PREDICTING LOAN REPAYMENT STATUS USING MACHINE LEARNING ALGORITHM

PROJECT REPORT

By

Nikhil U

UNDER THE GUIDANCE OF

Yashvanth M

Assistant Professor

Department of Studies in Statistics

Manasagangotri Mysuru – 06

August - 2024

Declaration

I declare that the investigations reported in this project work entitled "PREDICTING LOAN REPAYMENT STATUS USING MACHINE LEARNING ALGORITHM" was carried out by me during the final semester of M.Sc. Statistics during the year 2024 in Department of Studies in Statistics, University of Mysore, Manasagangotri.

Date: 09 August 2024

Place: Mysuru (Nikhil U)

CERTIFICATE

This is to certify that project entitled, "PREDICTING LOAN REPAYMENT STATUS USING MACHINE LEARNING ALGORITHM" has been submitted by Mr.Nikhil U, is the result of work done by her during her final semester of Master of Science in Statistics 2023-2024 under my supervision in the Department of Studies in Statistics, University of Mysore, Manasagangotri, Mysuru-570006.

Date: 09 August 2024

Place: Mysuru   Yashvanth M

ABSTRACT

Accurate prediction of loan repayment is a crucial task for financial institutions, as it directly impacts risk management and profitability. This project explores the use of the k-Nearest Neighbors (KNN) algorithm to predict whether a loan will be repaid or not, leveraging historical data. KNN, a non-parametric and intuitive classification method, determines the repayment status by evaluating the similarities between new loan applications and existing cases. The study involves several key phases: data collection and preprocessing, feature selection, model training, and performance evaluation. Initially, a comprehensive dataset containing various attributes of loan applicants, such as credit score, income, and loan amount, is curated and prepared for analysis. Essential features are then identified to enhance the predictive power of the model. The KNN algorithm is trained and validated on this processed data, with an emphasis on optimizing its parameters for maximum accuracy. The performance of the KNN model is assessed using metrics such as accuracy, precision, recall, and F1 score, ensuring a thorough evaluation of its predictive capabilities. The results demonstrate the effectiveness of KNN in distinguishing between repaid and defaulted loans, providing valuable insights for lenders. This project underscores the potential of KNN in financial risk assessment, offering a practical tool for improving loan approval processes. By implementing this model, financial institutions can enhance their decision-making processes, minimize defaults, and better manage lending risks.

1. INTRODUCTION

In the financial sector, accurately predicting whether a loan will be repaid or defaulted is crucial for managing risk and ensuring the stability of lending institutions. With the rise of machine learning, data-driven approaches have become increasingly effective in enhancing the predictive accuracy of these decisions. The project focuses on leveraging the K-Nearest Neighbors (KNN) algorithm to predict the repayment status on loans based on historical loan data. The KNN algorithm is a simple, yet powerful, non-parametric method used for classification and regression tasks. By evaluating the similarities between new loan applications and previously observed cases, KNN can provide insights into the likelihood of loan repayment. This method is particularly advantageous due to its simplicity and the intuitive nature of its approach. The primary objective of this project is to develop a predictive model using KNN that can effectively classify loan applications into two categories: repaid or not repaid. By analyzing various features of loan applicants, such as credit score, income, loan amount, and other relevant factors, the model aims to assist financial institutions in making more informed lending decisions.

This project will cover the following key aspects:

- Data Collection and Preprocessing.
- Exploratory data analysis.
- Model Building.
- Model Evaluation.
- Insights and interpretation.

1.1. MACHINE LEARNING

1.1.1. What is Machine Learning?

Machine Learning is a subfield of artificial intelligence (AI) that focuses on developing algorithms and statistical models that enable computers to perform specific tasks without being explicitly programmed to do so. Instead of following pre-defined instructions, machine learning systems learn patterns and make decisions based on data.

Machine learning algorithms are the programs that can learn the hidden patterns from the data, predict the output, and improve the performances from experiences on their own.

1.1.2. **Classification of Machine Learning Algorithms:**

Machine Learning Algorithms are broadly classified into three types:

I. Supervised Learning Algorithms
II. Unsupervised Learning Algorithms
III. Reinforcement Learning Algorithms

I. Supervised Learning Algorithms:

Supervised learning is a type of machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labelled dataset. Once the training and processing are done, the model is tested by providing a sample test data to check whether it predicts the correct output. The goal of supervised learning is to map input data with the output data. Supervised learning is based on supervision, and it same as when a student learns things in the teacher’s supervision.

Supervised learning can be divided further into two categories of problem:

- Classification
- Regression

Examples of some popular supervised learning algorithms are Simple Linear Regression, Decision tree, Logistic Regression, KNN algorithm, etc.

II. Unsupervised Learning Algorithms:

It is a type of machine learning algorithm in which machine does not need any external supervision to learn from the data, hence called unsupervised learning. The unsupervised models can be trained using unlabeled dataset that is not classified, nor categorized, and the algorithm needs to act on that data without any supervision. In unsupervised learning, the model doesn’t have a predefined output, and it tries to find useful insights from the huge amount of data. These are further classified into two categories of problems:

- Clustering
- Association

Examples of some unsupervised learning algorithm are K-means Clustering, Apriori Algorithm, Eclat, etc.

III. Reinforcement Learning Algorithms:

In Reinforcement learning, an agent interacts with its environment by producing actions, and learn with the help of feedback. The feedback is given to the agent in the form of rewards, such as for each good action, he gets a positive reward, and for each bad action, he gets a negative reward. There is no supervision provided to the agent. ***Q-Learning*** is used in reinforcement learning.

1.2. Types of Machine Learning Algorithms:

1.2.1. Linear Regression:

Linear regression is one of the most fundamental and widely used algorithms in machine learning and statistics. It is supervised learning technique used for modelling the relationship between a dependent variable and one or more independent variables. The primary goal of linear regression is to predict the value of the dependent variable based on the values of the independent variables. Some common types of linear regression are simple linear regression, multiple linear regression, ridge regression, polynomial regression, etc. The simple linear regression model is given by,

$$Y = \beta_0 + \beta_1 * X$$

Y	–	Response Variable
β_0	–	Intercept
β_1	–	Slope
X	–	Regressor Variable

1.2.2. Decision Tree:

A decision tree is a versatile and powerful machine learning algorithm used for both classification and regression tasks. A decision tree models the decision logics i.e., tests and corresponds outcomes for classifying data items into tree-like structure. The nodes of a decision tree normally have multiple levels where the first and top-most node is called the root node. All internal nodes represent tests on input variables or attributes. Depending on the test outcome, the classification algorithm branches towards the appropriate child node where the process of test and branching repeats until it reaches the leaf node. The leaf or terminal nodes corresponds to the decision. It modes decisions and their possible consequences as a tree-like structure of nodes and branches, making it easy to interpret and understand. Decision trees are easy to visualize and interpret. The decision-making process is transparent.

1.2.3. Logistic Regression:

Logistic regression is a widely used statistical method for binary classification problems, where the goal is to predict the probability that a given input belongs to one of two possible classes. Despite its name, logistic regression is a classification algorithm rather than a regression algorithm. Logistic regression finds the best-fitting linear decision boundary that separates the two classes. Logistic regression is a fundamental algorithm for binary classification tasks, valued for its simplicity, interpretability, and efficiency. It can be considered as an extension of ordinary regression and can model only a dichotomous variable which usually represent the occurrence or non-occurrence of an event. Logistic regression helps in finding the probability that a new instance belongs to a certain class.

1.2.4. Random Forest:

A random forest is an ensemble classifier and consisting of many decision trees similar to the way a forest is a collection of many trees. Decision trees that are grown very deep often cause over-fitting of the training data, resulting a high variation in classification outcome for a small change in the input data. They are very sensitive to their training data, which makes the errorprone to the test dataset. The different decision trees of a random forest are trained using the different parts of the training dataset. To classify a new sample, the input vector of that sample are required to pass down with each decision tree of the forest. Each decision tree then considers a different part of the input vector and gives a classification outcome.

1.2.5. Naïve Bayes:

Naïve Bayes (NB) is a classification technique based on the Bayes' theorem. These algorithms assumes the presence of a particular feature in a class is independent of the presence of any other features, hence the term 'naïve'. This theorem can describe the probability of an event based on the prior knowledge of conditions related to that event. This classifier assumes that a particular feature in a class is not directly related to any other feature although features for that class could have interdependence among themselves.

1.2.6. K-Nearest Neighbor:

The K-nearest neighbor (KNN) algorithm is one of the simplest and earliest classification algorithms. KNN is a simple, non-parametric, and lazy learning algorithm used for both classification and regression tasks. The algorithm is based on the principle that similar data points are likely to have similar outcomes. It classifies a data point based on how its neighbors are classified. It can be thought a simpler version of an NB classifier. Unlike the NB technique, the KNN algorithm does not require to consider probability values. The 'K' in the KNN algorithm is the number of nearest neighbors considered to take 'vote' from. The selection of different values for 'K' can generate different classification results for the same sample object.

2. LITERATURE SURVEY

[1] "**Loan Approval Prediction based on Machine Learning Approach**" Author- Kumar Arun, Garg Ishan, Kaur Sanmeet, Year-2018. The main Objective of this paper is to predict whether assigning the loan to particular person will be safe or not. This paper is divided into four sections (i) Data Collection (ii) Comparison of machine learning models on collected data (iii) Training of systems on most promising model (iv) Testing.

[2] "**Exploring the Machine Learning Algorithm for Prediction the Loan Sanctioning Process**" Author- E. Chandra Blessie, R. Rekha - Year- 2019. Extending credits to corporates and individuals for the smooth functioning of growing economies like India is inevitable. As increasing number of customers apply for loans in the banks and non-banking financial companies (NBFC), it is really challenging for banks and NBFCs with limited capital to device a standard resolution and safe procedure to lend money to its borrowers for their financial needs. In addition, in recent times NBFC inventories have suffered a significant downfall in terms of the stock price. It has contributed to a contagion that has also spread to other financial stocks, adversely affecting the benchmark in recent times. In this paper, an attempt is made to condense the risk involved in selecting the suitable person who could repay the loan on time thereby keeping the bank's nonperforming assets (NPA) on the hold. This is achieved by feeding the past records of the customer who acquired loans from the bank into a trained machine learning model which could yield an accurate result. The prime focus of the paper is to determine whether or not it will be safe to allocate the loan to a particular person. This paper has the following sections (i) Collection of Data, (ii) Data Cleaning and (iii) Performance Evaluation. Experimental tests found that the Naïve Bayes model has better performance Evaluation. Experimental tests found that the Naïve Bayes model has better performance than other models in terms of loan forecasting.

[3] "**Loan Prediction using machine learning model**" Year 2019 whether or not it will be safe to allocate the loan to a particular person. This paper has the following sections (i) Collection of Data, (ii) Data Cleaning and (iii) Performance Evaluation. Experimental tests found that the Naïve Bayes model has better performance than other models in terms of loan forecasting. With the enhancement in the banking sector lots of people are applying for bank loans but the bank has its limited assets which it has to grant to limited people only, so finding out to whom the loan can be granted which will be a safer option for the bank is a typical process. So in this project we try to reduce this risk factor behind selecting the safe person so as to save lots of bank efforts and assets. This is done by mining the Big Data of the previous records of the people to whom the loan was granted before and on the basis of these records/experiences the machine was trained using the machine learning model which give the most accurate result The main objective of this project is to predict whether assigning the loan to particular person will be safe or not. This paper is divided into four sections (i)Data Collection (ii) Comparison of machine learning models on collected data (iii) Training of

system on most promising model (iv) Testing. In this paper we are predict the loan data by using some machine learning algorithms they are classification, logic regression, Decision Tree and gradient boosting.

[4] “**Loan Prediction using Decision Tree and Random Forest**” Author- Kshitiz Gautam, Arun Pratap Singh, Keshav Tyagi, Mr. Suresh Kumar Year-2020. In India the number of people or organization applying for loan gets increased every year. The bank have to put in a lot of work to analyse or predict whether the customer can pay back the loan amount or not (defaulter or non-defaulter) in the given time. The aim of this paper is to find the nature or background or credibility of client that is applying for the loan. We use exploratory data analysis technique to deal with problem of approving or rejecting the loan request or in short loan prediction. The main focus of this paper is to determine whether the loan given to a particular person or an organization shall be approved or not.

