CITS4009 Project

Rui QIN 24121014

Click this link to watch video (https://youtu.be/xqrcFVzIEUY)

Click this link to get dataset (https://www.kaggle.com/datasets/lainguyn123/student-performance-factors/data)

Introduction

In today's education system, there are many factors that influence students' academic performance. These factors range from the amount of study time to the level of parental involvement, and each plays a role in students' academic outcomes. The availability of such data provides a unique opportunity for data-driven insights and predictive modeling.

In this study, we will explore a dataset focused on student performance, examining various factors that may impact academic success. By identifying patterns and relationships within this data, our goal is to clarify how different variables (such as study habits and family background) affect students' exam scores.

Loading libraries

```
library(dplyr)
library(ggplot2)
library(ggcorrplot)
library(caret)
library(ROCR)
library(e1071)
library(pROC)
library(rpart)
library(lime)
library(cluster)
library(purrr)
```

Loading dataset

```
# replace empty space with NA
dataset <- read.csv("StudentPerformanceFactors.csv", na.strings = c("", "NA"))
```

1.Exploratory Study of the Data

```
str(dataset)
```

```
## 'data.frame':    6607 obs. of  20 variables:
##  $ Hours_Studied             : int  23 19 24 29 19 19 29 25 17 23 ...
##  $ Attendance                : int  84 64 98 89 92 88 84 78 94 98 ...
##  $ Parental_Involvement      : chr  "Low" "Low" "Medium" "Low" ...
##  $ Access_to_Resources       : chr  "High" "Medium" "Medium" "Medium" ...
##  $ Extracurricular_Activities: chr  "No" "No" "Yes" "Yes" ...
##  $ Sleep_Hours               : int  7 8 7 8 6 8 7 6 6 8 ...
##  $ Previous_Scores           : int  73 59 91 98 65 89 68 50 80 71 ...
##  $ Motivation_Level          : chr  "Low" "Low" "Medium" "Medium" ...
##  $ Internet_Access           : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ Tutoring_Sessions         : int  0 2 2 1 3 3 1 1 0 0 ...
##  $ Family_Income             : chr  "Low" "Medium" "Medium" "Medium" ...
##  $ Teacher_Quality           : chr  "Medium" "Medium" "Medium" "Medium" ...
##  $ School_Type               : chr  "Public" "Public" "Public" "Public" ...
##  $ Peer_Influence            : chr  "Positive" "Negative" "Neutral" "Negative" ...
##  $ Physical_Activity         : int  3 4 4 4 4 3 2 2 1 5 ...
##  $ Learning_Disabilities     : chr  "No" "No" "No" "No" ...
##  $ Parental_Education_Level  : chr  "High School" "College" "Postgraduate" "High School"
...
##  $ Distance_from_Home        : chr  "Near" "Moderate" "Near" "Moderate" ...
##  $ Gender                    : chr  "Male" "Female" "Male" "Male" ...
##  $ Exam_Score                : int  67 61 74 71 70 71 67 66 69 72 ...
```

The response variable is student Exam Score. It is a value that can change depending on other factors

- Categorical data: Parental_Involvement、Access_to_Resources、Extracurricular_Activities、Motivation_Level、Internet_Access、Family_Income、Teacher_Quality、School_Type、Peer_Influence、Learning_Disabilities、Parental_Education_Level、Distance_from_Home、Gender.
- Numerical data: Hours_Studied、Attendance、Sleep_Hours、Previous_Scores、Tutoring_Sessions、Physical_Activity、Exam_Score.
- The dataset contains 6607 rows, each row representing a students, and contains 20 variables.

1.1 Explore the data

```
summary(dataset$Exam_Score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   55.00   65.00   67.00   67.24   69.00  101.00
```

From this we can see:

- The lowest test score in the data set was 55.
- 25% of students scored below 65 on the exam, and 75% scored below 69 on the exam.
- The average score is 67.24, the median is 67, and they are very close, meaning that the data is normally distributed.
- The highest test score in the data set is 101, but it’s likely an outlier.

```
sum(dataset$Exam_Score == 101)
```

```
## [1] 1
```

Only one person scored 101, which means there are outliers in the data, and we need to identify them later and remove them.

1.2 Data cleaning and transformation

1.2.1 Recognise NAs

```
missing_data <- colSums(is.na(dataset))
print(missing_data)
```

```
##              Hours_Studied                 Attendance
##                          0                          0
##       Parental_Involvement        Access_to_Resources
##                          0                          0
## Extracurricular_Activities                Sleep_Hours
##                          0                          0
##            Previous_Scores           Motivation_Level
##                          0                          0
##            Internet_Access          Tutoring_Sessions
##                          0                          0
##              Family_Income            Teacher_Quality
##                          0                         78
##                School_Type             Peer_Influence
##                          0                          0
##          Physical_Activity      Learning_Disabilities
##                          0                          0
##   Parental_Education_Level         Distance_from_Home
##                         90                         67
##                     Gender                 Exam_Score
##                          0                          0
```

- Most columns in the data set have no missing values.
- The Teacher_Quality column has 78 missing values.
- The Parental_Education_Level column contains 90 missing values.
- The Distance_from_Home column has 67 missing values.

We need to explore which NAs can be directly filled in the mode, which need to determine its value through the distribution of data and fill in, and which need to be directly deleted

```
# Teacher_Quality distribution
table(dataset$Teacher_Quality, useNA = "ifany")
```

```
##
##   High    Low Medium   <NA>
##   1947    657   3925     78
```

```
# Distance_from_Home distribution
table(dataset$Distance_from_Home, useNA = "ifany")
```

```
##
##      Far Moderate     Near     <NA>
##      658     1998     3884       67
```

```
# Parental_Education_Level distribution
table(dataset$Parental_Education_Level, useNA = "ifany")
```

```
##
##      College  High School Postgraduate         <NA>
##         1989         3223         1305           90
```

- Since “Medium” is overwhelming, the missing values here can reasonably be filled in with “Medium”
- Since “Near” is very large, mode filling is appropriate in this case
- Although “High School” accounts for the largest proportion, the direct filling mode may not be accurate enough given that parental education level is an important indicator. So consider padding based on other variables, such as Family_Income.

```
# View the distribution of Family_Income and Parental_Education_Level
table(dataset$Family_Income, dataset$Parental_Education_Level, useNA = "ifany")
```

```
##
##          College High School Postgraduate <NA>
##   High       370         633          252   14
##   Low        805        1305          523   39
##   Medium     814        1285          530   37
```

Based on the results for Family_Income and Parental_Education_Level, we can populate the missing values based on the distribution. For example, for Low Family_Income, we could randomly fill in missing values by the ratio of High School, College, and Postgraduate.

1.2.2 Replace NAs

```
# For Parental_Education_Level
dataset <- dataset %>%
  group_by(Family_Income) %>%
  mutate(Parental_Education_Level = ifelse(is.na(Parental_Education_Level),
                                           sample(Parental_Education_Level[!is.na(Parental_Ed
ucation_Level)],
                                                  sum(is.na(Parental_Education_Level)), repla
ce = TRUE),
                                           Parental_Education_Level))

# For Teacher_Quality
dataset$Teacher_Quality[is.na(dataset$Teacher_Quality)] <- mode(dataset$Teacher_Quality)

#For Distance_from_Home
dataset$Distance_from_Home[is.na(dataset$Distance_from_Home)] <- mode(dataset$Distance_from_H
ome)

# Check for missing values
sum(is.na(dataset$Parental_Education_Level))
```

```
## [1] 0
```

```
sum(is.na(dataset$Teacher_Quality))
```

```
## [1] 0
```

