Project 4: Classification Analysis
Group 4

100/100

TEAM MEMBERS

VIVEK KARICHETI (vk1296)
HARJOT SINGH DADIAL (hsd283)
SWATI RAVICHANDRAN (sr5227)

"We, Vivek Karicheti, Harjot Singh Dadial and Swati Ravichandran did not give or receive any assistance on this project from other groups, and the report submitted is wholly by us."

DATA UNDER CONSIDERATION

Steps to access the dataset:

1. Log onto Kaggle.com
2. Search for general datasets. (According to the prerequisites of the project, we were looking for datasets with at least 4 continuous variables and one binary variable, while keeping in mind there were more than or equal to 50 observations)
3. Download Loan Prediction dataset.
4. Check to see whether the dataset has any null values.
5. Using R Studio's na.omit(object name) function, remove any null values from the dataset.
6. From the total 614 observations, there were 85 which had Null values and we removed them from our consideration. Or data set used to implement this project had total of 529 observations.

Variables into consideration: The dataset gives us information on clients who want to apply for a loan, and various other information related to them. For this analysis, the dependent variable that we have chosen is the **Loan Status** variable and the four independent variables are **Applicant Income, Co-applicant Income, Loan Amount** and **Loan Amount Term.** Loan Status (binary class variable) is either **Y** (For yes) or **N** (For No), which tells us whether the loan was approved or disapproved. The variable Applicant Income is the client's income, Co-applicant Income is co-applicant's income, Loan Amount specifies the amount of loan the client is applying for and Loan Amount Term specifies the duration of the loan.

Scatter plot: Figure (1) shows the scatter plot for all variables under consideration for this project.

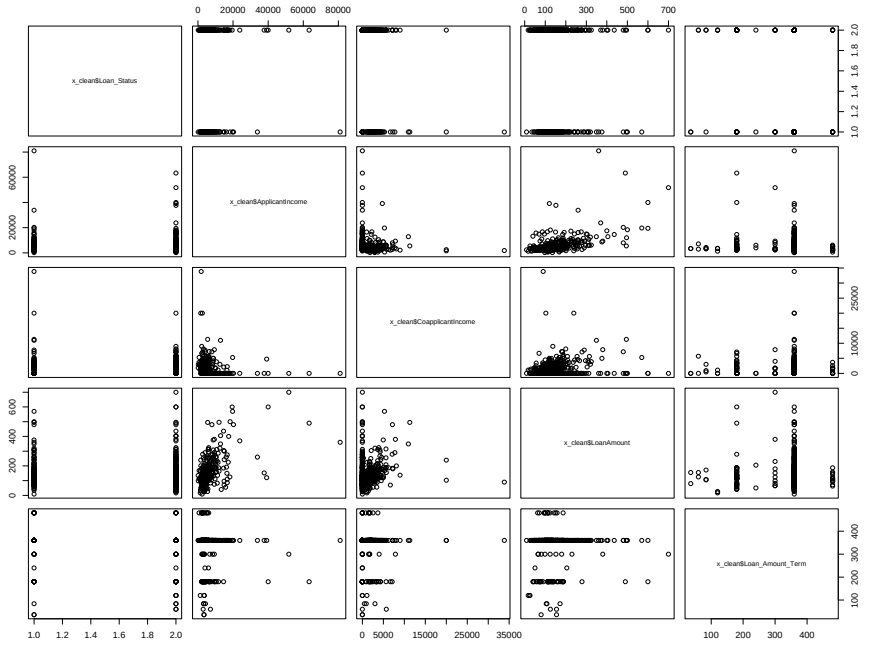

Figure (1): Scatter plot of all variables

The first row in Figure (1) shows the interaction of the class variable i.e., Loan status and all independent variables. Since there are only two classes, Y and N, we can see them clearly in the scatter plot as 2.0 and 1.0 respectively. In the first row, the y-axis is the class variable, Loan status and the x-axis is four independent variables giving us in total four plots. Further, in the Figure (1), interaction between the independent variables can be seen as well. Consider the plot in second row and fourth column; as per our observation, even though we do not see considerable correlation between most independent variables, there seems to be **a slight trend between the Applicant Income and the loan amount**. Thus, the scatter plot really gives a preliminary inference about both, the class variable and all independent variables.

Training and Test data set: The dataset is split into two parts in which, 80% of the original data is the training data set and the remaining 20% is the test data set. We used the R programming language to perform this step of our project. Figure (2) shows the snapshot of the R code used to divide the data into training and testing datasets: -

```
#training and testing bifurcation
n= nrow(work_data)
set.seed(567)
index_train_data <- sample(1:n, 0.8*n)
training_set <- work_data[index_train_data,]
testing_set <- work_data[-index_train_data,]
```

Figure (2): Code for bifurcating training and test data

In Figure (2), the variables are as follows:
work_data = Table of data after cleaning (removing observations with Null values) the original dataset and combining only variables under consideration for this project.
n = Number of observations in work_data
index_train_data = The index choosing 80% of numbers from n randomly
training_set = The training data for classification analysis (80% of work_data).
testing_set = The test data for classification analysis (20% of work_data).