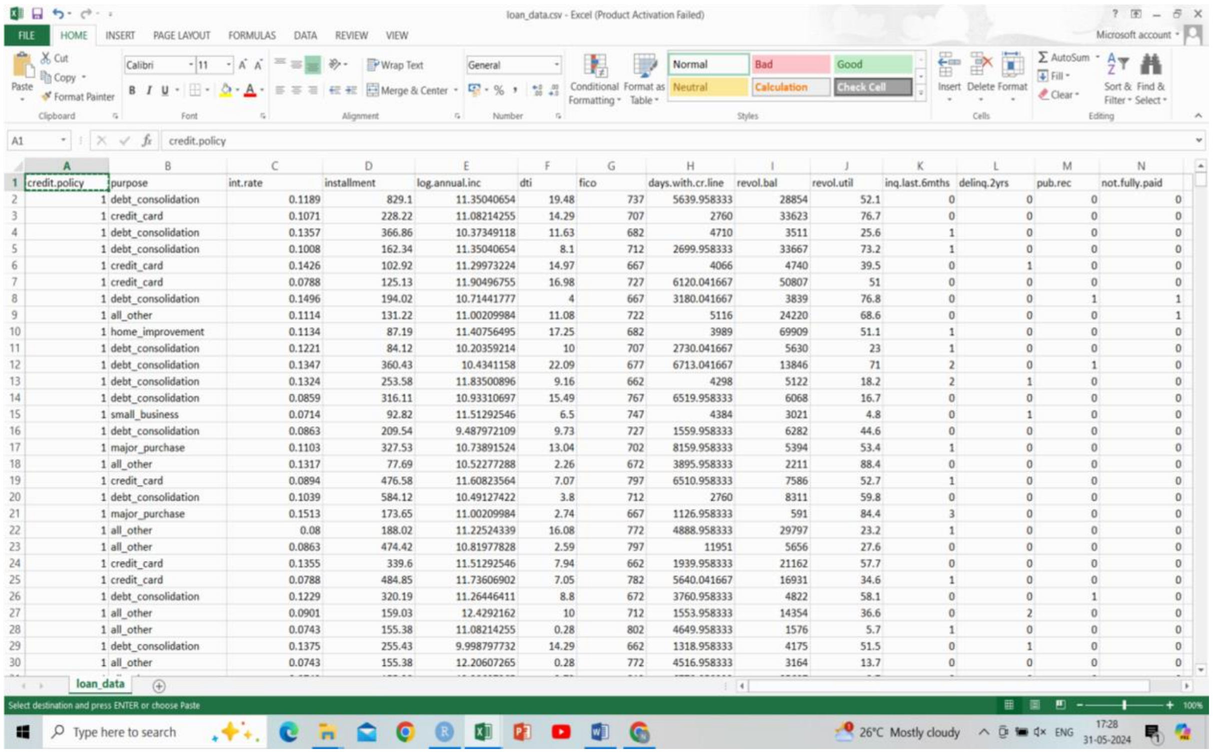
3. EXPLORATORY DATA ANALYSIS ON THE DATASET

3.1. Description for the Loan Data:

The dataset consists of 9578 entries with 14 features related to loan applications. Here is a detailed description of each feature:

Variable	Description
credit.policy	Indicates whether the customer meets the credit underwriting criteria(1:meets criteria, 0:does not meet criteria)
purpose	The purpose of the loan (categorical variable with values such as “debt_consolidation”, “credit_card”, etc.).
int.rate	The interest rate on the loan (a continuous variable).
installment	The monthly installment to be paid by the borrower (a continuous variable).
log.annual.inc	The natural logarithm of the annual income of the borrower (a continuous variable).
dti	Debt-to-income ratio of the borrower (a continuous variable).
fico	The FICO credit score of the borrower (a continuous variable).
days.with.cr.line	The number of days the borrower has had a credit line (a continuous variable).
revol.bal	The revolving balance on the borrower’s credit line (a continuous variable).
revol.util	The borrower’s revolving line utilization rate (the amount of the credit line used relative to the total credit available, a continuous variable).
inq.last.6mths	The number of inquiries by creditors in the last 6 months (a continuous variable).
delinq.2yrs	The number of times the borrower has been 30+ days past due on a payment in the past 2 years (a continuous variable).

pub.rec	The number of derogatory public records (a continuous variable).
not.fully.paid	Indicates whether the loan was not fully paid back (1: not fully paid, 0:fully paid).



loan_data.csv - Excel (Product Activation Failed)

A1 credit.policy

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	credit.policy	purpose	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs	pub.rec	not.fully.paid
2	1	debt_consolidation	0.1189	829.1	11.35040654	19.48	737	5639.958333	28854	52.1	0	0	0	0
3	1	credit_card	0.1071	228.22	11.08214255	14.29	707	2760	33623	76.7	0	0	0	0
4	1	debt_consolidation	0.1357	366.86	10.37349118	11.63	682	4710	3511	25.6	1	0	0	0
5	1	debt_consolidation	0.1008	162.34	11.35040654	8.1	712	2699.958333	33667	73.2	1	0	0	0
6	1	credit_card	0.1426	102.92	11.29973224	14.97	667	4066	4740	39.5	0	1	0	0
7	1	credit_card	0.0788	125.13	11.90496755	16.98	727	6120.041667	50807	51	0	0	0	0
8	1	debt_consolidation	0.1496	194.02	10.71441777	4	667	3180.041667	3839	76.8	0	0	1	1
9	1	all_other	0.1114	131.22	11.00209984	11.08	722	5116	24220	68.6	0	0	0	1
10	1	home_improvement	0.1134	87.19	11.40756495	17.25	682	3989	69909	51.1	1	0	0	0
11	1	debt_consolidation	0.1221	84.12	10.20359214	10	707	2730.041667	5630	23	1	0	0	0
12	1	debt_consolidation	0.1347	360.43	10.4341158	22.09	677	6713.041667	13846	71	2	0	1	0
13	1	debt_consolidation	0.1324	253.58	11.83500896	9.16	662	4298	5122	18.2	2	1	0	0
14	1	debt_consolidation	0.0859	316.11	10.93310697	15.49	767	6519.958333	6068	16.7	0	0	0	0
15	1	small_business	0.0714	92.82	11.51292546	6.5	747	4384	3021	4.8	0	1	0	0
16	1	debt_consolidation	0.0863	209.54	9.487972109	9.73	727	1559.958333	6282	44.6	0	0	0	0
17	1	major_purchase	0.1103	327.53	10.73891524	13.04	702	8159.958333	5394	53.4	1	0	0	0
18	1	all_other	0.1317	77.69	10.52277288	2.26	672	3895.958333	2211	88.4	0	0	0	0
19	1	credit_card	0.0894	476.58	11.60823564	7.07	797	6510.958333	7586	52.7	1	0	0	0
20	1	debt_consolidation	0.1039	584.12	10.49127422	3.8	712	2760	8311	59.8	0	0	0	0
21	1	major_purchase	0.1513	173.65	11.00209984	2.74	667	1126.958333	591	84.4	3	0	0	0
22	1	all_other	0.08	188.02	11.22524339	16.08	772	4888.958333	29797	23.2	1	0	0	0
23	1	all_other	0.0863	474.42	10.81977828	2.59	797	11951	5656	27.6	0	0	0	0
24	1	credit_card	0.1355	339.6	11.51292546	7.94	662	1939.958333	21162	57.7	0	0	0	0
25	1	credit_card	0.0788	484.85	11.73606902	7.05	782	5640.041667	16931	34.6	1	0	0	0
26	1	debt_consolidation	0.1229	320.19	11.26446411	8.8	672	3760.958333	4822	58.1	0	0	1	0
27	1	all_other	0.0901	159.03	12.4292162	10	712	1553.958333	14354	36.6	0	2	0	0
28	1	all_other	0.0743	155.38	11.08214255	0.28	802	4649.958333	1576	5.7	1	0	0	0
29	1	debt_consolidation	0.1375	255.43	9.998797732	14.29	662	1318.958333	4175	51.5	0	1	0	0
30	1	all_other	0.0743	155.38	12.20607265	0.28	772	4516.958333	3164	13.7	0	0	0	0

loan_data

Select destination and press ENTER or choose Paste

3.2. Summary Statistics:

Variable	**Description**
Credit Policy	Most customers (80.5%) meet the credit underwriting criteria.
Interest Rate	Varies from 6% to 21.64%, with a mean of 12.26%.
Installment	Monthly payments ranges from \$15.67 to \$940.14, with an average of \$319.09
Log Annual Income	Varies from 7.55 to 14.53, with a mean of 10.93.
Debt-to-Income Ratio	Ranges from 0 to 29.96, with a mean of 12.61.
FICO Score	Ranges from 612 to 827, with a mean of 710.85
Days with Credit Line	Varies significantly, from 179 to 17,640 days.

Revolving Balance	Ranges from $0 to $1,207,359, with an average of $16,913.96.
Revolving Utilization Rate	Varies from 0 to 119%, with a mean of 46.80%.
Inquiries in Last 6 Months	Varies from 0 to 33, with a mean of 1.58.
Delinquencies in Last 2 Years	Most borrowers have no delinquencies (mean of 0.16).
Public Records	Mostly zero, with a few expectations (mean of 0.06).
Not Fully Paid	16% of loans were not fully paid back.

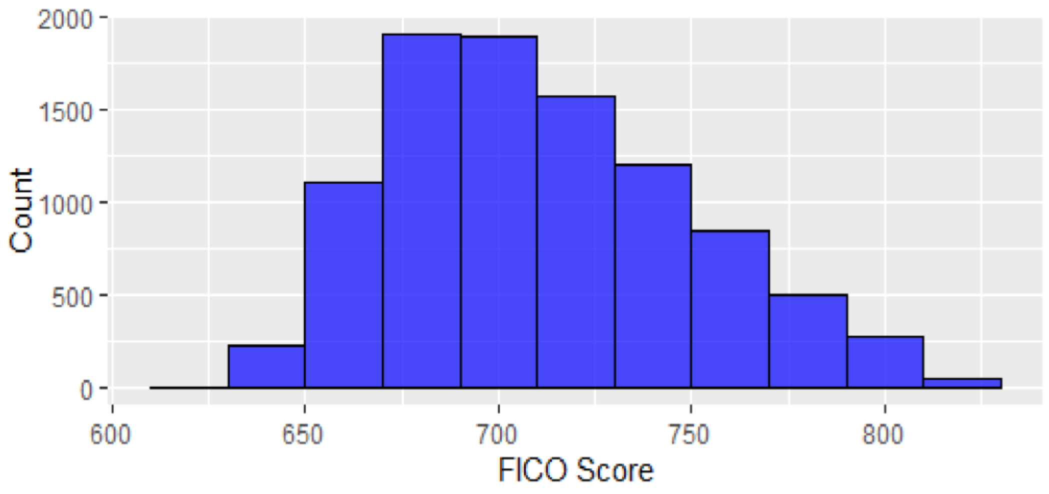
3.3. Data Insights:

- The dataset is fairly balanced in terms of meeting the credit policy.
- The financial characteristics (interest rates, income, credit scores) exhibit significant variation.
- A small percentage of loans (16%) were not fully repaid, indicating a potential risk factor for lenders.

This dataset is suitable for a machine learning project aimed at predicting loan repayment status using K-Nearest Neighbors (KNN) algorithm. The features provide a comprehensive set of predictors related to the borrower's financial health and loan characteristics.