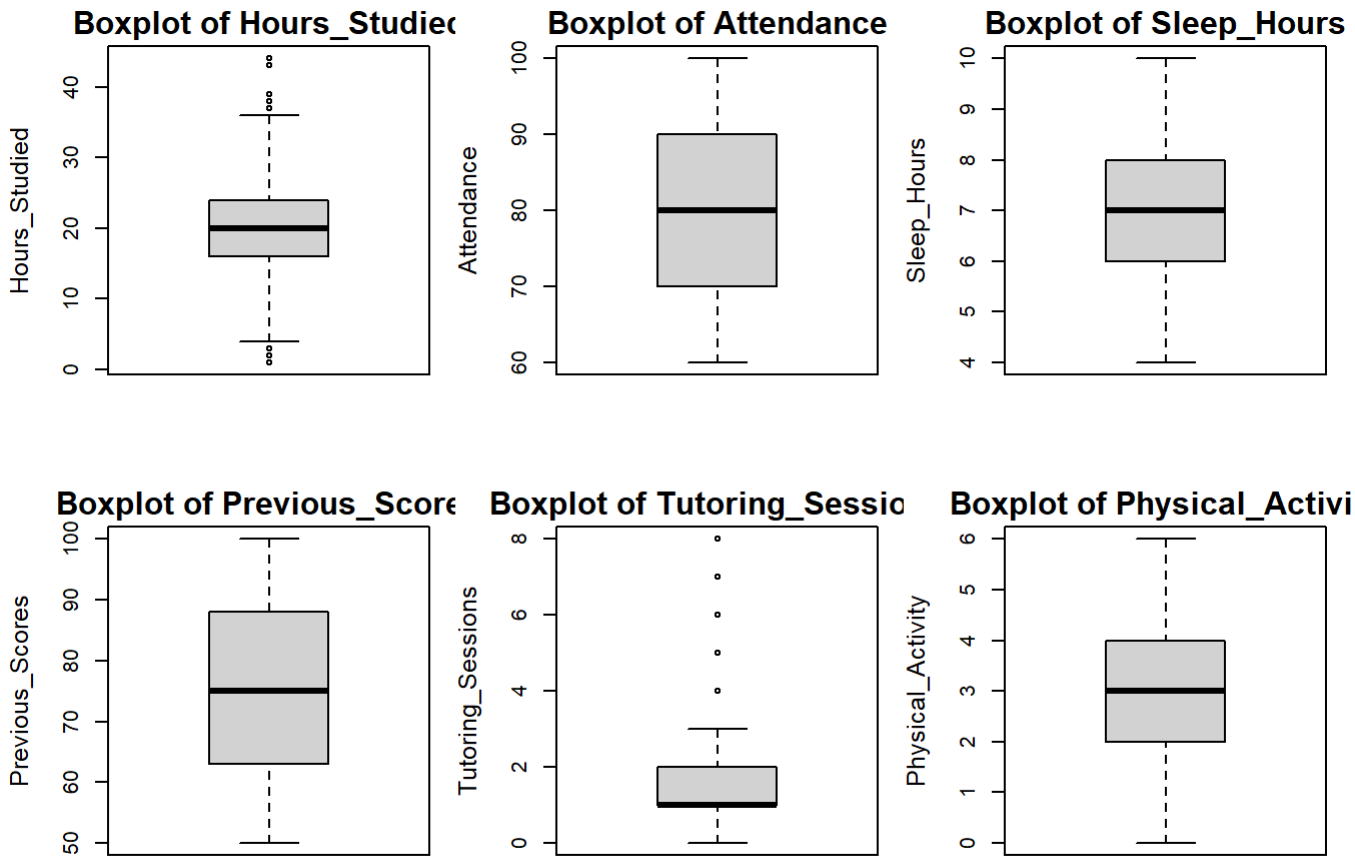
```
sum(is.na(dataset$Distance_from_Home))
```

```
## [1] 0
```

1.2.3 Remove outlier

After we remove the NAs, we can use the IQR to determine the outliers in the variable and remove them

```
# Visualize outliers for numeric columns
numeric_columns <- c("Hours_Studied", "Attendance", "Sleep_Hours",
                     "Previous_Scores", "Tutoring_Sessions", "Physical_Activity")

par(mfrow = c(2, 3),  # The layout is 2 rows and 4 columns
    mar = c(4, 4, 2, 1),  # Adjust margins
    cex.main = 1.5,  # Increase font size
    cex.lab = 1.2)

# Create boxplot
for (col in numeric_columns) {
  boxplot(dataset[[col]], main = paste("Boxplot of", col), ylab = col)
}
```

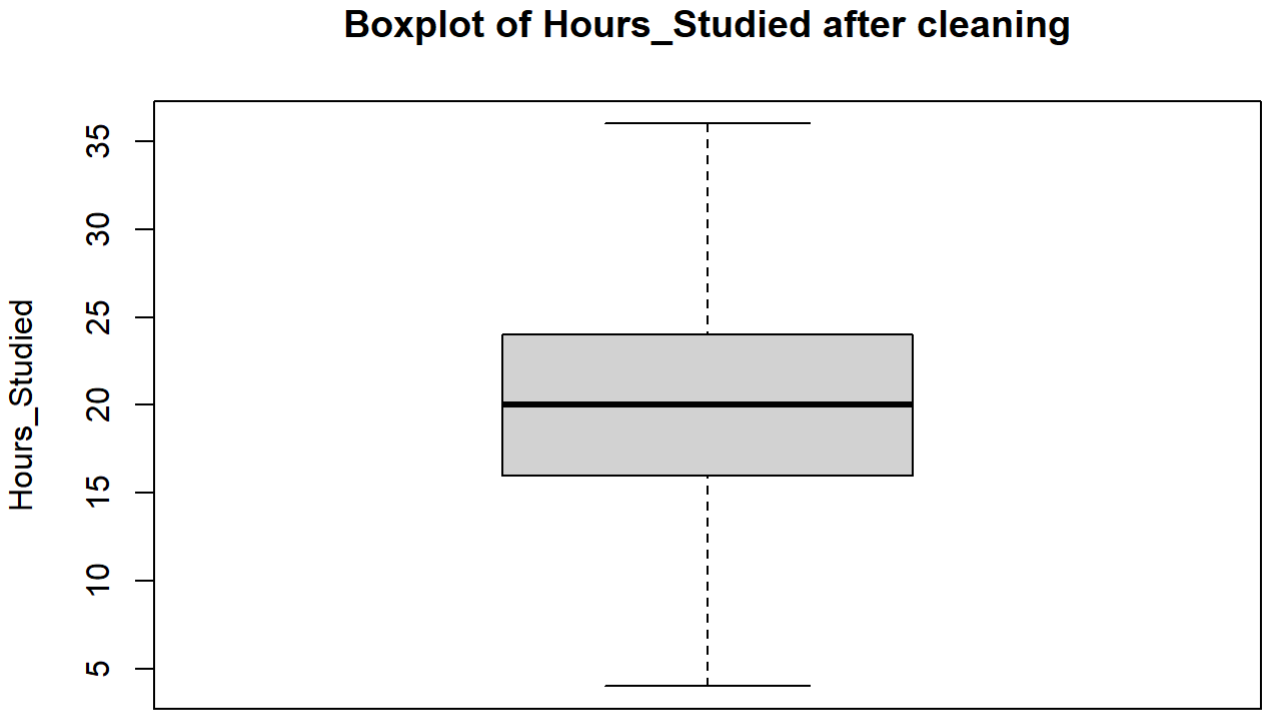


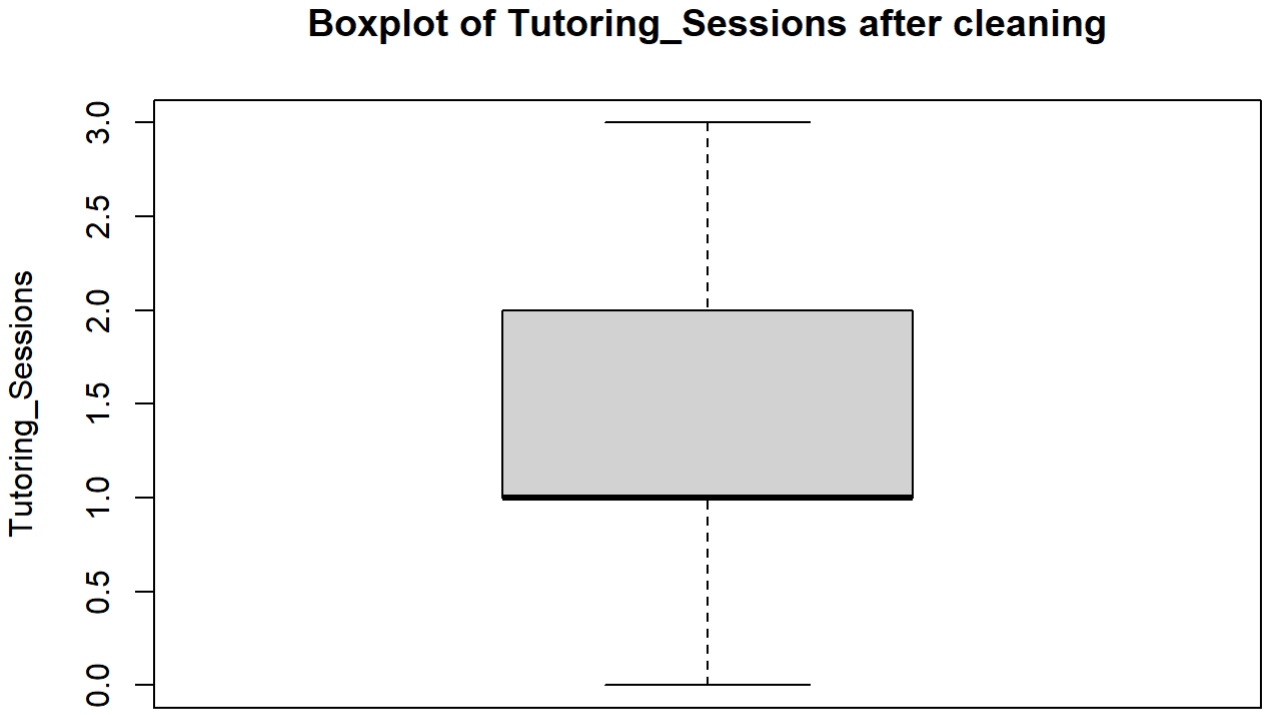
According to the figure, only Hours_Studied and Tutoring_Sessions in the data have outliers

```
numeric_columns <- c("Hours_Studied", "Tutoring_Sessions")

remove_outliers <- function(data, cols) {
  for (col in cols) {

    Q1 <- quantile(data[[col]], 0.25, na.rm = TRUE)
    Q3 <- quantile(data[[col]], 0.75, na.rm = TRUE)
    IQR <- Q3 - Q1


    lower_bound <- Q1 - 1.5 * IQR
    upper_bound <- Q3 + 1.5 * IQR

    # Filter outliers that exceed the lower and upper limits
    data <- data %>%
      filter(!!sym(col) >= lower_bound & !!sym(col) <= upper_bound)
  }
  return(data)
}

dataset_cleaned <- remove_outliers(dataset, numeric_columns)

# Check whether the size of the cleaned data set has been reduced
nrow(dataset_cleaned)
```

```
## [1] 6134
```

```
for (col in numeric_columns) {
  boxplot(dataset_cleaned[[col]], main = paste("Boxplot of", col, "after cleaning"), ylab = c
ol)
}
```



