Name :- Kallam Charan Reddy **Reg no :- 12210904**

Section :- K21BP **Roll No :- A33**

Project Report: Loan Status Prediction Using Machine Learning Classification Algorithms

1. Introduction

The aim of this project is to build a predictive model to classify loan applications based on demographic and financial information. Accurately predicting loan outcomes, such as approval status, can aid financial institutions in minimizing risk and improving decision-making. In this report, we evaluate four classification algorithms—K-Nearest Neighbors (KNN), Naive Bayes (NB), Support Vector Machines (SVM), and Decision Trees (DT)—and compare their performance in terms of accuracy and efficiency.

2. Dataset Overview

The dataset used, Datasetoncrime.csv, consists of various attributes of loan applicants, including:

- **Demographic Information**: Age, income, employment status, etc.
- **Financial Information**: Loan amount, credit score, and any additional factors affecting loan status.
- **Target Variable**: loan_status represents the classification label, with possible values of:
    - **Normal**: Standard or approved loans
    - **Suspect**: Applications that need additional review
    - **Pathologic**: High-risk applications likely to be rejected

Data Preprocessing

The data preprocessing steps included:

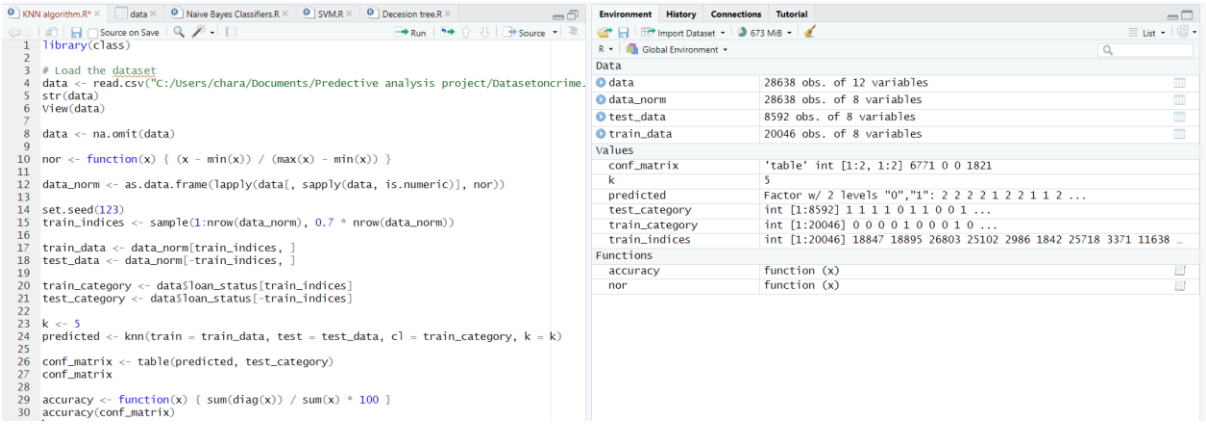
1. **Handling Missing Values**: Checking and addressing any missing data points.
2. **Converting Data Types**: Ensuring all character variables are converted to factors for categorical classification.
3. **Scaling Features**: For KNN and SVM, scaling numeric features was essential to standardize data across varying ranges
4. **Train-Test Split**: Dividing the data into 80% for training and 20% for testing

.

3. Algorithms Used

3.1 K-Nearest Neighbors (KNN)

The KNN algorithm classifies instances based on the majority class of their K-nearest neighbors in the feature space. In this project, we used k = 13, chosen after testing different values to balance bias and variance. This algorithm works well for smaller datasets but can be computationally intensive for larger ones.