3.4. VISUALIZATION OF THE DATASET

3.4.1. Histogram of FICO scores:

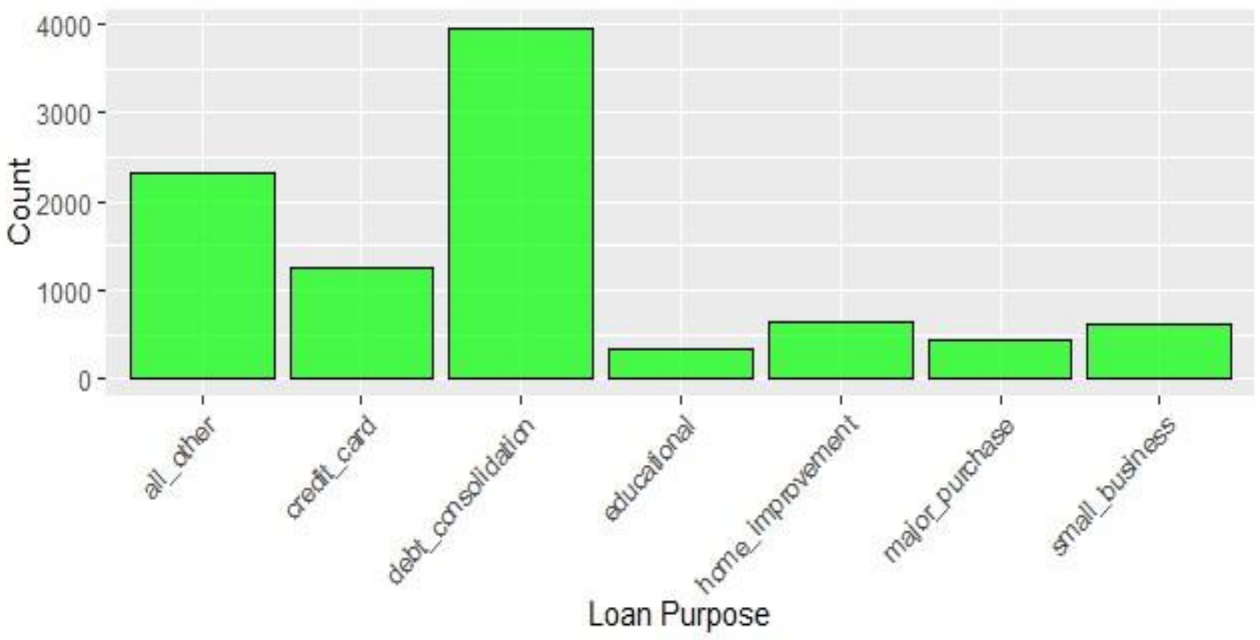


This histogram shows the distribution of FICO scores among the loan applicant.

- If the histogram is bell-shaped or has a peak, it indicates that most applicants have FICO scores around that peak value.
- A right-skewed distribution would indicate more applicants with lower FICO scores.

- A left-skewed distribution would indicate more applicants with higher FICO scores. The shape and spread of the histogram provide insights into the creditworthiness of the loan applicants, as FICO scores are a key indicator of credit risk.

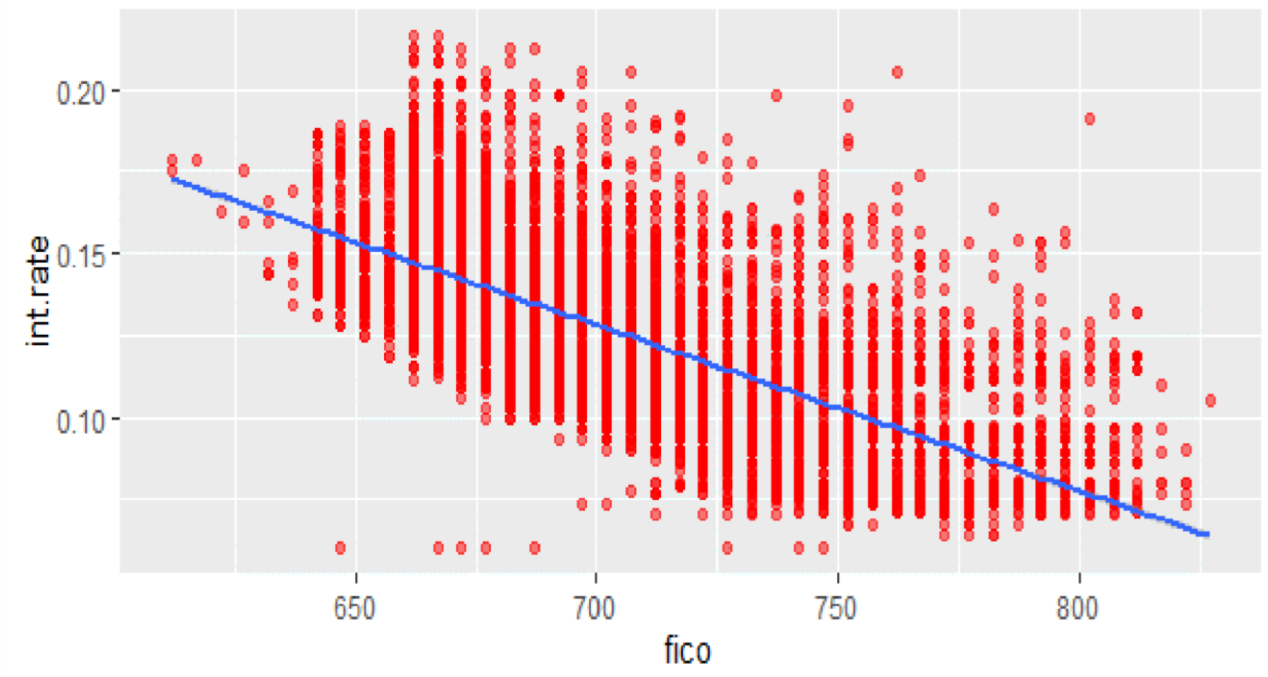
3.4.2. Bar Plot of Loan Purposes:



This bar plot illustrates the frequency of different loan purposes among the applicants.

- Taller bars indicate more common loan purposes.
   Shorter bars indicate less common loan purposes.
- The most frequent loan purposes can indicate trends in why individuals are seeking loans. For example, if "debt_consolidation" is the tallest bar, it suggests that many individuals are seeking loans to consolidate their debts.

3.4.3. Scatter Plot of Interest Rate vs. FICO Score:

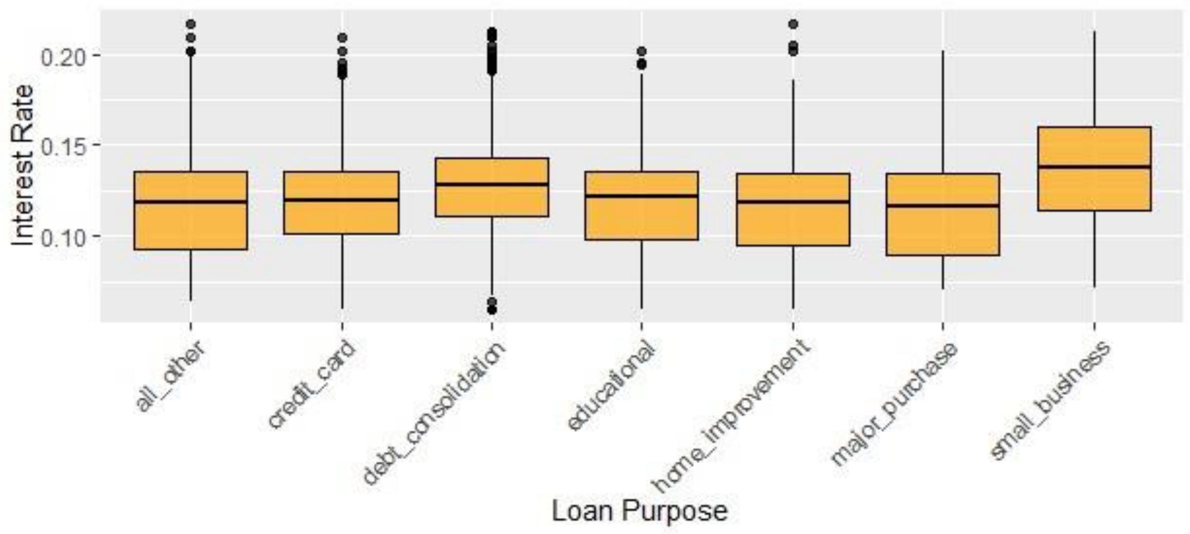


This scatter plot shows the relationship between interest rates and FICO scores.

- A downward trend would indicate that higher FICO scores are associated with lower interest rates, reflecting better creditworthiness and lower risk.
- A lack of clear trend might suggest that factors other than FICO score significantly influence interest rates.

This plot helps in understanding how credit scores impact the cost of borrowing. Typically, higher FICO scores should correspond to lower interest rates, indicating a lower risk for lenders.

3.4.4. Boxplot of Interest Rate by Loan Purpose:



This boxplot displays the distribution of interest rates for different loan purposes.

- Each box represents the interquartile range (IQR) of interest rates for a specific loan purpose, with the median as a line inside the box.

- The whiskers extend to the smallest and largest values within 1.5 * IQR from the lower and upper quartiles.
- Outliers are shown as individual points beyond the whiskers.

This plot allows comparison of interest rate distributions across different loan purposes. For instance, if the median interest rate for "credit_card" loans is lower than that for "small_business" loans, it suggests that credit card loans generally have lower interest rates. The spread of the boxes also indicates the variability in interest rates for each purpose.

4. METHODOLOGY

4.1. MISSING VALUES IN THE DATASET

Missing values in data can pose significant challenges in data analysis and modeling. In R, handling missing values efficiently is crucial to ensure the quality and reliability of the results. Below is a note on missing values, including some common techniques to handle them along with corresponding R code examples.

Missing values, often represented as NA in R, can occur for various reasons such as data entry errors, measurement issues, or unrecorded observations. Handling these missing values appropriately is essential to maintain the integrity of the analysis. Below are some common methods to deal with missing values in R. To identify missing values in a dataset, we can use functions like is.na() and sum() to get a count of missing values.

Handling missing values is a critical step in data preprocessing. Depending on the nature and extent of missing data, one can choose to either remove or impute the missing values. Simple imputation methods like mean or median substitution are easy to implement but may not be suitable for all datasets. Advanced techniques such as those provided by the mice and missForest packages offer robust alternative for handling missing data, especially in complex datasets.

Upon examining the dataset for missing values using the provided R code, it is observed that there are no missing values in any of the columns. Every column has complete data without any null entries. This indicates that the dataset is fully populated and ready for further analysis without the need for handling missing values.

4.2. SAMPLING TECHNIQUES USED IN THIS ANALYSIS

Simple random sampling is a fundamental statistical method used to select a subset of individual from a large population. The basic idea is to ensure that every individual in the population has an equal chance of being selected. This method is crucial for reducing bias and ensuring that the sample accurately represents the population, allowing for valid and reliable statistical inferences. In our data, we divided the total population into 75% as the training dataset and 25% as the test dataset by using simple random sampling technique.

4.2.1. Steps in Simple Random Sampling:

1. **Define the Population:** The first step is to clearly define the population from which you want to draw a sample. This could be dataset or a list of elements.
2. **Determine the Sample Size:** Decide the number of samples (n) you want to draw from the population. This can be based on the study's requirements, resources, or statistical guidelines.
3. **Select the Sample:** Use R's built-in functions to randomly select samples from the population.

The ‘sample ()’ function is used to randomly select rows from the dataset. The ‘set.seed(123)’ function ensures that the results are reproducible by setting a seed for the random number generator.

4.3. VALIDATION METHODS

To discuss validation methods for your dataset, we'll first need to load the data and understand its structure. Once we have a grasp on the data, we can explore various validation techniques to ensure the robustness of our models.

Validation methods are essential to evaluate the performance of a model and ensure that it generalizes well to unseen data. Here, we discuss a few common validation techniques:

1. Train -Test Split
2. K -Fold Cross -Validation
3. Leave -One -Out Cross -Validation (LOOCV)
4. Stratified K -Fold Cross -Validation

In our case, we used most commonly used validation technique Test -Train Split. The TrainTest Split validation technique is a method used in machine learning to evaluate the performance of a model. The dataset is divided into two parts: a training set and a testing set. The training set is used to train the model, while the testing set is used to evaluate its performance. This approach helps in understanding how well the model generalizes to new, unseen data.

Steps in Train-Test Split Validation:

1. **Split the Data:** Divide the dataset into a training set and a testing set. A common split ratio is 80/20 or 70/30, where 80% (70%) of the data is used for training, and 20% (or 30%) is used for testing. In our analysis, we split the dataset into 75/25 ratio, 75% for training and 25% for test set.
2. **Train the model:** Use the training set to train the machine learning model.
3. **Test the model:** Evaluate the model’s performance on the testing set.
4. **Assess Performance:** Calculate performance metrics such as accuracy, precision, recall, F1-score, etc., to understand the model's effectiveness. The confusion-Matrix function is used to evaluate the performance of the model by comparing the predicted labels with the true labels in the testing set.

The **caret** package is used for data partitioning and model evaluation, while the **e1071** package provides the implementation of the **SVM** (Support Vector Machine) model. This approach is essential for building robust and generalizable machine learning models, as it helps in understanding the model's performance on unseen data.

