Linear Regression and Logistic Regression Accuracy to check more accuracy. We have also experimented with different hyperparameters and performed cross-validation to avoid overfitting. Finally, we have selected SVM and Decision Tree as our final model and trained it on the entire preprocessed dataset to make loan eligibility predictions.

1. **Linear Regression Accuracy: 65.08%**
2. **Logistic Regression Accuracy: 72.60%**
3. **Random Forest Accuracy: 76.77%**
4. **Accuracy of SVM Model : 83.2%**
5. **Accuracy of Decision Tree model: 85.16%**

5.2 Conclusion

It is observed from the results that the Decision Tree Model and Support Vector Machine (SVM) predicts the eligibility range highly and correctly, whereas other such as Random Forest, Logistic Regression, the eligibility in decreasing order comparatively. The dataset is categorical. Hence, using these algorithms, we will be able to categorize whether the incoming customers are eligible to apply for bank loans. These interpretations help the bank employees to easily access and understand the customer data and analyze the applicants eligibility accurately; hence improper sanctioning of loans and fraud can be highly reduced. The prediction will be helpful in further formalities in the bank as it saves time and produces accuracy in the results. 17

Bibliography

[1] K Ashwitha, Arhath Kumar, Balachandra Rao, Preethi Salian, AP Supravi, et al. An approach to predict loan eligibility using machine learning. In *2022 International Conference on Artificial Intelligence and Data Engineering (AIDE)*, pages 23–28. IEEE, 2022.

[2] Prateek Dutta. A study on machine learning algorithm for enhancement of loan prediction. *International Research Journal of Modernization in Engineering Technology and Science*, 3, 2021.

[3] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[4] Aleksi Kallio and Jarno Tuimala. *Data Mining*, pages 525–528. Springer New York, New York, NY, 2013. doi:10.1007/978-1-4419-9863-7_599.

[5] Mehmed Kantardzic. *Data mining: concepts, models, methods, and algorithms*. John Wiley & Sons, 2011.

5.3 Reference Links

[1] https://towardsdatascience.com/predict-loan-eligibility-using-machine-learning-models-7a14ef904057

[2] https://www.geeksforgeeks.org/introduction-to-support-vector-machines-svm/

[3] https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

[4] https://www.analyticsvidhya.com/blog/2020/12/beginners-take-how-logistic-regression-is-related-to-linear-regression/

[5] https://ranasinghiitkgp.medium.com/mathematics-behind-decision-tree-73ee2ef82164#:~:text=The%20decision%20tree%20builds%20classification,decision%20nodes%20and%20leaf%20nodes.

[6] https://www.geeksforgeeks.org/understanding-logistic-regression/

[7] https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

[8] https://medium.com/@KrishnaRaj_Parthasarathy/ml-classification-why-accuracy-is-not-a-best-measure-for-assessing-ceeb964ae47c#:~:text=Even%20when%20model%20fails%20to,evaluation%20for%20our%20classification%20model

[9] https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/#:~:text=A%20confusion%20matrix%20is%20a,related%20terminology%20can%20be%20confusing

MINI PROJECT

ORIGINALITY REPORT

14%	7%	8%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Sardar Vallabhbhai National Inst. of Tech.Surat Student Paper	7%
2	Samantha Di Loreto, Fabio Serpilli, Valter Lori. "Application of the SVM algorithm for the development of a model classification of the visual and sound landscape", INTER-NOISE and NOISE-CON Congress and Conference Proceedings, 2023 Publication	1%
3	Mehmed Kantardzic. "Data Mining", Wiley, 2019 Publication	1%
4	digitalcommons.du.edu Internet Source	1%
5	Submitted to The University of the West of Scotland Student Paper	1%
6	alvarestech.com Internet Source	1%

7	mafiadoc.com Internet Source	1%
8	Submitted to Middle East College of Information Technology Student Paper	1%
9	Submitted to The University of Manchester Student Paper	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		