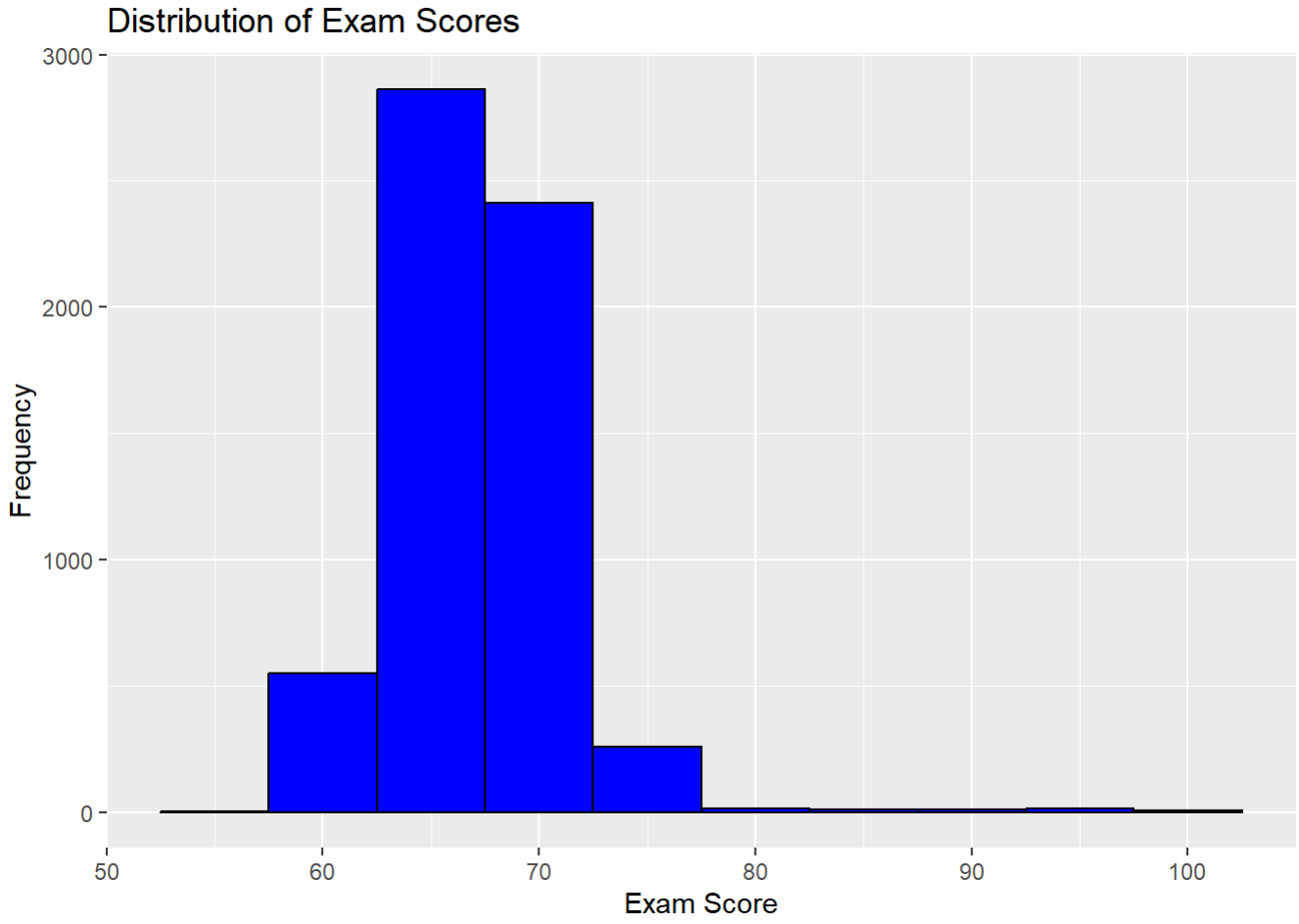


According to the figures, both the Hours_Studied and Tutoring_Sessions outliers have been removed

1.3 Visualisation

1.3.1 Distribution of Exam Scores

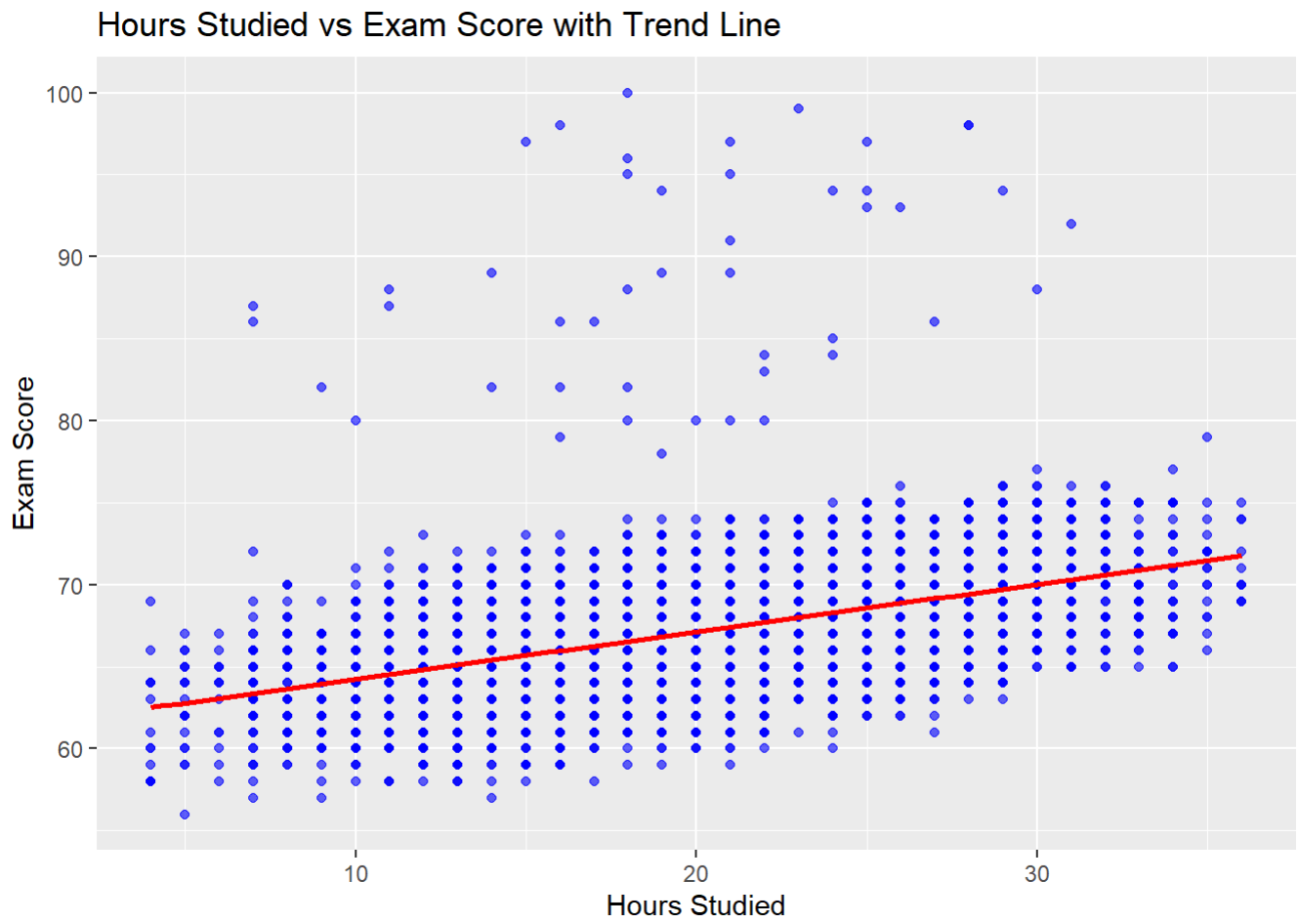
```
ggplot(dataset_cleaned, aes(x = Exam_Score)) +
  geom_histogram(binwidth = 5, fill = "blue", color = "black") +
  labs(title = "Distribution of Exam Scores", x = "Exam Score", y = "Frequency")
```



The histogram shows that most students' scores are concentrated between 65 and 69, indicating that the scores are relatively centered in this range. However, a small number of students can achieve scores between 80 and 100, although this is quite rare.