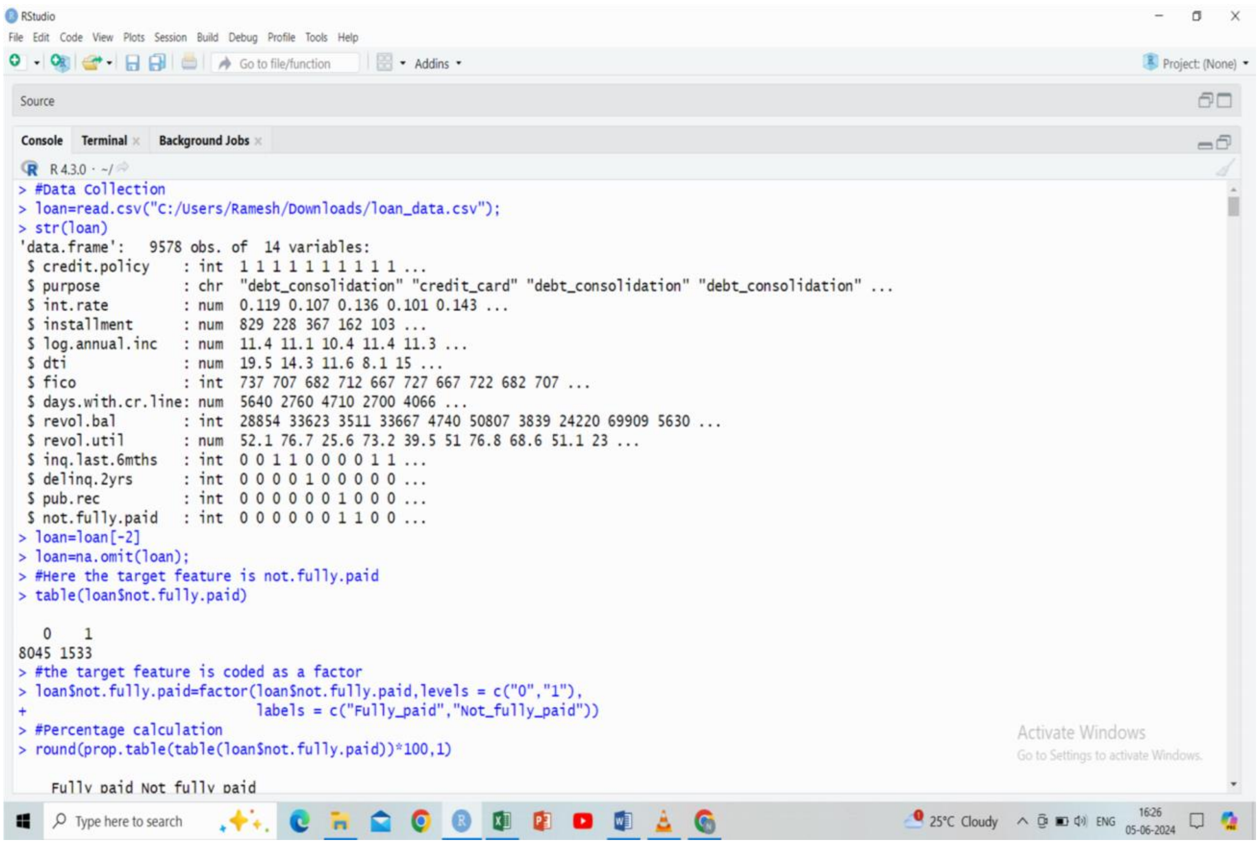
5. CODE

#Data Collection

- loan=read.csv("C:/Users/Ramesh/Downloads/loan_data.csv");
- str(loan)
- loan=loan[-2]
- loan=na.omit(loan);

#Here the Target Feature is not.fully.paid

- table(loan$not.fully.paid)



```
> #Data Collection
> loan=read.csv("C:/Users/Ramesh/Downloads/loan_data.csv");
> str(loan)
'data.frame':	9578 obs. of  14 variables:
 $ credit.policy    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ purpose          : chr  "debt_consolidation" "credit_card" "debt_consolidation" "debt_consolidation" ...
 $ int.rate         : num  0.119 0.107 0.136 0.101 0.143 ...
 $ installment      : num  829 228 367 162 103 ...
 $ log.annual.inc   : num  11.4 11.1 10.4 11.4 11.3 ...
 $ dti              : num  19.5 14.3 11.6 8.1 15 ...
 $ fico             : int  737 707 682 712 667 727 667 722 682 707 ...
 $ days.with.cr.line: num  5640 2760 4710 2700 4066 ...
 $ revol.bal        : int  28854 33623 3511 33667 4740 50807 3839 24220 69909 5630 ...
 $ revol.util       : num  52.1 76.7 25.6 73.2 39.5 51 76.8 68.6 51.1 23 ...
 $ inq.last.6mths   : int  0 0 1 1 0 0 0 0 1 1 ...
 $ delinq.2yrs      : int  0 0 0 0 1 0 0 0 0 0 ...
 $ pub.rec          : int  0 0 0 0 0 0 1 0 0 0 ...
 $ not.fully.paid   : int  0 0 0 0 0 0 1 1 0 0 ...
> loan=loan[-2]
> loan=na.omit(loan);
> #Here the target feature is not.fully.paid
> table(loan$not.fully.paid)

   0    1 
8045 1533 
> #the target feature is coded as a factor
> loan$not.fully.paid=factor(loan$not.fully.paid,levels = c("0","1"),
+                             labels = c("Fully_paid","Not_fully_paid"))
> #Percentage calculation
> round(prop.table(table(loan$not.fully.paid))*100,1)

    Fully paid Not fully paid
```

#The Target Feature is Coded as a Factor

➢ loan$not.fully.paid=factor(loan$not.fully.paid,levels = c("0","1"),

labels = c("Fully_paid","Not_fully_paid"))

#Percentage Calculation

➢ round(prop.table(table(loan$not.fully.paid))*100,1)

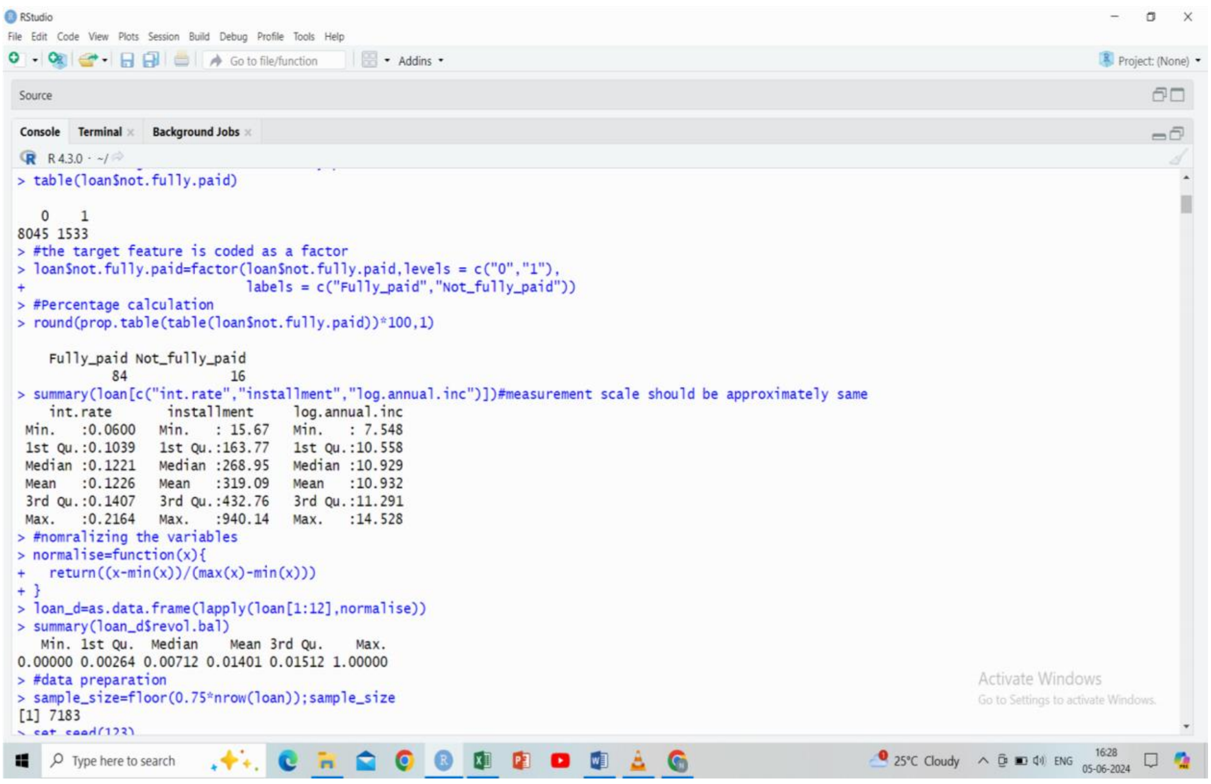
➢ summary(loan[c("int.rate","installment","log.annual.inc")])#measurement scale should be approximately same

#Normalizing the Variables

```
➢ normalise=function(x){
        return((x-min(x))/(max(x)-min(x)))
  }
```

➢ loan_d=as.data.frame(lapply(loan[1:12],normalise))

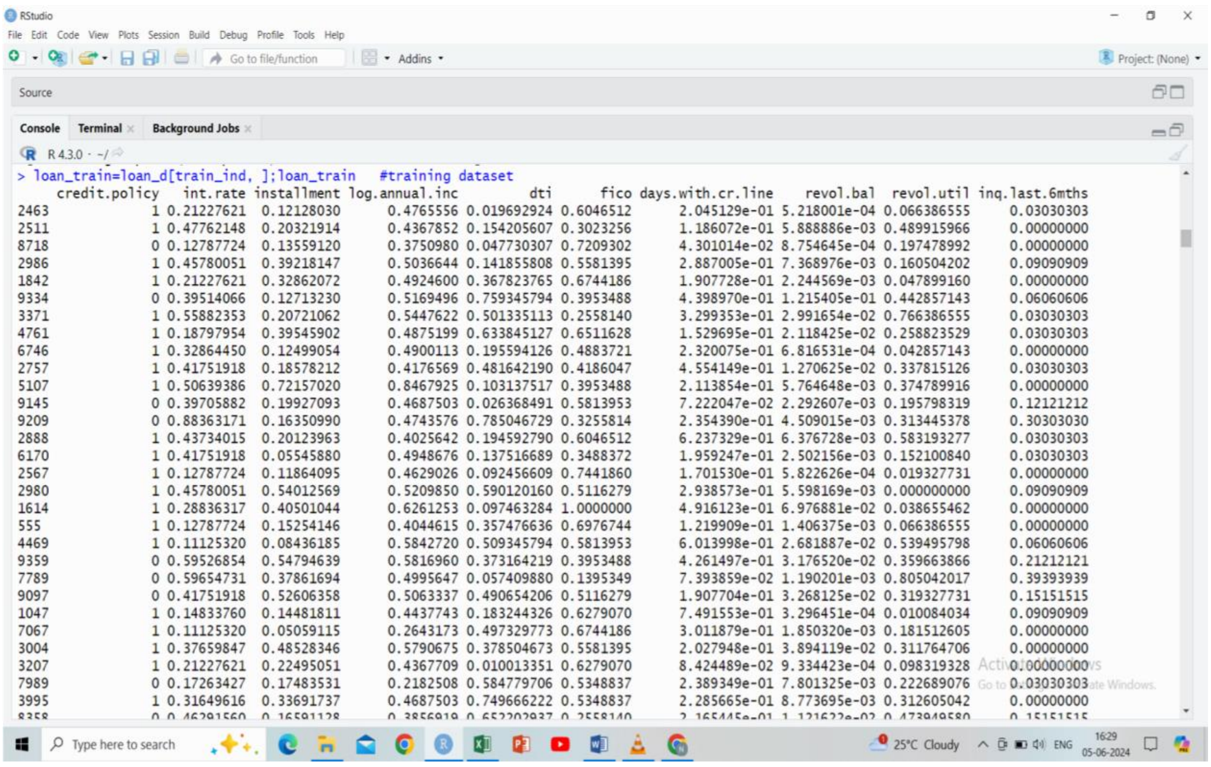
➢ summary(loan_d$revol.bal)