BOXPLOTS

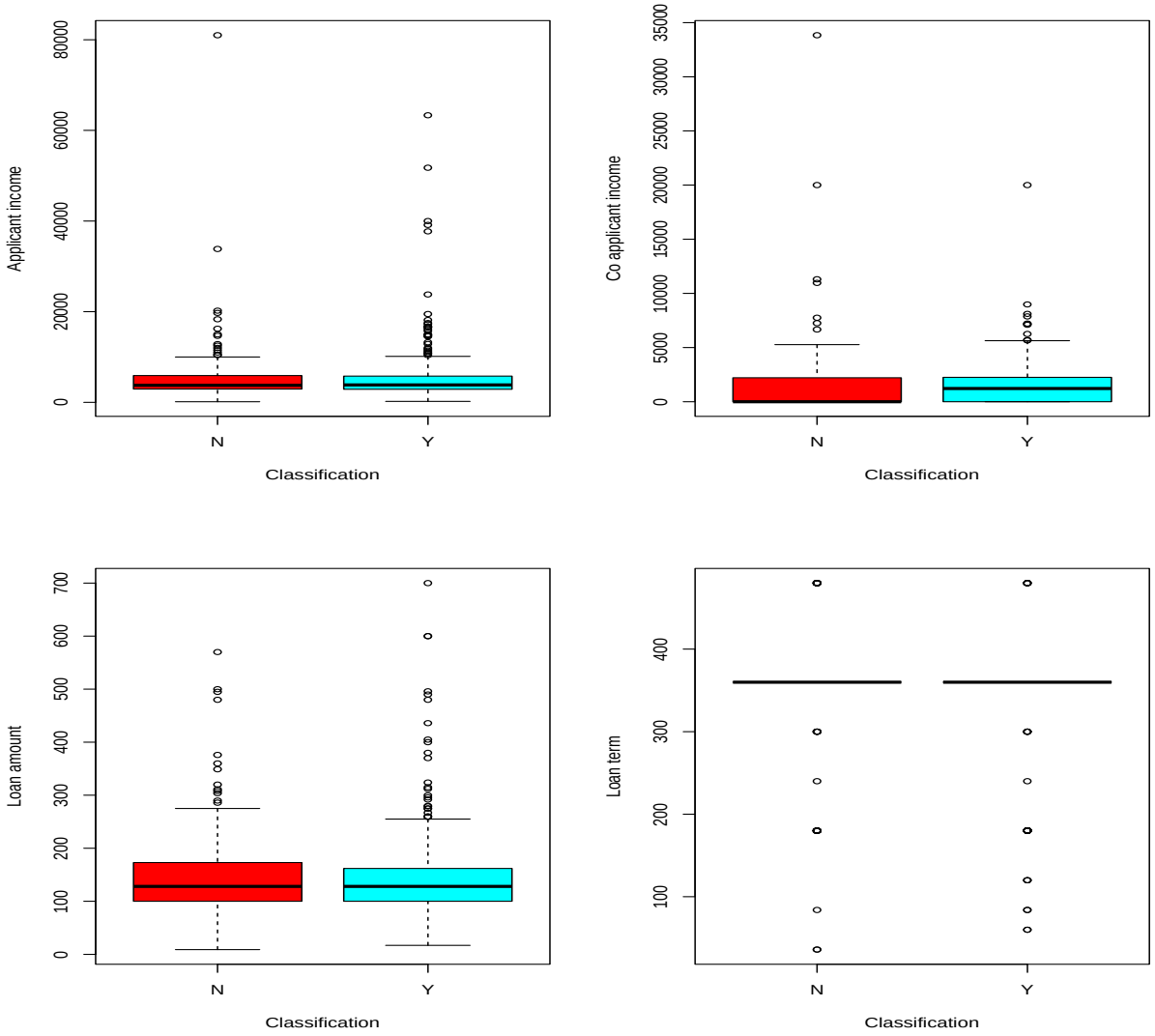

Figure (3): Boxplot of independent variables versus class variable

In Figure (3) we have obtained boxplots using R programming language to compare each independent variable with the class variable. Since we have four independent variables and two classes, we have in total eight boxplots in the above figure. Each box plot represents the distribution of independent variables into the two classes, Y and N. Let us consider each boxplot in Figure (3) one by one. In the top left, we have the box plot of **Applicant income versus the two classes**. By observation we can infer that the first quartile (Q1), the third quartile (Q3) and the median of Applicant's income is fairly same for both classes, the one that had their loan approved and the one that didn't get approval. The top right plot in Figure (3) shows the boxplot of **Co-applicant income versus the two classes**. We can infer that even though the Q1 and the Q3 for the both classes are the same, the median of co-applicant's income for the approved loans is way more than for the loans not approved. This shows that the co-applicant's income is of higher importance while considering loan requests. The bottom left plot in Figure (3) shows the box plot of **Loan amount versus the two classes.** We are able to observe that the Q3 for non-approved loan requests is more than the Q3 for approved loans. Finally, the bottom right part of Figure (3) shows the box plot of **Loan term versus the two classes.** We can observe that box plot for both classes is exactly the same. This suggests that the effect of loan term is not significant when building the classification models. This is also proved by observing the variable importance table (Table 1.) from the model summary in R and the decision tree in the report below.