1.3.2 Hours Studied vs Exam Score with Trend Line

```
ggplot(dataset_cleaned, aes(x = Hours_Studied, y = Exam_Score)) +
  geom_point(color = "blue", alpha = 0.6) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = "Hours Studied vs Exam Score with Trend Line", x = "Hours Studied", y = "Exam
Score")
```



There is a slight positive correlation between the number of study hours and exam scores, as shown by the upward-sloping red trend line. This indicates that most students who study less than 10 hours score around 60-70, while those who study more tend to score slightly higher. However, some students who study for longer periods still score low, suggesting that more study time does not always guarantee better performance, and it is likely that high scores are related to natural ability.

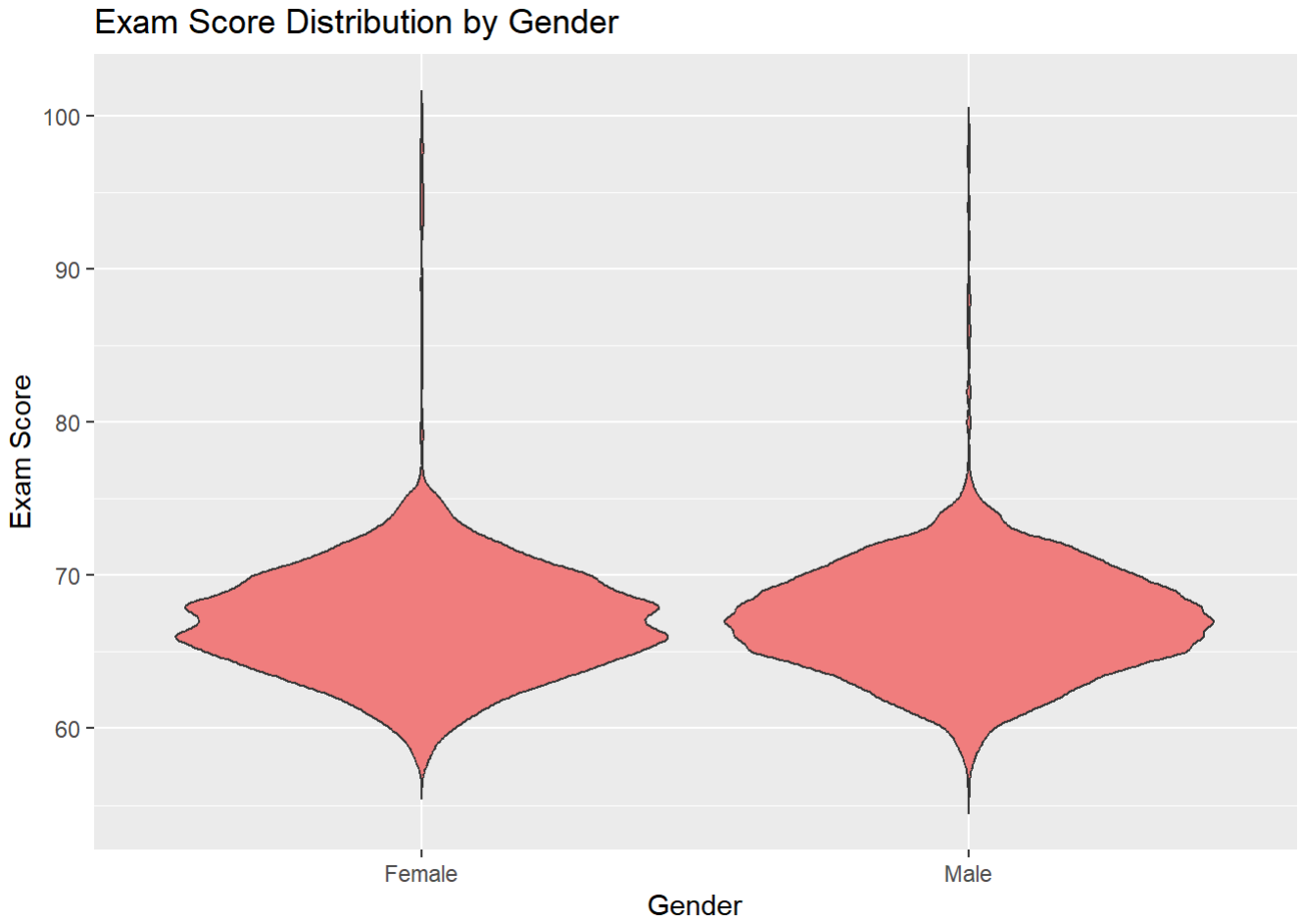
1.3.3 Family Income vs Exam Score

```
ggplot(dataset_cleaned, aes(x = Family_Income, y = Exam_Score)) +
  geom_boxplot(fill = "lightcoral") +
  labs(title = "Family Income vs Exam Score", x = "Family Income", y = "Exam Score")
```

The median exam scores of students from different household income groups (high, middle, and low) are very similar, all around 70. However, the chart does show that students from low-income families tend to perform slightly worse, as their highest scores, including outliers, are around 96-97, with fewer scores in the 95-100 range (only one), and their IQR distribution is lower.

1.3.4 Exam Score Distribution by Gender

```
ggplot(dataset_cleaned, aes(x = Gender, y = Exam_Score)) +
  geom_violin(trim = FALSE, fill = "lightcoral") +
  labs(title = "Exam Score Distribution by Gender", x = "Gender", y = "Exam Score")
```



The distribution of males and females is almost identical, with most exam scores concentrated around 70. Both genders have some high outliers above 90, but the overall distribution is similar. Gender does not appear to be a significant factor in determining exam scores.

1.3.5 Correlation Matrix

```
correlation_matrix <- cor(dataset_cleaned[, sapply(dataset_cleaned, is.numeric)])
ggcorrplot(correlation_matrix, lab = TRUE)
```

Exam_Score shows a moderate positive correlation with Attendance (0.59) and Hours_Studied (0.44), indicating that higher attendance and more study hours are associated with higher scores. It is also relatively correlated with Previous_Scores and Tutoring_Sessions.

2.Modelling

2.1 Preparing

2.1.1 Target Variable

We chose Exam_Score as the target variable. To translate this into a binary classification problem, we can choose a score threshold to divide grades, classifying exam_scores above a certain score (say 67) as “pass” and below that as “failed.”

```
dataset_classification <- dataset_cleaned

# Convert Exam_Score to a binary categorical variable (threshold 67), Pass = 1 Fail =- 0
dataset_classification$Pass_Fail <- ifelse(dataset_classification$Exam_Score >= 67, 1, 0)

# Delete the Exam_Score column because we have generated binary categorical variables
dataset_classification <- dataset_classification %>% select(-Exam_Score)

# Check the distribution of categories
table(dataset_classification$Pass_Fail)
```

```
##
##    0    1
## 2742 3392
```

We can find that when the threshold is 67, the distribution of fail and pass is relatively average

2.1.2 Modify

- Previous_Scores: If we want to predict whether students' scores pass or not, but we directly use the historical score Previous_Scores as the feature, it may lead to overfitting of the model, so we remove it.
- Gender: Can be removed because base on previous observation, we know gender has little effect on predicted performance.

```
dataset_classification <- dataset_classification %>%
  select(-Previous_Scores)

dataset_classification <- dataset_classification %>%
  select(-Gender)
```

one-hot encodes all character variables

```
dataset_classification_dummy <- dataset_classification

target <- dataset_classification$Pass_Fail

# one-hot encoding of character variables, excluding target variables
dataset_classification_dummy <- dataset_classification_dummy %>%
  select(-Pass_Fail) %>%
  mutate_at(vars(Parental_Involvement, Access_to_Resources, Extracurricular_Activities,
                 Motivation_Level, Internet_Access, Family_Income, Teacher_Quality,
                 School_Type, Peer_Influence, Learning_Disabilities,
                 Parental_Education_Level, Distance_from_Home),
            funs(as.factor(.)))

# one-hot code the remaining features using dummyVars
dummies <- dummyVars(~., data = dataset_classification)
dataset_classification_dummy$Pass_Fail <- target

dataset_classification_dummy <- data.frame(predict(dummies, newdata = dataset_classification_
dummy))
```