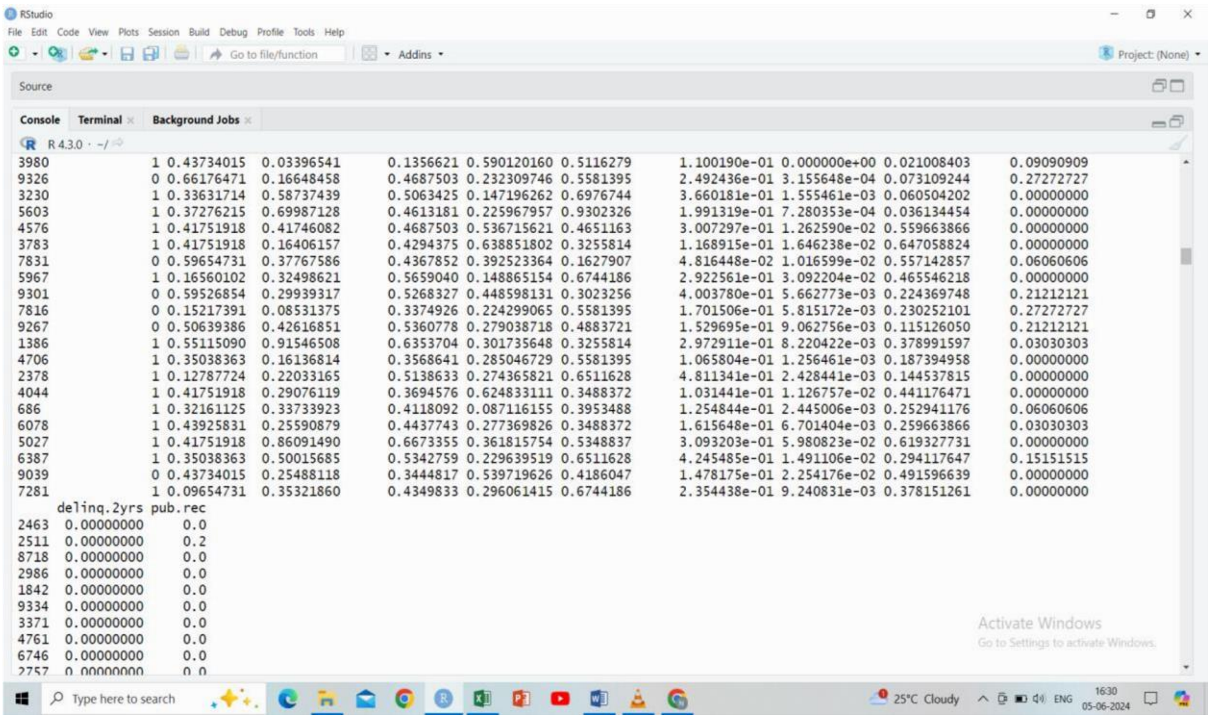
```
> table(loan$not.fully.paid)

   0    1 
8045 1533 
> #the target feature is coded as a factor
> loan$not.fully.paid=factor(loan$not.fully.paid,levels = c("0","1"),
+                            labels = c("Fully_paid","Not_fully_paid"))
> #Percentage calculation
> round(prop.table(table(loan$not.fully.paid))*100,1)

    Fully_paid Not_fully_paid 
            84             16 
> summary(loan[c("int.rate","installment","log.annual.inc")])#measurement scale should be approximately same
    int.rate       installment     log.annual.inc  
 Min.   :0.0600   Min.   : 15.67   Min.   : 7.548  
 1st Qu.:0.1039   1st Qu.:163.77   1st Qu.:10.558  
 Median :0.1221   Median :268.95   Median :10.929  
 Mean   :0.1226   Mean   :319.09   Mean   :10.932  
 3rd Qu.:0.1407   3rd Qu.:432.76   3rd Qu.:11.291  
 Max.   :0.2164   Max.   :940.14   Max.   :14.528  
> #nomralizing the variables
> normalise=function(x){
+   return((x-min(x))/(max(x)-min(x)))
+ }
> loan_d=as.data.frame(lapply(loan[1:12],normalise))
> summary(loan_d$revol.bal)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
0.00000 0.00264 0.00712 0.01401 0.01512 1.00000 
> #data preparation
> sample_size=floor(0.75*nrow(loan));sample_size
[1] 7183
```

#Data Preparation

- sample_size=floor(0.75*nrow(loan));sample_size
- set.seed(123)
- train_ind=sample(seq_len(nrow(loan)),size=sample_size);train_ind
- loan_train=loan_d[train_ind,];loan_train   **#Training Dataset**

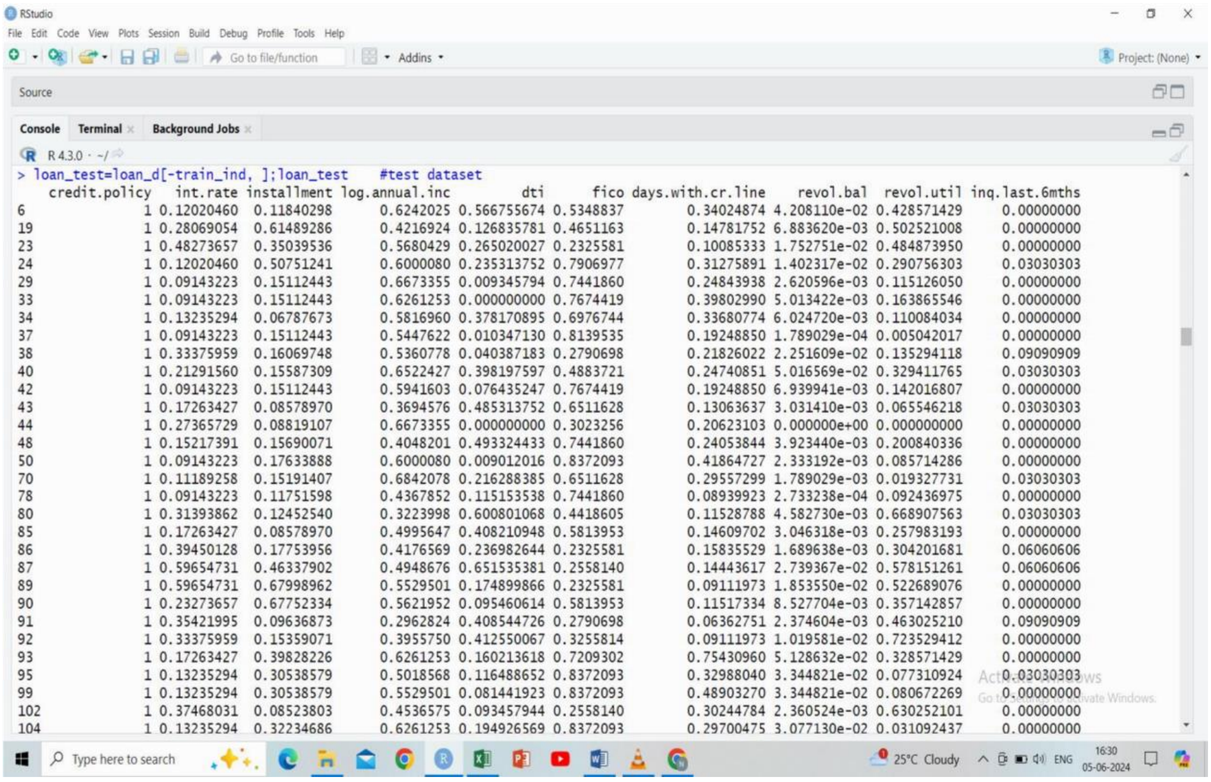


```
> loan_train=loan_d[train_ind, ];loan_train   #training dataset
     credit.policy   int.rate installment log.annual.inc         dti      fico days.with.cr.line    revol.bal  revol.util inq.last.6mths
2463             1 0.21227621  0.12128030      0.4765556 0.019692924 0.6046512      2.045129e-01 5.218001e-04 0.066386555     0.03030303
2511             1 0.47762148  0.20321914      0.4367852 0.154205607 0.3023256      1.186072e-01 5.888886e-03 0.489915966     0.00000000
8718             0 0.12787724  0.13559120      0.3750980 0.047730307 0.7209302      4.301014e-02 8.754645e-04 0.197478992     0.00000000
2986             1 0.45780051  0.39218147      0.5036644 0.141855808 0.5581395      2.887005e-01 7.368976e-03 0.160504202     0.09090909
1842             1 0.21227621  0.32862072      0.4924600 0.367823765 0.6744186      1.907728e-01 2.244569e-03 0.047899160     0.00000000
9334             0 0.39514066  0.12713230      0.5169496 0.759345794 0.3953488      4.398970e-01 1.215405e-01 0.442857143     0.06060606
3371             1 0.55882353  0.20721062      0.5447622 0.501335113 0.2558140      3.299353e-01 2.991654e-02 0.766386555     0.03030303
4761             1 0.18797954  0.39545902      0.4875199 0.633845127 0.6511628      1.529695e-01 2.118425e-02 0.258823529     0.03030303
6746             1 0.32864450  0.12499054      0.4900113 0.195594126 0.4883721      2.320075e-01 6.816531e-04 0.042857143     0.00000000
2757             1 0.41751918  0.18578212      0.4176569 0.481642190 0.4186047      4.554149e-01 1.270625e-02 0.337815126     0.03030303
5107             1 0.50639386  0.72157020      0.8467925 0.103137517 0.3953488      2.113854e-01 5.764648e-03 0.374789916     0.00000000
9145             0 0.39705882  0.19927093      0.4687503 0.026368491 0.5813953      7.222047e-02 2.292607e-03 0.195798319     0.12121212
9209             0 0.88363171  0.16350990      0.4743576 0.785046729 0.3255814      2.354390e-01 4.509015e-03 0.313445378     0.30303030
2888             1 0.43734015  0.20123963      0.4025642 0.194592790 0.6046512      6.237329e-01 6.376728e-03 0.583193277     0.03030303
6170             1 0.41751918  0.05545880      0.4948676 0.137516689 0.3488372      1.959247e-01 2.502156e-03 0.152100840     0.03030303
2567             1 0.12787724  0.11864095      0.4629026 0.092456609 0.7441860      1.701530e-01 5.822626e-04 0.019327731     0.00000000
2980             1 0.45780051  0.54012569      0.5209850 0.590120160 0.5116279      2.938573e-01 5.598169e-03 0.000000000     0.09090909
1614             1 0.28836317  0.40501044      0.6261253 0.097463284 1.0000000      4.916123e-01 6.976881e-02 0.038655462     0.00000000
555              1 0.12787724  0.15254146      0.4044615 0.357476636 0.6976744      1.219909e-01 1.406375e-03 0.066386555     0.00000000
4469             1 0.11125320  0.08436185      0.5842720 0.509345794 0.5813953      6.013998e-01 2.681887e-02 0.539495798     0.06060606
9359             0 0.59526854  0.54794639      0.5816960 0.373164219 0.3953488      4.261497e-01 3.176520e-02 0.359663866     0.21212121
7789             0 0.59654731  0.37861694      0.4995647 0.057409880 0.1395349      7.393859e-02 1.190201e-03 0.805042017     0.39393939
9097             0 0.41751918  0.52606358      0.5063337 0.490654206 0.5116279      1.907704e-01 3.268125e-02 0.319327731     0.15151515
1047             1 0.14833760  0.14481811      0.4437743 0.183244326 0.6279070      7.491553e-01 3.296451e-04 0.010084034     0.09090909
7067             1 0.11125320  0.05059115      0.2643173 0.497329773 0.6744186      3.011879e-01 1.850320e-03 0.181512605     0.00000000
3004             1 0.37659847  0.48528346      0.5790675 0.378504673 0.5581395      2.027948e-01 3.894119e-02 0.311764706     0.00000000
3207             1 0.21227621  0.22495051      0.4367709 0.010013351 0.6279070      8.424489e-02 9.334423e-04 0.098319328     0.00000000
7989             0 0.17263427  0.17483531      0.2182508 0.584779706 0.5348837      2.389349e-01 7.801325e-03 0.222689076     0.03030303
3995             1 0.31649616  0.33691737      0.4687503 0.749666222 0.5348837      2.285665e-01 8.773695e-03 0.312605042     0.00000000
```