CLASSIFICATION ANALYSIS AND CONFUSION MATRIX

K-Nearest Neighbors (K-NN) and Decision Tree Algorithm: The K-NN algorithm is a supervised machine learning algorithm that is used for classification and regression. It is one of the simplest classification algorithms and one of the most commonly used as well. It is applied on a dataset in which the data points are separated into several classes to predict the classification of a new sample point. K-NN Algorithm is used for classification in our project, the output of which is a class membership. An observation is classified by a majority vote of its neighbors, with the object being assigned to the class (Y or N) most common among its K nearest neighbors. Decision Trees are also a supervised learning algorithm used for classification and regression. It builds classification models in the form of a tree structure and breaks down data set into smaller subsets. While breaking it down, a tree-like structure with decision nodes, branches and leaf nodes is developed. A decision node has two or more branches. The leaf node represents a classification or decision (Y or N). The topmost node in a tree that corresponds to the best predictor is called the root node. Decision trees can handle both categorical and numerical data. In this project, we used K-Nearest Neighbor and the Decision Tree algorithms to build two classification models using the same training data set.

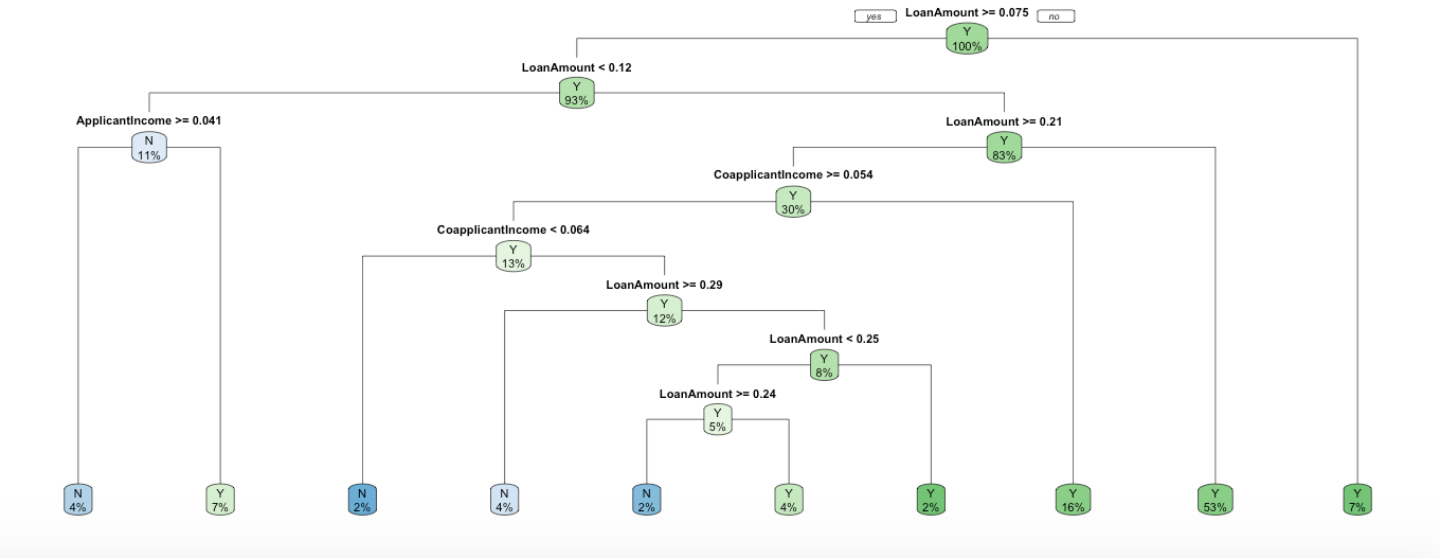

Figure (4): Decision Tree output

To generate and plot the decision tree shown in Figure (4), we used the **rpart.plot()** function. To understand the flow of the decision tree in Figure (4), let us take an example from the dataset. Consider the following information of a client from the dataset:

⇒ Applicant income = 2583
⇒ Co Applicant Income = 2358
⇒ Loan amount = 120
⇒ Loan Amount Term = 360

After normalizing, the values are as follows:

⇒ Applicant income = 0.030
⇒ Co-applicant income = 0.069
⇒ Loan amount = 0.160
⇒ Loan amount term = 0.729

At each node of the decision tree, the algorithm checks a specific condition. **If the data under consideration satisfies the condition, the flow goes to the left**. **If not, the flow goes to the right**. Also, at each node, the % signifies the percentage of clients from our dataset at that condition.
Below are the steps explaining the decision tree:

1. At the first node, the algorithm checks if the LoanAmount >= 0.075. As the client fulfills this condition, the algorithm goes to the left.
2. At this point, the algorithm checks if LoanAmount < 0.12. As this is not the case, algorithm goes to the right.
3. The algorithm checks if LoanAmount >= 0.21. This condition is false since the loan amount is 0.160, so algorithm goes to the right.
4. After going to the right, the algorithm comes to a decision that the Loan request for this client be Approved (Y).

Further, as seen in Figure (4), we cannot find the Loan Amount term variable in the decision tree. This is because the variable importance for Loan amount term for our model is zero. We can confirm this inference by referring to the variable importance table from summary of our model in R.

Variable Importance	Loan amount	Co-applicant income	Applicant income
	52	**29**	**19**

Table 1. Variable importance table

Confusion Matrix: We used the R programming language to generate the confusion matrix using our test data for each algorithm. The confusion matrix gives us values of 4 instances: True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Using these, we are able to calculate sensitivity of each model using the following formulae: -

⇒ **Sensitivity = TP / (TP+FN)**
⇒ **Specificity = TN / (TN+FP)**

We computed the Accuracy of models generated by both algorithms using R. Table 1. represents the confusion matrix for the K - Nearest Neighbor algorithm.

		Truth	
		Not approved	**Approved**
Predicted	**Not approved**	**3**	**1**
	Approved	**31**	**71**

Table 2. Confusion matrix for model generated using the K-NN algorithm

From Table 2. we can get, TP = 71, TN = 3, FP = 31and FN = 1
Therefore, **for the K-NN algorithm**, **Sensitivity = 0.9861** and **Specificity = 0.0882**
From output of R code, **Accuracy = 0.6981**

		Truth	
		Not approved	Approved
Predicted	Not approved	2	9
	Approved	32	63

Table 3. Confusion matrix for model generated using Decision tree algorithm

From Table 3. we get, TP = 63, TN = 2, FP = 32 and FN = 9
Therefore**, for the Decision tree algorithm**, **Sensitivity = 0.875** and **Specificity = 0.058**
From output of R code, **Accuracy = 0.6132**

COMPARISION OF THE TWO MODELS

After building the classification models for our dataset using two different algorithms, it is essential to compare various deductions and computations like sensitivity, specificity and accuracy which can help us decide based on merit of both the models, which one to go ahead with. **Sensitivity** is the ability to check the true positive rate i.e., to correctly identify those who qualify for a loan. Whereas, **Specificity** is the ability to check true negative rate i.e., correctly identify those who do not qualify for a loan. **Accuracy** can be defined as the ratio of correct classifications to the total number of input samples. In our case, the accuracy of the algorithm tells us how correctly will the developed model predict the outcome of a new client.

Both the algorithms have high sensitivity and low specificity. Our K-NN model has a sensitivity of 98.61%, which means using this model, we can correctly identify 98.81% of applicants who qualify for the loan. Similiarly, using our decision tree model, we can correctly identify 87.5% of applicants who qualify for the loan. Our K-NN model has a specificity of 8.82%, which means we can correctly recognize 8.82% of applicants who donot qualify for the loan. Similarly, our Decision tree model has a specificity rate of 5.8%, which means, we can correctly identify only 5.8% of applicants who do not qualify for the loan.The accuracy of our K-NN model is 69.81% whereas the accuracy of our decision tree model is 61.32%. Additionally, as observed in the scatter plot of the data set, the accuracy of our model could be increased by using the interaction of two independent variables (Applicant income and loan amount) that show a possible trend as seen in Figure(1).

Conclusion: Looking at these values, we can clearly say that our K-NN model is better suited for this dataset with a higher specificity, sensitivity and accuracy rate than our Decision Tree model. **Hence, we recommend forming a classification model using the K-Nearest Neighbour Algorithm for this dataset.**