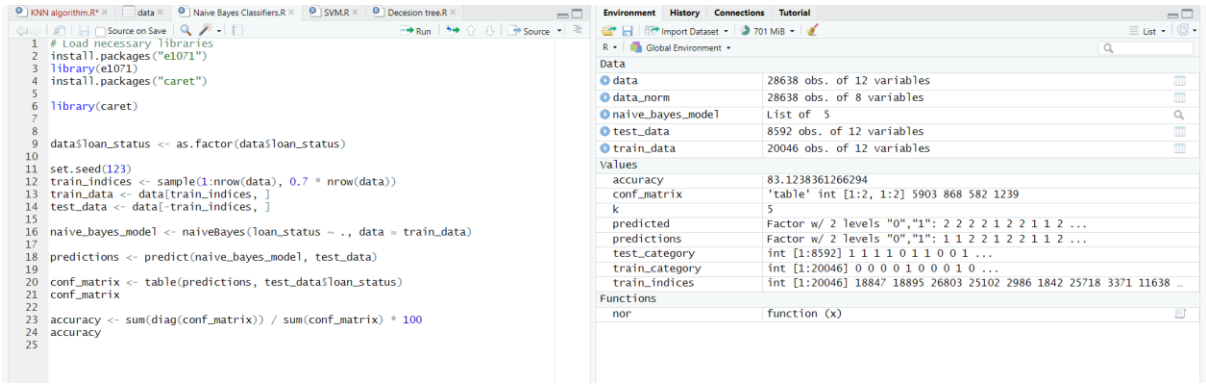


```r
library(class)

# Load the dataset
data <- read.csv("C:/Users/chara/Documents/Predective analysis project/Datasetoncrime.
str(data)
View(data)

data <- na.omit(data)

nor <- function(x) { (x - min(x)) / (max(x) - min(x)) }

data_norm <- as.data.frame(lapply(data[, sapply(data, is.numeric)], nor))

set.seed(123)
train_indices <- sample(1:nrow(data_norm), 0.7 * nrow(data_norm))

train_data <- data_norm[train_indices, ]
test_data <- data_norm[-train_indices, ]

train_category <- data$loan_status[train_indices]
test_category <- data$loan_status[-train_indices]

k <- 5
predicted <- knn(train = train_data, test = test_data, cl = train_category, k = k)

conf_matrix <- table(predicted, test_category)
conf_matrix

accuracy <- function(x) { sum(diag(x)) / sum(x) * 100 }
accuracy(conf_matrix)
```

Environment		History	Connections	Tutorial
Import Dataset	673 MiB	List		
Global Environment				
Data				
data	28638 obs. of 12 variables			
data_norm	28638 obs. of 8 variables			
test_data	8592 obs. of 8 variables			
train_data	20046 obs. of 8 variables			
Values				
conf_matrix	'table' int [1:2, 1:2] 6771 0 0 1821			
k	5			
predicted	Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 1 1 2 ...			
test_category	int [1:8592] 1 1 1 1 0 1 1 0 0 1 ...			
train_category	int [1:20046] 0 0 0 0 1 0 0 0 1 0 ...			
train_indices	int [1:20046] 18847 18895 26803 25102 2986 1842 25718 3371 11638 ...			
Functions				
accuracy	function (x)			
nor	function (x)			

3.2 Naive Bayes (NB)

The Naive Bayes classifier is a probabilistic model that assumes independence between features. This algorithm was chosen for its simplicity and speed, particularly effective with categorical data. The model assigns a probability to each class based on the training data and uses this probability to classify test data.



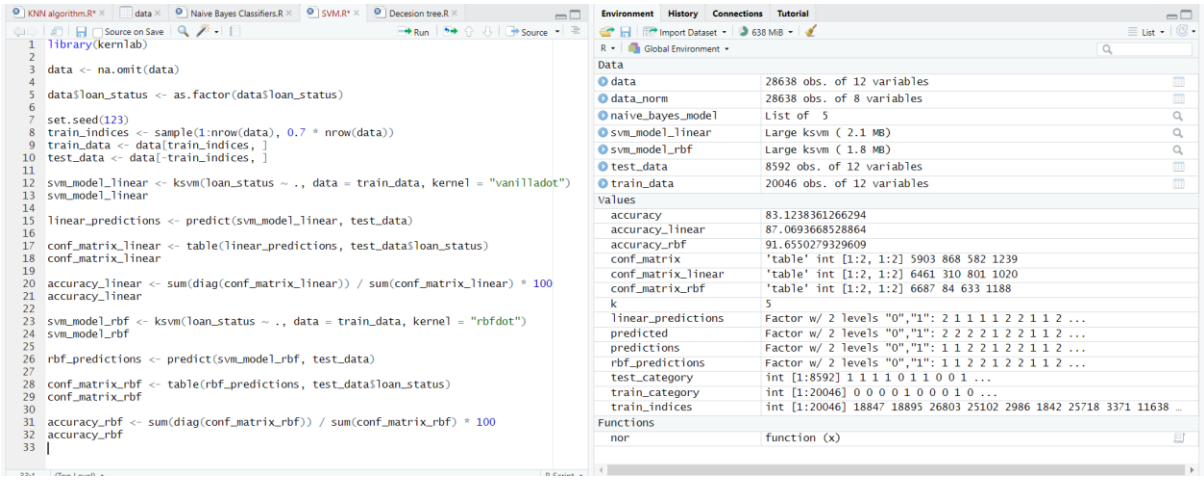
```r
# Load necessary libraries
install.packages("e1071")
library(e1071)
install.packages("caret")

library(caret)


data$loan_status <- as.factor(data$loan_status)

set.seed(123)
train_indices <- sample(1:nrow(data), 0.7 * nrow(data))
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]

naive_bayes_model <- naiveBayes(loan_status ~ ., data = train_data)

predictions <- predict(naive_bayes_model, test_data)

conf_matrix <- table(predictions, test_data$loan_status)
conf_matrix

accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix) * 100
accuracy
```