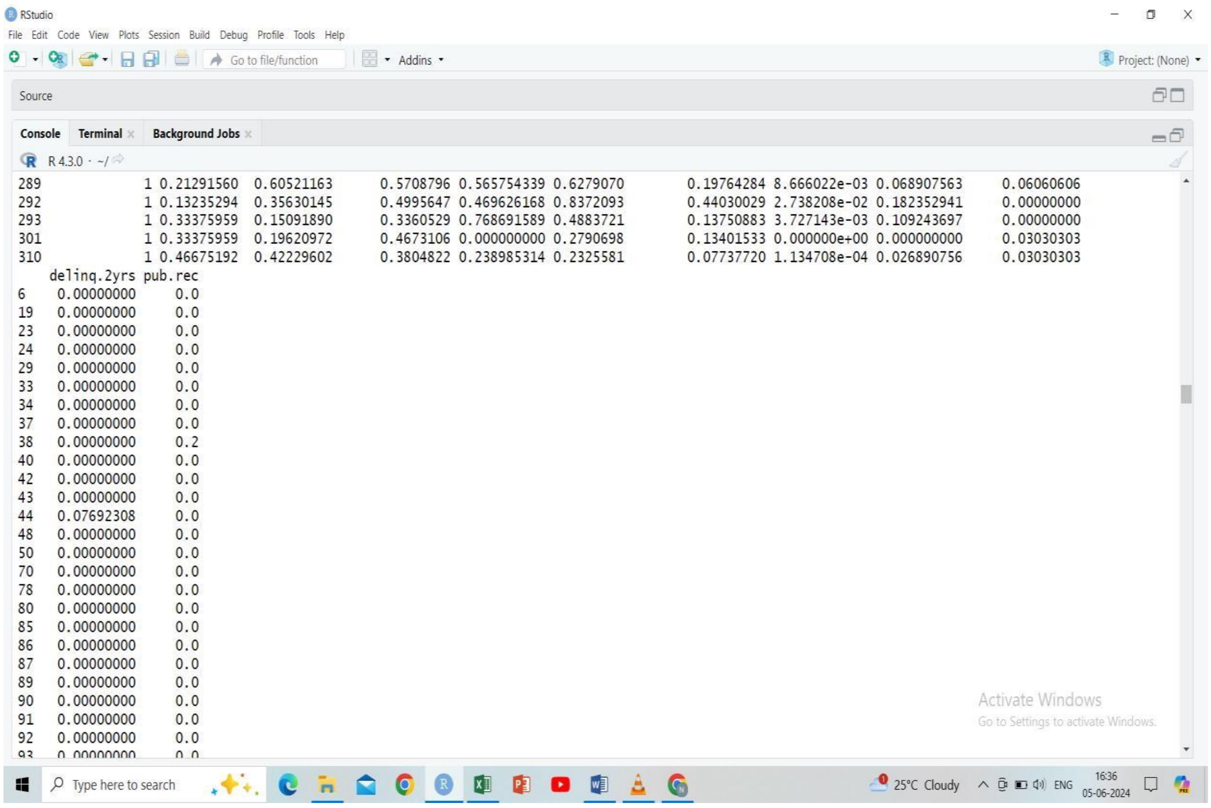


```
3980            1 0.43734015  0.03396541      0.1356621 0.590120160 0.5116279     1.100190e-01 0.000000e+00 0.021008403     0.09090909
9326            0 0.66176471  0.16648458      0.4687503 0.232309746 0.5581395     2.492436e-01 3.155648e-04 0.073109244     0.27272727
3230            1 0.33631714  0.58737439      0.5063425 0.147196262 0.6976744     3.660181e-01 1.555461e-03 0.060504202     0.00000000
5603            1 0.37276215  0.69987128      0.4613181 0.225967957 0.9302326     1.991319e-01 7.280353e-04 0.036134454     0.00000000
4576            1 0.41751918  0.41746082      0.4687503 0.536715621 0.4651163     3.007297e-01 1.262590e-02 0.559663866     0.00000000
3783            1 0.41751918  0.16406157      0.4294375 0.638851802 0.3255814     1.168915e-01 1.646238e-02 0.647058824     0.00000000
7831            0 0.59654731  0.37767586      0.4367852 0.392523364 0.1627907     4.816448e-02 1.016599e-02 0.557142857     0.06060606
5967            1 0.16560102  0.32498621      0.5659040 0.148865154 0.6744186     2.922561e-01 3.092204e-02 0.465546218     0.00000000
9301            0 0.59526854  0.29939317      0.5268327 0.448598131 0.3023256     4.003780e-01 5.662773e-03 0.224369748     0.21212121
7816            0 0.15217391  0.08531375      0.3374926 0.224299065 0.5581395     1.701506e-01 5.815172e-03 0.230252101     0.27272727
9267            0 0.50639386  0.42616851      0.5360778 0.279038718 0.4883721     1.529695e-01 9.062756e-03 0.115126050     0.21212121
1386            1 0.55115090  0.91546508      0.6353704 0.301735648 0.3255814     2.972911e-01 8.220422e-03 0.378991597     0.03030303
4706            1 0.35038363  0.16136814      0.3568641 0.285046729 0.5581395     1.065804e-01 1.256461e-03 0.187394958     0.00000000
2378            1 0.12787724  0.22033165      0.5138633 0.274365821 0.6511628     4.811341e-01 2.428441e-03 0.144537815     0.00000000
4044            1 0.41751918  0.29076119      0.3694576 0.624833111 0.3488372     1.031441e-01 1.126757e-02 0.441176471     0.00000000
686             1 0.32161125  0.33733923      0.4118092 0.087116155 0.3953488     1.254844e-01 2.445006e-03 0.252941176     0.06060606
6078            1 0.43925831  0.25590879      0.4437743 0.277369826 0.3488372     1.615648e-01 6.701404e-03 0.259663866     0.03030303
5027            1 0.41751918  0.86091490      0.6673355 0.361815754 0.5348837     3.093203e-01 5.980823e-02 0.619327731     0.00000000
6387            1 0.35038363  0.50015685      0.5342759 0.229639519 0.6511628     4.245485e-01 1.491106e-02 0.294117647     0.15151515
9039            0 0.43734015  0.25488118      0.3444817 0.539719626 0.4186047     1.478175e-01 2.254176e-02 0.491596639     0.00000000
7281            1 0.09654731  0.35321860      0.4349833 0.296061415 0.6744186     2.354438e-01 9.240831e-03 0.378151261     0.00000000
     delinq.2yrs pub.rec
2463  0.00000000     0.0
2511  0.00000000     0.2
8718  0.00000000     0.0
2986  0.00000000     0.0
1842  0.00000000     0.0
9334  0.00000000     0.0
3371  0.00000000     0.0
4761  0.00000000     0.0
6746  0.00000000     0.0
2757  0.00000000     0.0
```

- loan_test=loan_d[-train_ind,];loan_test   **#Test Dataset**

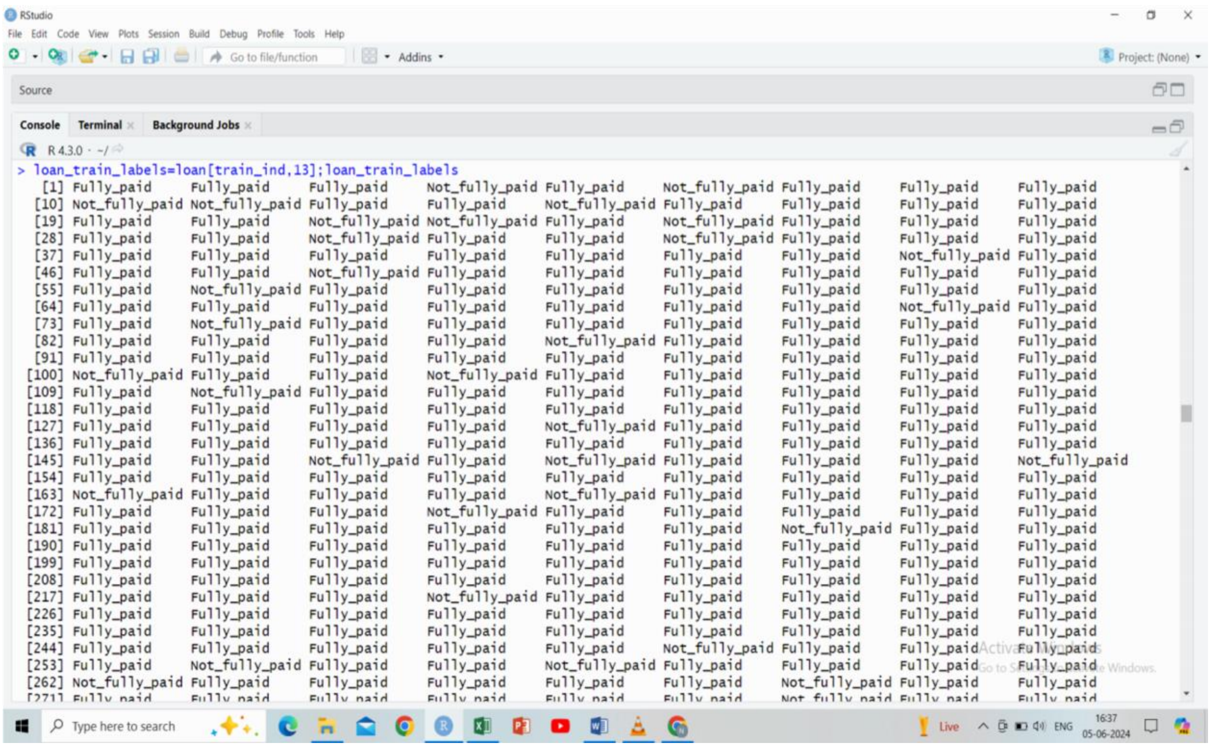


```
> loan_test=loan_d[-train_ind, ];loan_test    #test dataset
    credit.policy   int.rate installment log.annual.inc         dti      fico days.with.cr.line     revol.bal  revol.util inq.last.6mths
6               1 0.12020460  0.11840298      0.6242025 0.566755674 0.5348837        0.34024874 4.208110e-02 0.428571429     0.00000000
19              1 0.28069054  0.61489286      0.4216924 0.126835781 0.4651163        0.14781752 6.883620e-03 0.502521008     0.00000000
23              1 0.48273657  0.35039536      0.5680429 0.265020027 0.2325581        0.10085333 1.752751e-02 0.484873950     0.00000000
24              1 0.12020460  0.50751241      0.6000080 0.235313752 0.7906977        0.31275891 1.402317e-02 0.290756303     0.03030303
29              1 0.09143223  0.15112443      0.6673355 0.009345794 0.7441860        0.24843938 2.620596e-03 0.115126050     0.00000000
33              1 0.09143223  0.15112443      0.6261253 0.000000000 0.7674419        0.39802990 5.013422e-03 0.163865546     0.00000000
34              1 0.13235294  0.06787673      0.5816960 0.378170895 0.6976744        0.33680774 6.024720e-03 0.110084034     0.00000000
37              1 0.09143223  0.15112443      0.5447622 0.010347130 0.8139535        0.19248850 1.789029e-04 0.005042017     0.00000000
38              1 0.33375959  0.16069748      0.5360778 0.040387183 0.2790698        0.21826022 2.251609e-02 0.135294118     0.09090909
40              1 0.21291560  0.15587309      0.6522427 0.398197597 0.4883721        0.24740851 5.016569e-02 0.329411765     0.03030303
42              1 0.09143223  0.15112443      0.5941603 0.076435247 0.7674419        0.19248850 6.939941e-03 0.142016807     0.00000000
43              1 0.17263427  0.08578970      0.3694576 0.485313752 0.6511628        0.13063637 3.031410e-03 0.065546218     0.03030303
44              1 0.27365729  0.08819107      0.6673355 0.000000000 0.3023256        0.20623103 0.000000e+00 0.000000000     0.00000000
48              1 0.15217391  0.15690071      0.4048201 0.493324433 0.7441860        0.24053844 3.923440e-03 0.200840336     0.00000000
50              1 0.09143223  0.17633888      0.6000080 0.009012016 0.8372093        0.41864727 2.333192e-03 0.085714286     0.00000000
70              1 0.11189258  0.15191407      0.6842078 0.216288385 0.6511628        0.29557299 1.789029e-03 0.019327731     0.03030303
78              1 0.09143223  0.11751598      0.4367852 0.115153538 0.7441860        0.08939923 2.733238e-04 0.092436975     0.00000000
80              1 0.31393862  0.12452540      0.3223998 0.600801068 0.4418605        0.11528788 4.582730e-03 0.668907563     0.03030303
85              1 0.17263427  0.08578970      0.4995647 0.408210948 0.5813953        0.14609702 3.046318e-03 0.257983193     0.00000000
86              1 0.39450128  0.17753956      0.4176569 0.236982644 0.2325581        0.15835529 1.689638e-03 0.304201681     0.06060606
87              1 0.59654731  0.46337902      0.4948676 0.651535381 0.2558140        0.14443617 2.739367e-02 0.578151261     0.06060606
89              1 0.59654731  0.67998962      0.5529501 0.174899866 0.2325581        0.09111973 1.853550e-02 0.522689076     0.00000000
90              1 0.23273657  0.67752334      0.5621952 0.095460614 0.5813953        0.11517334 8.527704e-03 0.357142857     0.00000000
91              1 0.35421995  0.09636873      0.2962824 0.408544726 0.2790698        0.06362751 2.374604e-03 0.463025210     0.09090909
92              1 0.33375959  0.15359071      0.3955750 0.412550067 0.3255814        0.09111973 1.019581e-02 0.723529412     0.00000000
93              1 0.17263427  0.39828226      0.6261253 0.160213618 0.7209302        0.75430960 5.128632e-02 0.328571429     0.00000000
95              1 0.13235294  0.30538579      0.5018568 0.116488652 0.8372093        0.32988040 3.344821e-02 0.077310924     0.03030303
99              1 0.13235294  0.30538579      0.5529501 0.081441923 0.8372093        0.48903270 3.344821e-02 0.080672269     0.00000000
102             1 0.37468031  0.08523803      0.4536575 0.093457944 0.2558140        0.30244784 2.360524e-03 0.630252101     0.00000000
104             1 0.13235294  0.32234686      0.6261253 0.194926569 0.8372093        0.29700475 3.077130e-02 0.031092437     0.00000000
```