Next, delete all variables with too high a correlation and Pass_Fail, where the threshold is set to 0.8

```
# Calculate the correlation matrix of the numerical variables
Vars <- setdiff(colnames(dataset_classification_dummy), "Pass_Fail")
correlation_matrix <- cor(dataset_classification_dummy$Pass_Fail, dataset_classification_dumm
y[, Vars], use = 'pairwise.complete.obs')
sorted_corr <- correlation_matrix[, order(abs(correlation_matrix), decreasing = TRUE)]

# Remove highly correlated features (assuming a correlation coefficient threshold of 0.8)
high_corr_vars <- names(sorted_corr[abs(sorted_corr) > 0.8])
dataset_classification_dummy <- dataset_classification_dummy %>%
  select(-all_of(high_corr_vars))
```

Splitting the data into training and testing dataset

```
set.seed(24121014)
trainIndex <- createDataPartition(dataset_classification_dummy$Pass_Fail, p = 0.8,
                                  list = FALSE,
                                  times = 1)

train_data <- dataset_classification_dummy[trainIndex,]

test_data <- dataset_classification_dummy[-trainIndex,]

dim(train_data)
```

```
## [1] 4908   40
```

```
dim(test_data)
```

```
## [1] 1226   40
```

2.2 Classification

2.2.1 Single Variable Classification

```
# Get all dummy variable column names (excluding Pass_Fail column)
dummyVars <- setdiff(colnames(dataset_classification_dummy), "Pass_Fail")

# Define the AUC evaluation function
calcAUC <- function(predcol, outcol) {
  perf <- performance(prediction(as.numeric(predcol), outcol == 1), "auc")
  as.numeric(perf@y.values)
}

# Make predictions for each single variable and calculate the AUC
for (v in dummyVars) {

    train_pred <- train_data[[v]]
  test_pred <- test_data[[v]]

# Calculate the AUC of the training set and the test set
  aucTrain <- calcAUC(train_pred, train_data$Pass_Fail)
  if (aucTrain >= 0.53) {
    aucTest <- calcAUC(test_pred, test_data$Pass_Fail)
    print(sprintf("%s: trainAUC: %4.3f; testAUC: %4.3f", v, aucTrain, aucTest))
  }
}
```

```
## [1] "Hours_Studied: trainAUC: 0.720; testAUC: 0.731"
## [1] "Attendance: trainAUC: 0.847; testAUC: 0.818"
## [1] "Parental_InvolvementHigh: trainAUC: 0.553; testAUC: 0.552"
## [1] "Access_to_ResourcesHigh: trainAUC: 0.565; testAUC: 0.537"
## [1] "Tutoring_Sessions: trainAUC: 0.561; testAUC: 0.564"
## [1] "Peer_InfluencePositive: trainAUC: 0.536; testAUC: 0.546"
## [1] "Parental_Education_LevelPostgraduate: trainAUC: 0.533; testAUC: 0.527"
## [1] "Distance_from_HomeNear: trainAUC: 0.531; testAUC: 0.543"
```

We set up lowest AUC threshold: 0.53, and create selected dataset.

```
selected_trainvars <- train_data[, c('Hours_Studied','Attendance','Parental_InvolvementHig
h','Access_to_ResourcesHigh','Tutoring_Sessions', 'Peer_InfluencePositive', 'Parental_Educati
on_LevelPostgraduate', 'Distance_from_HomeNear','Pass_Fail')]

selected_testvars <- test_data[, c('Hours_Studied','Attendance','Parental_InvolvementHigh','A
ccess_to_ResourcesHigh','Tutoring_Sessions', 'Peer_InfluencePositive', 'Parental_Education_Le
velPostgraduate', 'Distance_from_HomeNear','Pass_Fail')]
```

By comparing the performance of all feature models with those built using subsets, we assess the impact of attribute and feature selection on the model.

2.2.2 Train the classification

Model with all features

```
train_data$Pass_Fail <- as.factor(train_data$Pass_Fail)
test_data$Pass_Fail <- as.factor(test_data$Pass_Fail)

tree_model <- rpart(Pass_Fail ~ ., data=train_data, method="class")

# Prediction for the test data
tree_pred_class <- predict(tree_model, newdata=test_data, type="class")

# Evaluate the performance (confusion matrix)
conf_matrix <- confusionMatrix(tree_pred_class, test_data$Pass_Fail)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 432 110
##          1 110 574
##
##                Accuracy : 0.8206
##                  95% CI : (0.7979, 0.8417)
##     No Information Rate : 0.5579
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6362
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7970
##             Specificity : 0.8392
##          Pos Pred Value : 0.7970
##          Neg Pred Value : 0.8392
##              Prevalence : 0.4421
##          Detection Rate : 0.3524
##    Detection Prevalence : 0.4421
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : 0
##
```

```
nb_model <- naiveBayes(Pass_Fail ~ ., data=train_data)

# Prediction for the test data
nb_pred_class <- predict(nb_model, newdata=test_data)

# Evaluate the performance (confusion matrix)
conf_matrix_nb <- confusionMatrix(nb_pred_class, test_data$Pass_Fail)
print(conf_matrix_nb)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 385  85
##          1 157 599
##
##                Accuracy : 0.8026
##                  95% CI : (0.7792, 0.8246)
##     No Information Rate : 0.5579
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5943
##
##  Mcnemar's Test P-Value : 5.018e-06
##
##             Sensitivity : 0.7103
##             Specificity : 0.8757
##          Pos Pred Value : 0.8191
##          Neg Pred Value : 0.7923
##              Prevalence : 0.4421
##          Detection Rate : 0.3140
##    Detection Prevalence : 0.3834
##       Balanced Accuracy : 0.7930
##
##        'Positive' Class : 0
##
```

For Accuracy:

The decision tree model has a slightly higher accuracy than the Naive Bayes model (82.06% > 80.51%). Both models show high accuracy, indicating that they generally perform well in distinguishing between the “Pass” and “Fail” categories.

Sensitivity and Specificity:

The decision tree performs better at identifying failing students (Sensitivity 79.70% > 71.22%), while the Naive Bayes model excels at identifying passing students (Specificity 83.92% > 87.87%). This means the decision tree tends to predict failing students more accurately, whereas the Naive Bayes model is more precise in predicting passing students.

Kappa:

The Kappa value of the decision tree model is slightly higher than that of the Naive Bayes model (0.6362 > 0.5992), indicating that its classification results are more reliable compared to random guessing.

McNemar’s Test P-Value:

McNemar’s Test is used to detect whether there is a significant misclassification bias between different categories. The Naive Bayes model shows significant classification bias, indicating that it makes more errors in certain situations.

Balanced Accuracy:

The decision tree model has slightly higher balanced accuracy, suggesting that its classification ability is more balanced between different categories.

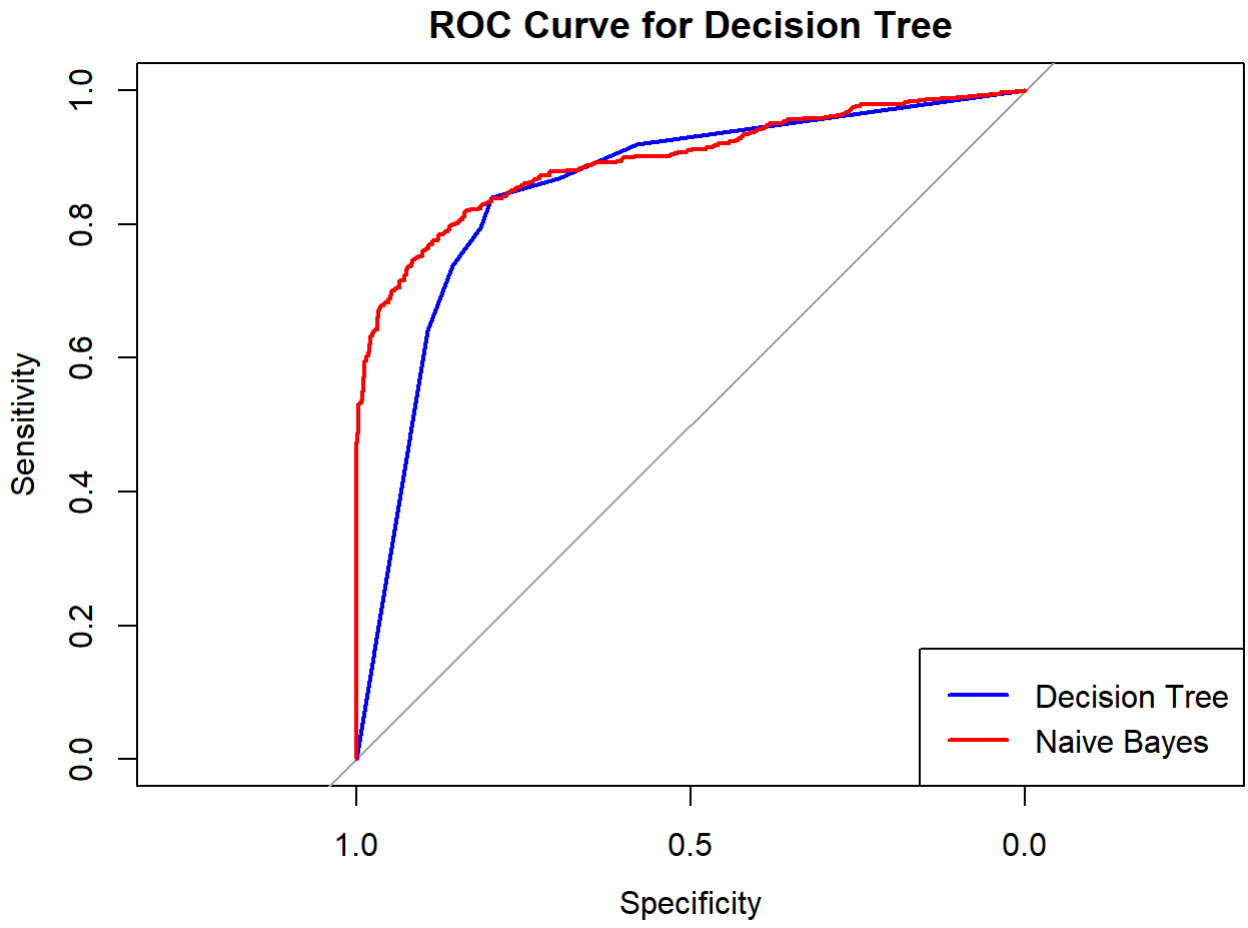
```
# Decision Tree Prediction probabilities (use type="prob" for predicted probabilities)
tree_pred_prob <- predict(tree_model, newdata=test_data, type="prob")

# Naive Bayes Prediction probabilities
nb_pred_prob <- predict(nb_model, newdata=test_data, type="raw")

# Plot ROC curve for Decision Tree
roc_tree <- roc(test_data$Pass_Fail, tree_pred_prob[,2])


# Plot ROC curve for Naive Bayes
roc_nb <- roc(test_data$Pass_Fail, nb_pred_prob[,2])

auc(roc_nb)
```

```
## Area under the curve: 0.8947
```

```
auc(roc_tree)
```

```
## Area under the curve: 0.8526
```

```
plot(roc_tree, col = "blue", lwd = 2, main = "ROC Curve for Decision Tree")
plot(roc_nb, col = "red", lwd = 2, add=TRUE)
legend("bottomright", legend=c("Decision Tree", "Naive Bayes"), col=c("blue", "red"), lwd=2)
```