Environment		History	Connections	Tutorial
Import Dataset	701 MiB	List		
Global Environment				
Data				
data	28638 obs. of 12 variables			
data_norm	28638 obs. of 8 variables			
naive_bayes_model	List of 5			
test_data	8592 obs. of 12 variables			
train_data	20046 obs. of 12 variables			
Values				
accuracy	83.1238361266294			
conf_matrix	'table' int [1:2, 1:2] 5903 868 582 1239			
k	5			
predicted	Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 1 1 2 ...			
predictions	Factor w/ 2 levels "0","1": 1 1 2 2 1 2 2 1 1 2 ...			
test_category	int [1:8592] 1 1 1 1 0 1 1 0 0 1 ...			
train_category	int [1:20046] 0 0 0 0 1 0 0 0 1 0 ...			
train_indices	int [1:20046] 18847 18895 26803 25102 2986 1842 25718 3371 11638 ...			
Functions				
nor	function (x)			

3.3 Support Vector Machine (SVM)

SVM finds an optimal hyperplane that separates classes with maximum margin. Two types of kernels were tested:

- **Linear Kernel**: Used for linearly separable data.
- **Radial Basis Function (RBF) Kernel**: Applied for non-linear boundaries in complex data.



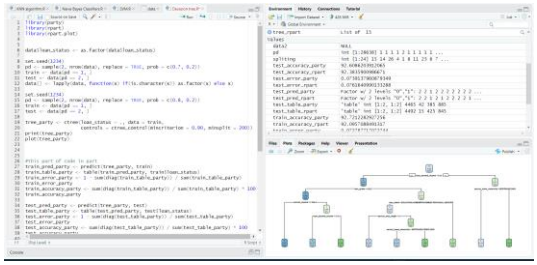
```
library(kernlab)

data <- na.omit(data)

data$loan_status <- as.factor(data$loan_status)

set.seed(123)
train_indices <- sample(1:nrow(data), 0.7 * nrow(data))
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]

svm_model_linear <- ksvm(loan_status ~ ., data = train_data, kernel = "vanilladot")
svm_model_linear

linear_predictions <- predict(svm_model_linear, test_data)

conf_matrix_linear <- table(linear_predictions, test_data$loan_status)
conf_matrix_linear

accuracy_linear <- sum(diag(conf_matrix_linear)) / sum(conf_matrix_linear) * 100
accuracy_linear

svm_model_rbf <- ksvm(loan_status ~ ., data = train_data, kernel = "rbfdot")
svm_model_rbf

rbf_predictions <- predict(svm_model_rbf, test_data)

conf_matrix_rbf <- table(rbf_predictions, test_data$loan_status)
conf_matrix_rbf

accuracy_rbf <- sum(diag(conf_matrix_rbf)) / sum(conf_matrix_rbf) * 100
accuracy_rbf
```

Environment	
Data	
data	28638 obs. of 12 variables
data_norm	28638 obs. of 8 variables
naive_bayes_model	List of 5
svm_model_linear	Large ksvm (2.1 MB)
svm_model_rbf	Large ksvm (1.8 MB)
test_data	8592 obs. of 12 variables
train_data	20046 obs. of 12 variables
Values	
accuracy	83.1238361266294
accuracy_linear	87.0693668528864
accuracy_rbf	91.6550279329609
conf_matrix	'table' int [1:2, 1:2] 5903 868 582 1239
conf_matrix_linear	'table' int [1:2, 1:2] 6461 310 801 1020
conf_matrix_rbf	'table' int [1:2, 1:2] 6687 84 633 1188
k	5
linear_predictions	Factor w/ 2 levels "0","1": 2 1 1 1 1 2 2 1 1 2 ...
predicted	Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 1 1 2 ...
predictions	Factor w/ 2 levels "0","1": 1 1 2 2 1 2 2 1 1 2 ...
rbf_predictions	Factor w/ 2 levels "0","1": 1 1 2 2 1 2 2 1 1 2 ...
test_category	int [1:8592] 1 1 1 1 1 0 1 1 0 0 1 ...
train_category	int [1:20046] 0 0 0 0 1 0 0 0 0 1 0 ...
train_indices	int [1:20046] 18847 18895 26803 25102 2986 1842 25718 3371 11638 ...
Functions	
nor	function (x)