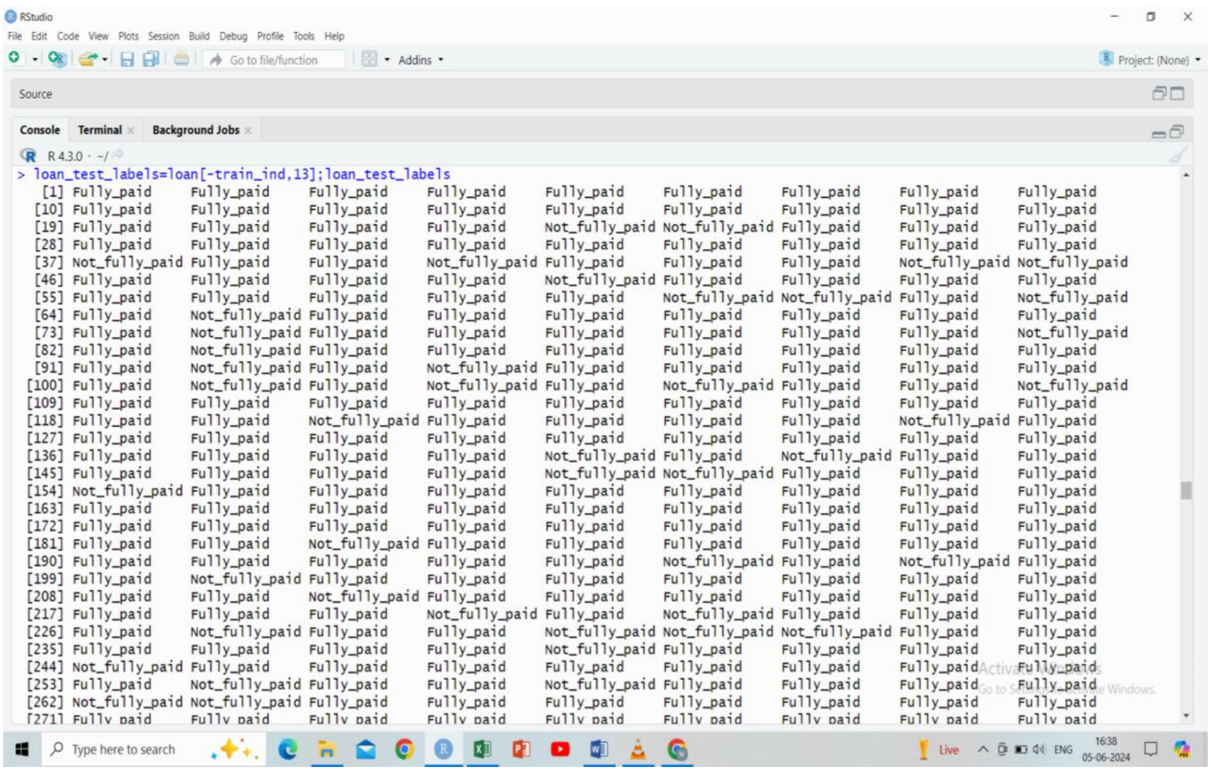
```
289              1 0.21291560  0.60521163       0.5708796 0.565754339 0.6279070        0.19764284 8.666022e-03 0.068907563      0.06060606
292              1 0.13235294  0.35630145       0.4995647 0.469626168 0.8372093        0.44030029 2.738208e-02 0.182352941      0.00000000
293              1 0.33375959  0.15091890       0.3360529 0.768691589 0.4883721        0.13750883 3.727143e-03 0.109243697      0.00000000
301              1 0.33375959  0.19620972       0.4673106 0.000000000 0.2790698        0.13401533 0.000000e+00 0.000000000      0.03030303
310              1 0.46675192  0.42229602       0.3804822 0.238985314 0.2325581        0.07737720 1.134708e-04 0.026890756      0.03030303
    delinq.2yrs pub.rec
6    0.00000000     0.0
19   0.00000000     0.0
23   0.00000000     0.0
24   0.00000000     0.0
29   0.00000000     0.0
33   0.00000000     0.0
34   0.00000000     0.0
37   0.00000000     0.0
38   0.00000000     0.2
40   0.00000000     0.0
42   0.00000000     0.0
43   0.00000000     0.0
44   0.07692308     0.0
48   0.00000000     0.0
50   0.00000000     0.0
70   0.00000000     0.0
78   0.00000000     0.0
80   0.00000000     0.0
85   0.00000000     0.0
86   0.00000000     0.0
87   0.00000000     0.0
89   0.00000000     0.0
90   0.00000000     0.0
91   0.00000000     0.0
92   0.00000000     0.0
93   0.00000000     0.0
```

- loan_train_labels=loan[train_ind,13];loan_train_labels



```
> loan_train_labels=loan[train_ind,13];loan_train_labels
  [1] Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid
 [10] Not_fully_paid Not_fully_paid Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [19] Fully_paid     Fully_paid     Not_fully_paid Not_fully_paid Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid
 [28] Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid
 [37] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid
 [46] Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [55] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [64] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid
 [73] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [82] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [91] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[100] Not_fully_paid Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[109] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[118] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[127] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid
[136] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[145] Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Not_fully_paid
[154] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[163] Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid
[172] Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[181] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid
[190] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[199] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[208] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[217] Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[226] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[235] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[244] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid
[253] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid
[262] Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid
[271] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid
```

- loan_test_labels=loan[-train_ind,13];loan_test_labels



```
> loan_test_labels=loan[-train_ind,13];loan_test_labels
  [1] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [10] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [19] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Not_fully_paid Fully_paid     Fully_paid     Fully_paid
 [28] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [37] Not_fully_paid Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Not_fully_paid
 [46] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [55] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Not_fully_paid Fully_paid     Not_fully_paid
 [64] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [73] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid
 [82] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
 [91] Fully_paid     Not_fully_paid Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[100] Fully_paid     Not_fully_paid Fully_paid     Not_fully_paid Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Not_fully_paid
[109] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[118] Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid
[127] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[136] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Not_fully_paid Fully_paid     Fully_paid
[145] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Not_fully_paid Fully_paid     Fully_paid     Fully_paid
[154] Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[163] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[172] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[181] Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[190] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Not_fully_paid Fully_paid
[199] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[208] Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[217] Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid
[226] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Not_fully_paid Not_fully_paid Not_fully_paid Fully_paid     Fully_paid
[235] Fully_paid     Fully_paid     Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid
[244] Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[253] Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid
[262] Not_fully_paid Not_fully_paid Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid     Fully_paid
[271] Fully paid     Fully paid     Fully paid     Fully paid     Fully paid     Fully paid     Fully paid     Fully paid     Fully paid
```

# Initialize a vector to store accuracy for each k

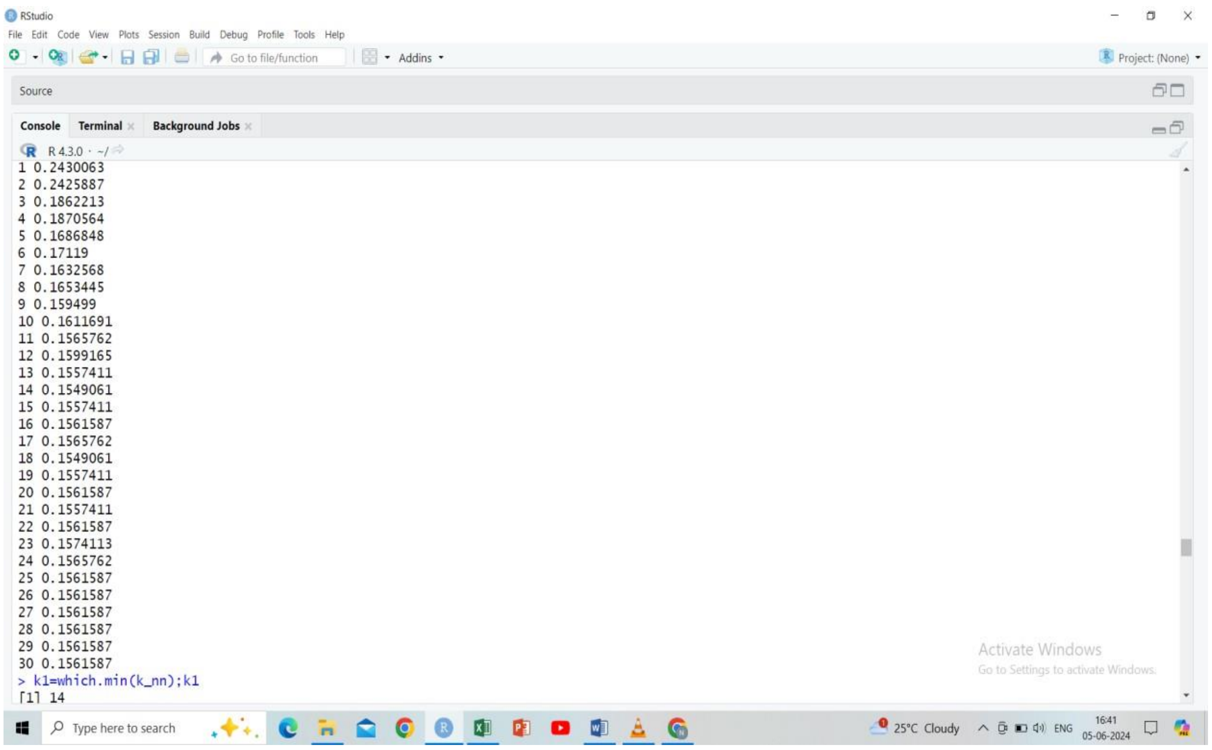
- ➢ k_values <- 1:30
- ➢ k_nn=0;
- ➢ accuracy <- numeric(length(k_values))
- ➢ library(class)

# Loop through each k value and perform k-NN

```
➢ for (k in k_values) {
        pred <- knn(train = loan_train, test = loan_test, cl = loan_train_labels, k);
incorrect=sum(loan_test_labels!=pred)
mis_rate=sum(incorrect)/length(loan_test_labels)        cat(k,mis_rate,"\n")
k_nn[k]=mis_rate

        accuracy[k] <- sum(pred == loan_test_labels) / length(pred)
  }
```