The ROC performance of the Decision Tree model appears to outperform that of the Naive Bayes model with higher accuracy (0.8945 vs 0.8526).

For this binary classification problem of student performance, the decision tree model is relatively more suitable for the dataset, especially with a more balanced ability to classify between categories. The Naive Bayes model performs better at correctly identifying passing students, but it is slightly worse at classifying failing students and has issues with classification bias.

Train the classification model with selected features

```
selected_trainvars$Pass_Fail <- as.factor(selected_trainvars$Pass_Fail)
selected_testvars$Pass_Fail <- as.factor(selected_testvars$Pass_Fail)

tree_model <- rpart(Pass_Fail ~ ., data=selected_trainvars, method="class")

# Prediction for the test data
tree_pred_class <- predict(tree_model, newdata=selected_testvars, type="class")

# Evaluate the performance (confusion matrix)
conf_matrix <- confusionMatrix(tree_pred_class, selected_testvars$Pass_Fail)
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 432 110
##          1 110 574
##
##                Accuracy : 0.8206
##                  95% CI : (0.7979, 0.8417)
##     No Information Rate : 0.5579
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6362
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7970
##             Specificity : 0.8392
##          Pos Pred Value : 0.7970
##          Neg Pred Value : 0.8392
##              Prevalence : 0.4421
##          Detection Rate : 0.3524
##    Detection Prevalence : 0.4421
##       Balanced Accuracy : 0.8181
##
##        'Positive' Class : 0
##
```

```r
nb_model <- naiveBayes(Pass_Fail ~ ., data=selected_trainvars)

# Prediction for the test data
nb_pred_class <- predict(nb_model, newdata=selected_testvars)

# Evaluate the performance (confusion matrix)
conf_matrix_nb <- confusionMatrix(nb_pred_class, selected_testvars$Pass_Fail)
print(conf_matrix_nb)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 447  90
##          1  95 594
##
##                Accuracy : 0.8491
##                  95% CI : (0.8278, 0.8687)
##     No Information Rate : 0.5579
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6938
##
##  Mcnemar's Test P-Value : 0.7687
##
##             Sensitivity : 0.8247
##             Specificity : 0.8684
##          Pos Pred Value : 0.8324
##          Neg Pred Value : 0.8621
##              Prevalence : 0.4421
##          Detection Rate : 0.3646
##    Detection Prevalence : 0.4380
##       Balanced Accuracy : 0.8466
##
##        'Positive' Class : 0
##
```

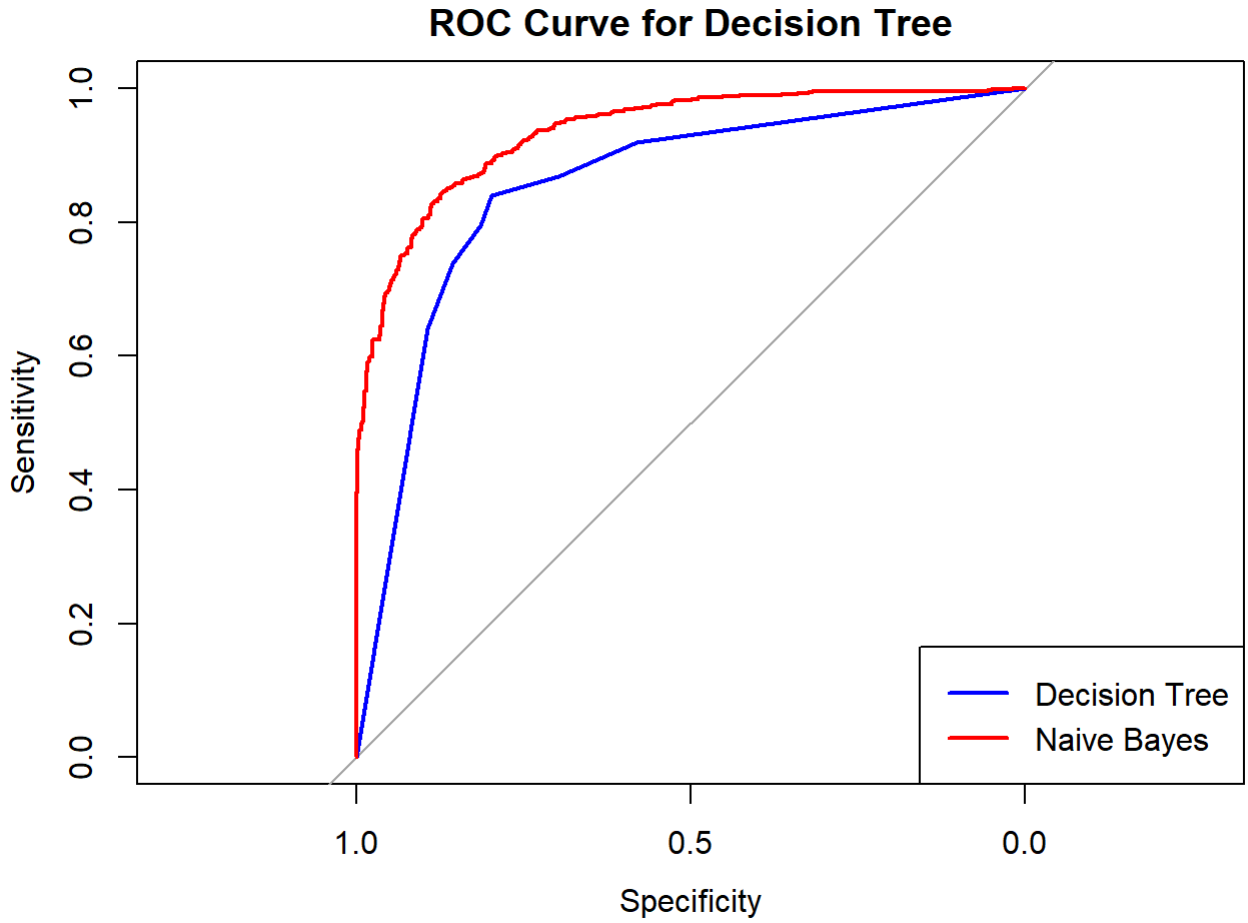
```r
# Decision Tree Prediction probabilities (use type="prob" for predicted probabilities)
tree_pred_prob <- predict(tree_model, newdata=selected_testvars, type="prob")

# Naive Bayes Prediction probabilities
nb_pred_prob <- predict(nb_model, newdata=selected_testvars, type="raw")

# Plot ROC curve for Decision Tree
roc_tree <- roc(selected_testvars$Pass_Fail, tree_pred_prob[,2])


# Plot ROC curve for Naive Bayes
roc_nb <- roc(selected_testvars$Pass_Fail, nb_pred_prob[,2])

auc(roc_nb)
```

```
## Area under the curve: 0.9347
```

```
auc(roc_tree)
```

```
## Area under the curve: 0.8526
```

```
plot(roc_tree, col = "blue", lwd = 2, main = "ROC Curve for Decision Tree")
plot(roc_nb, col = "red", lwd = 2, add=TRUE)
legend("bottomright", legend=c("Decision Tree", "Naive Bayes"), col=c("blue", "red"), lwd=2)
```



When we used the selected variable, we found that all the data showed that Naive Bayes had a higher classification ability than the Decision Tree

Conclusion:

In the analysis of the binary classification problem, both the decision tree model and the naive Bayes model show high accuracy, and the ability to identify the “pass” and “fail” categories is generally good. Decision tree models were better at identifying students who failed, while naive Bayes models were better at identifying students who passed.