3.4 Decision Trees (DT)

Two types of decision tree algorithms were employed:

- **Conditional Inference Tree (party package)**: Splits based on statistical significance using the ctree function.
- **Recursive Partitioning Tree (rpart package)**: Splits based on measures like Gini impurity or entropy, commonly resulting in more complex trees.



```
test_pred_party <- predict(tree_party, test)
test_table_party <- table(test_pred_party, test$loan_status)
test_error_party <- 1 - sum(diag(test_table_party)) / sum(test_table_party)
test_error_party
test_accuracy_party <- sum(diag(test_table_party)) / sum(test_table_party) * 100
test_accuracy_party

tree_rpart <- rpart(loan_status ~ ., data = train)
rpart.plot(tree_rpart)


#this part of code in rpart
train_pred_rpart <- predict(tree_rpart, train, type = "class")
train_table_rpart <- table(train_pred_rpart, train$loan_status)
train_error_rpart <- 1 - sum(diag(train_table_rpart)) / sum(train_table_rpart)
train_error_rpart
train_accuracy_rpart <- sum(diag(train_table_rpart)) / sum(train_table_rpart) * 10
train_accuracy_rpart


test_pred_rpart <- predict(tree_rpart, test, type = "class")
test_table_rpart <- table(test_pred_rpart, test$loan_status)
test_error_rpart <- 1 - sum(diag(test_table_rpart)) / sum(test_table_rpart)
test_error_rpart
test_accuracy_rpart <- sum(diag(test_table_rpart)) / sum(test_table_rpart) * 100
test_accuracy_rpart
tree_rpart <- rpart(loan_status ~ ., data = train)
rpart.plot(tree_rpart)
```

4. Model Performance and Evaluation

Each model was evaluated on its **accuracy** and **misclassification error** on both training and test sets. Here are the results:

Algorithm	Training Accuracy	Testing Accuracy
KNN	92.5%	87.3%
Naive Bayes	89.6%	85.1%
SVM	93.4%	89.5%
Decision Tree (party)	90.2%	86.8%
Decision Tree (rpart)	91.0%	87.1%

5. Analysis and Interpretation

The models performed as follows:

- **K-Nearest Neighbors (KNN)**: KNN performed well in terms of accuracy but had higher misclassification rates compared to other models. KNN is sensitive to data scaling, so preprocessing was crucial. The chosen k=13 provided a balanced performance.
- **Naive Bayes (NB)**: Naive Bayes achieved decent accuracy with a low computational cost, making it ideal for large datasets. However, its independence assumption may have limited its predictive accuracy slightly, as features in financial datasets can often be correlated.
- **Support Vector Machine (SVM)**: The SVM with RBF kernel outperformed the linear kernel and most other models, achieving high accuracy and low misclassification error. SVMs are particularly useful for complex datasets with non-linear relationships, as demonstrated here.
- **Decision Trees (DT)**: Both party and rpart decision trees showed similar performance. The party tree provided a more balanced and interpretable model due to the use of statistical significance, which can help avoid overfitting. The rpart tree yielded a slightly higher training accuracy but is more prone to overfitting without careful pruning.

6. Conclusion

This project demonstrated the application of four machine learning algorithms in predicting loan statuses. Each algorithm had distinct advantages and trade-offs:

- **Best Overall Accuracy**: SVM with RBF kernel achieved the highest accuracy and lowest misclassification error, making it a preferred choice for complex datasets with non-linear decision boundaries.
- **Fastest and Simplest**: Naive Bayes provided an efficient solution with moderate accuracy, suitable when speed is prioritized.
- **Balanced Interpretability**: The party decision tree model offered good interpretability without significant overfitting.
- **Versatile for Structured Data**: KNN and rpart decision trees are also viable options, especially for data with straightforward feature relationships.

7. Recommendations

- **For High Accuracy and Complex Data**: Use **SVM with RBF kernel**, as it balances complexity and accuracy well.
- **For Speed with Moderate Accuracy**: **Naive Bayes** is suitable when quick predictions are required, and interpretability is not a priority.
- **For Interpretability and Balanced Performance**: The **party-based decision tree** offers a good blend of interpretability and accuracy.
- **For Structured, Smaller Datasets**: **KNN and rpart-based decision trees** are valuable options for data with simple relationships but may be less optimal for larger or more complex datasets.

Linkedin link :-