#Choosing the k which has minimum misclassification rate

➢ k1=which.min(k_nn);k1

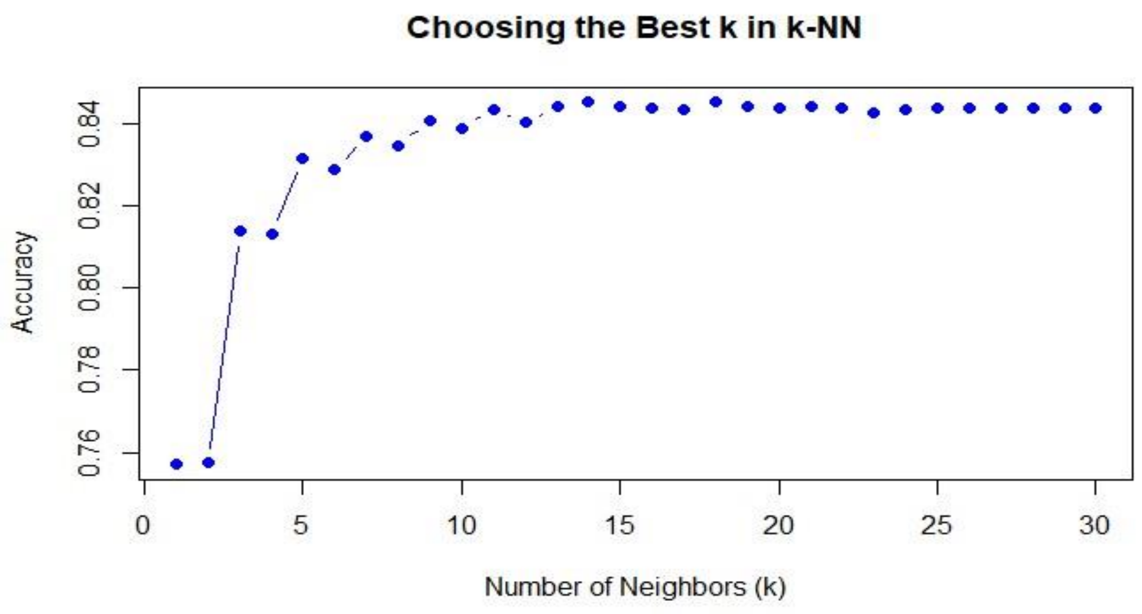

```
1 0.2430063
2 0.2425887
3 0.1862213
4 0.1870564
5 0.1686848
6 0.17119
7 0.1632568
8 0.1653445
9 0.159499
10 0.1611691
11 0.1565762
12 0.1599165
13 0.1557411
14 0.1549061
15 0.1557411
16 0.1561587
17 0.1565762
18 0.1549061
19 0.1557411
20 0.1561587
21 0.1557411
22 0.1561587
23 0.1574113
24 0.1565762
25 0.1561587
26 0.1561587
27 0.1561587
28 0.1561587
29 0.1561587
30 0.1561587
> k1=which.min(k_nn);k1
[1] 14
```

# Plot of Accuracy v/s K

➢ plot(k_values, accuracy, type = "b", col = "blue", pch = 19,

xlab = "Number of Neighbors (k)", ylab = "Accuracy",

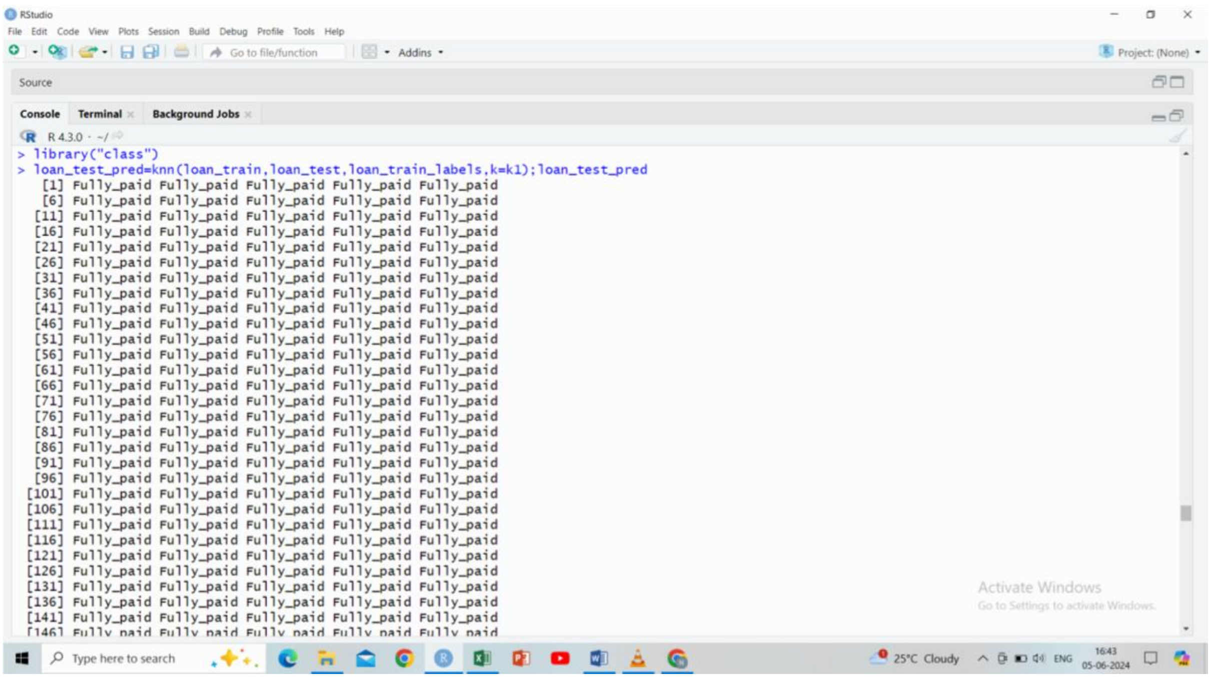
main = "Choosing the Best k in k-NN")



#Training a Model on the Data

➢ library(class)

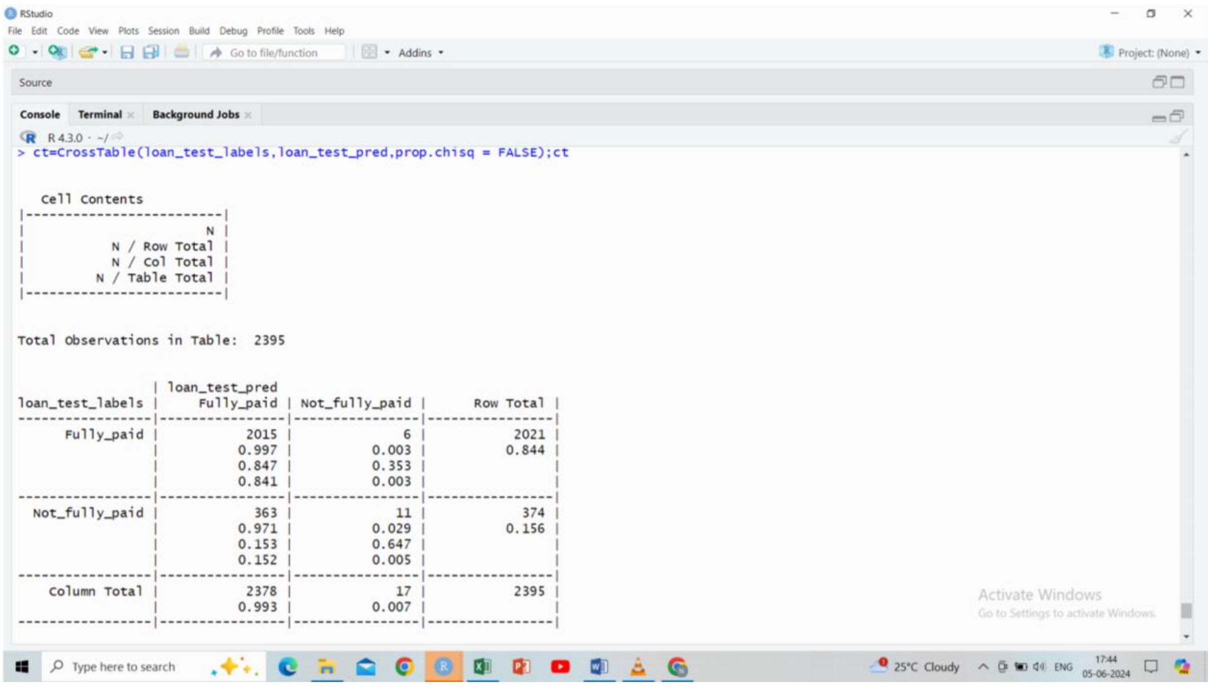
➢ loan_test_pred=knn(loan_train,loan_test,loan_train_labels,k=k1);loan_test_pred



```
> library("class")
> loan_test_pred=knn(loan_train,loan_test,loan_train_labels,k=k1);loan_test_pred
   [1] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
   [6] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [11] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [16] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [21] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [26] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [31] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [36] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [41] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [46] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [51] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [56] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [61] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [66] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [71] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [76] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [81] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [86] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [91] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
  [96] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [101] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [106] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [111] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [116] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [121] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [126] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [131] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [136] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [141] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
 [146] Fully_paid Fully_paid Fully_paid Fully_paid Fully_paid
```

#Evaluating Model Performance

- library(gmodels)
- ct= CrossTable(loan_test_labels, loan_test_pred, prop.chisq = FALSE);ct



```
> ct=CrossTable(loan_test_labels,loan_test_pred,prop.chisq = FALSE);ct


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  2395


                 | loan_test_pred 
loan_test_labels |     Fully_paid | Not_fully_paid |      Row Total | 
-----------------|----------------|----------------|----------------|
      Fully_paid |           2015 |              6 |           2021 | 
                 |          0.997 |          0.003 |          0.844 | 
                 |          0.847 |          0.353 |                | 
                 |          0.841 |          0.003 |                | 
-----------------|----------------|----------------|----------------|
  Not_fully_paid |            363 |             11 |            374 | 
                 |          0.971 |          0.029 |          0.156 | 
                 |          0.153 |          0.647 |                | 
                 |          0.152 |          0.005 |                | 
-----------------|----------------|----------------|----------------|
    Column Total |           2378 |             17 |           2395 | 
                 |          0.993 |          0.007 |                | 
-----------------|----------------|----------------|----------------|
```

- Counts=ct$t; ➢ Counts

	Y	
x	Fully_paid	Not_fully_paid
Fully_paid	2015	6
Not_fully_paid	363	11

- TP=counts[1,1]; #**True Positives**

- TP
  2015
- TN=counts[2,2]; #**True Negatives**
- TN
  11
- FP=counts[1,2]; # **False Positives**
- FP
  6
- FN=counts[2,1]; # **False Negatives**
- FN
  363

Interpretation of Cross-Table:

Cross-Table in R is a powerful tool for exploring and analyzing the relationships between categorical variables. By offering detailed statistics and proportions, it helps in understanding the underlying patterns and dependencies within the data. This function is particularly useful for preliminary data analysis and hypothesis testing in categorical data analysis.

- The top left cell indicates True Positive (TP), means, person was fully paid and the k-NN correctly identified it as such
- The bottom right cell indicates True Negative (TN), means, person was not fully paid and k-NN correctly identified it as such
- The bottom left cell indicates False Negative (FN), means, person was not fully paid and k-NN identified it as fully paid
- The top right cell indicates False Positive (FP), means, person is fully paid and the k-NN identified it as not fully paid

6. EVALUATION METRICS

Accuracy: Accuracy is a common metric used to evaluate the performance of classification models in machine learning. It measures the proportion of correctly predicted instances out of the total instances in the dataset.

Accuracy is defined as the ratio of the number of correct predictions to the total number of predictions.

Accuracy = (TP+TN) / (TP+TN+FP+FN)

- accuracy <- (TP+TN) / (TP+TN+FP+FN)
- accuracy
  [1] 0.8438413

Precision: Precision is a performance metric for classification models, especially useful in scenarios with imbalanced classes. It measures the accuracy of positive predictions, specifically the ratio of true positives to the sum of true positives and false positives.

Precision = TP / (TP+FP)

- precision <- TP / (TP + FP)
- precision
  [1] 0.9955468

Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of actual positives that are correctly identified by the model. It is particularly useful in situations where it is important to capture as many positives as possible.

Recall = TP / (TP+FN)

- ➢ recall <- TP / (TP+FN)
- ➢ recall
  [1] 0.8464451

F1- Score: The F1-Score is a performance metric that combines precision and recall into a single measure. It is the harmonic mean of precision and recall, providing a balance between the two metrics.

F1-Score = 2 * (precision * recall) / (precision + recall)

- ➢ f1_score <- 2 * (precision * recall) / (precision + recall)
- ➢ f1_score ➢ [1] 0.9149613

conclusion:

- Accuracy is 84%, it means that 84% of the predictions made by the model are correct. In other words, out of all the instances it predicted, 84% were classified correctly as either positive or negative. Since the accuracy is relatively high, it suggests that the model's performance is very reliable.
- Precision is 99.5%, Out of all the loans predicted as repaid, 99.5% were actually approved. This high precision indicates a low number of false positives, which means the model is correctly predicting loans as repaid.

REFERENCE:

[1] 2018. Kumar Arun. Loan Approval Prediction based on Machine Learning Approach.

[2] 2019. E. Chandra Blessie. Exploring the Machine Learning Algorithm for Prediction of Loan Sanctioning process.

[3] 2019. Loan Prediction using Machine Learning.

[4] 2020. Kshitiz Gautam. Loan Prediction using Decision Tree and Random Forest.