When considering all features, the decision tree model is slightly more accurate than the naive Bayes model, and also slightly better in balance accuracy, indicating that its classification ability is more balanced across different classes. However, when we used the selected variables for training, the performance of the naive Bayes model improved significantly, its accuracy and Kappa value were better than that of the decision tree model, and the area under the ROC curve also increased from 0.8946 to 0.9343, indicating that its classification ability was enhanced.

Therefore, the selection of appropriate features and appropriate models is crucial to improve prediction performance, especially in the face of data sets with complex feature Spaces

LIME

LIME needs to know whether the model is a classification or regression model. To ensure that LIME can correctly handle the Decision Tree and Naive Bayes models, we need to explicitly inform it that these models are classification models. Additionally, since the default prediction output of the Naive Bayes model may not match LIME's requirements, we need to customize the prediction function and wrap the model to ensure that LIME can use this customized model for interpretation.

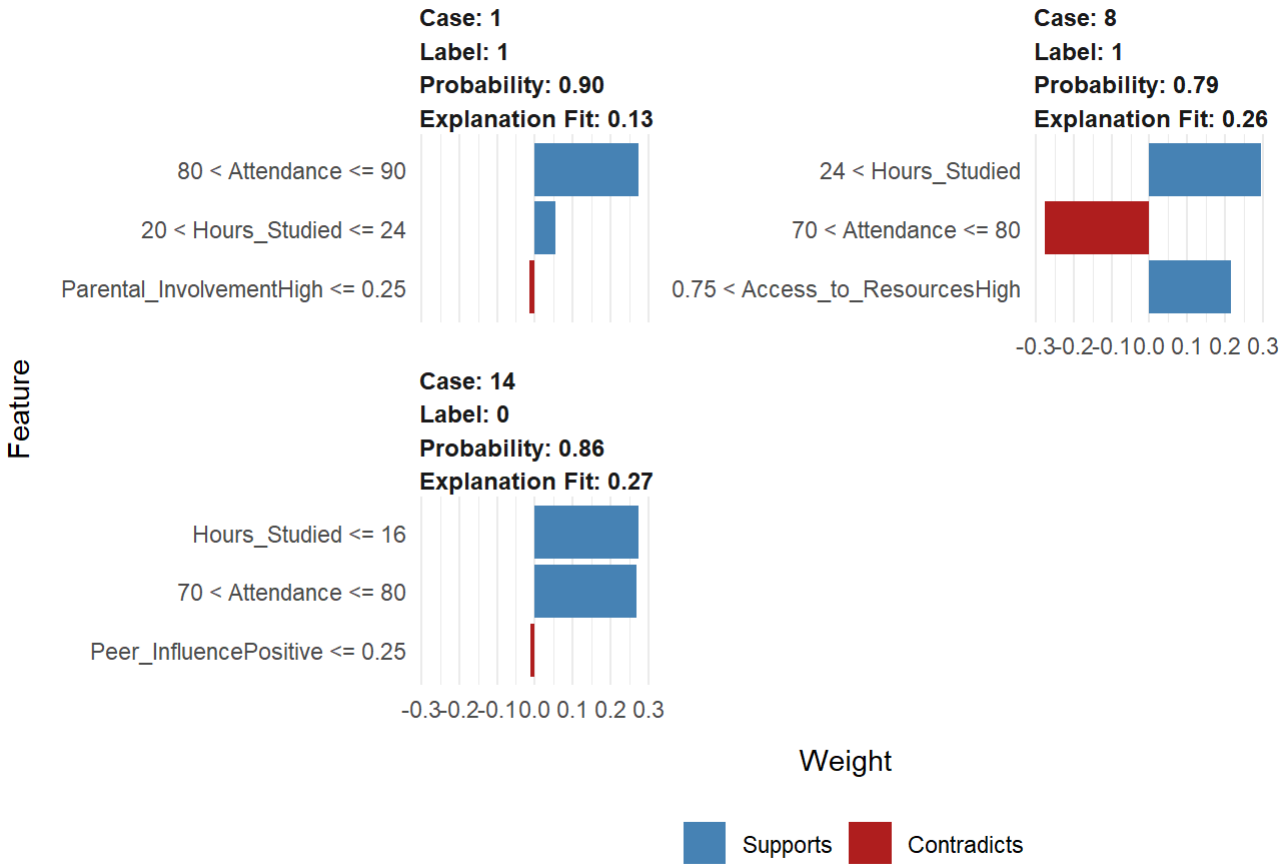
```
# Define a custom function for the Decision Tree model
assign("model_type.lime_model_rpart", function(object, ...) {
  return("classification")
}, envir = .GlobalEnv)

lime_model_rpart <- function(x) {
  class(x) <- c("lime_model_rpart", class(x))
  x
}
# Define a custom predict function for Naive Bayes model
predict_proba <- function(model, newdata) {
  predict(model, newdata, type = "raw")
}

# Define model type and prediction functions for Naive Bayes
assign("model_type.nb", function(x, ...) {
  return(x$type)
}, envir = .GlobalEnv)

assign("predict_model.nb", function(x, newdata, ...) {
  predict_result <- predict(x$model, newdata = newdata, type = "raw")
  return(data.frame(predict_result))
}, envir = .GlobalEnv)

nb_model_proba <- list(model = nb_model, predict = predict_proba, type = "classification")
class(nb_model_proba) <- 'nb'


# Prepare the explainer
explainer_tree <- lime(selected_trainvars, lime_model_rpart(tree_model))
explainer_nb <- lime(selected_trainvars, nb_model_proba)

# Generate explanations
explanation_tree <- explain(selected_testvars[1:3, ], explainer_tree, n_labels = 1, n_feature
s = 3)
explanation_nb <- explain(selected_testvars[1:3, ], explainer_nb, n_labels = 1, n_features =
3)


print(explanation_tree)
```

```
## # A tibble: 9 × 13
##   model_type    case  label label_prob model_r2 model_intercept model_prediction
##   <chr>         <chr> <chr>      <dbl>    <dbl>           <dbl>            <dbl>
## 1 classificati… 1     1          0.904    0.131           0.471            0.788
## 2 classificati… 1     1          0.904    0.131           0.471            0.788
## 3 classificati… 1     1          0.904    0.131           0.471            0.788
## 4 classificati… 8     1          0.785    0.264           0.551            0.785
## 5 classificati… 8     1          0.785    0.264           0.551            0.785
## 6 classificati… 8     1          0.785    0.264           0.551            0.785
## 7 classificati… 14    0          0.856    0.267           0.319            0.853
## 8 classificati… 14    0          0.856    0.267           0.319            0.853
## 9 classificati… 14    0          0.856    0.267           0.319            0.853
## # ℹ 6 more variables: feature <chr>, feature_value <dbl>, feature_weight <dbl>,
## #   feature_desc <chr>, data <list>, prediction <list>
```

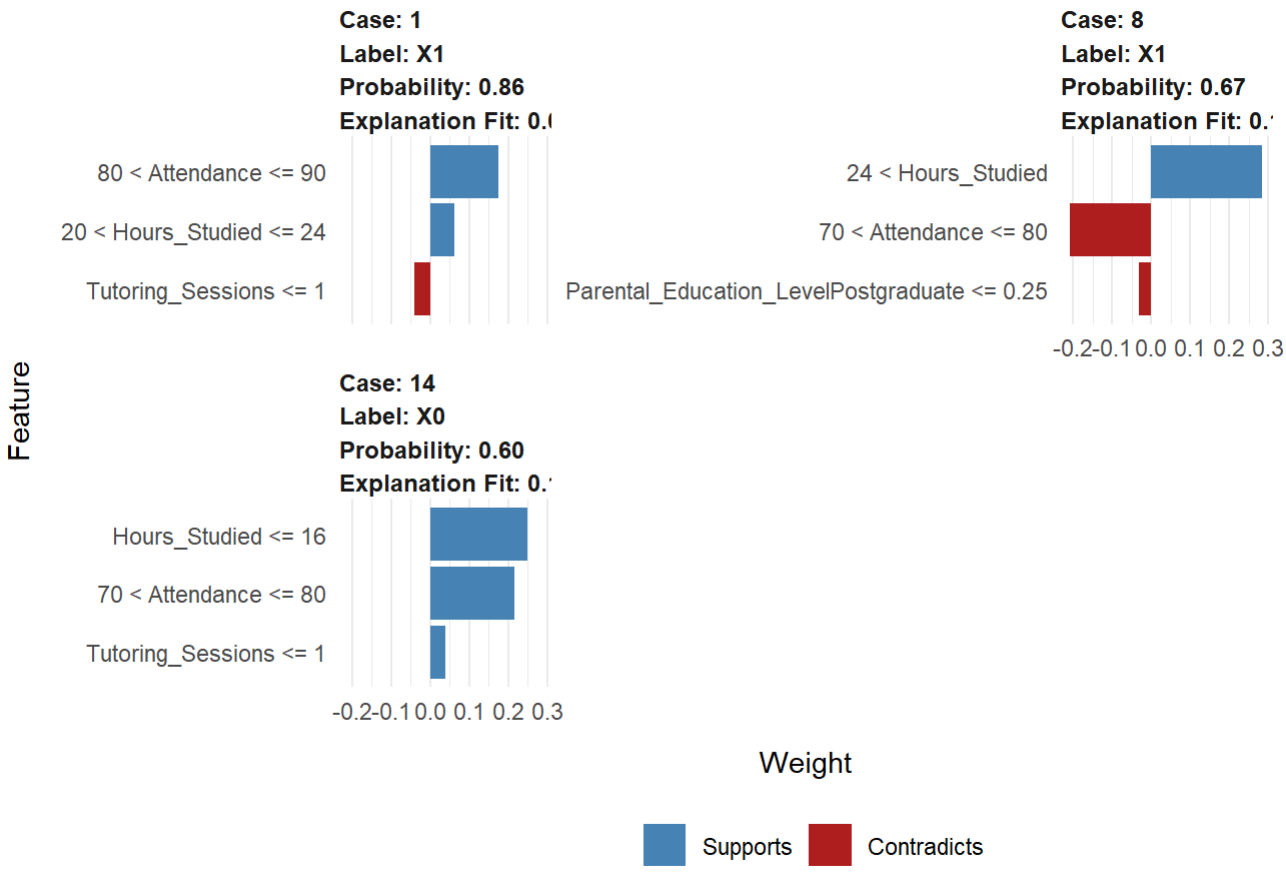
```
print(explanation_nb)
```

```
## # A tibble: 9 × 13
##   model_type    case  label label_prob model_r2 model_intercept model_prediction
##   <chr>         <chr> <chr>      <dbl>    <dbl>           <dbl>            <dbl>
## 1 classificati… 1     X1         0.859   0.0592           0.421            0.619
## 2 classificati… 1     X1         0.859   0.0592           0.421            0.619
## 3 classificati… 1     X1         0.859   0.0592           0.421            0.619
## 4 classificati… 8     X1         0.668   0.197            0.463            0.507
## 5 classificati… 8     X1         0.668   0.197            0.463            0.507
## 6 classificati… 8     X1         0.668   0.197            0.463            0.507
## 7 classificati… 14    X0         0.597   0.196            0.404            0.908
## 8 classificati… 14    X0         0.597   0.196            0.404            0.908
## 9 classificati… 14    X0         0.597   0.196            0.404            0.908
## # ℹ 6 more variables: feature <chr>, feature_value <dbl>, feature_weight <dbl>,
## #   feature_desc <chr>, data <list>, prediction <list>
```

```
plot_features(explanation_tree)
```




```
plot_features(explanation_nb)
```



LIME breaks down the contribution of each feature to specific predictions. For each instance, it shows which features support the prediction (blue) and which features contradict it (red).

Decision Tree:

- Study time and attendance are the most important features across multiple cases, consistently showing strong contributions to predicting both “Pass” and “Fail” outcomes.
- Features like parental involvement and access to resources also play a role, but their impact varies significantly across different instances.
- The Decision Tree model tends to capture more direct relationships between individual features and the target label, which is reflected in the weights provided by LIME explanations.

Naive Bayes Model:

- The Naive Bayes model also focuses on study time and attendance, but the feature tutoring sessions contradicts the prediction.
- Naive Bayes handles feature effects probabilistic, leading to more nuanced contributions, where even minor features like tutoring sessions can slightly influence the probability of passing or failing.
- Compared to the Decision Tree model, LIME explanations for the Naive Bayes model generally show smaller weights, reflecting the smoother handling of feature impacts by the Naive Bayes model.

2.3 Clustering

In this project, clustering helps us to identify patterns in various learning behaviors and performance of students. By analyzing multidimensional data such as students’ learning time, attendance, and parental engagement, we can divide students into different groups for further study of their learning patterns

```
selected_features <- dataset_cleaned%>%
  select(Hours_Studied, Attendance, Tutoring_Sessions, Exam_Score)

# Standardized numerical features
numeric_features <- selected_features[, c("Hours_Studied", "Attendance", "Tutoring_Session
s","Exam_Score")]

scaled_data <- scale(numeric_features)
```

```
pca_model <- prcomp(scaled_data, center = TRUE, scale. = TRUE)

summary(pca_model)
```

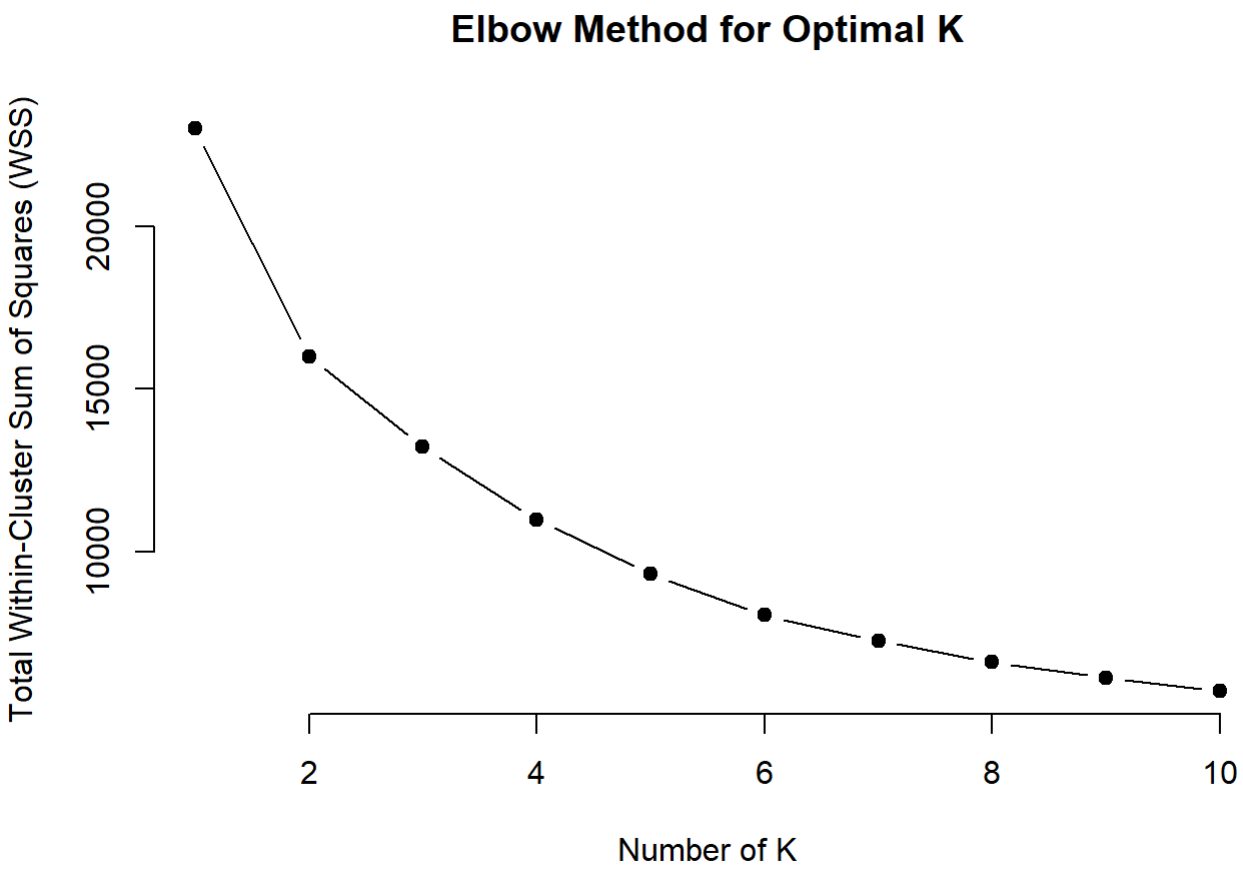
```
## Importance of components:
##                           PC1    PC2    PC3     PC4
## Standard deviation     1.3184 1.0089 0.9964 0.50140
## Proportion of Variance 0.4345 0.2545 0.2482 0.06285
## Cumulative Proportion  0.4345 0.6890 0.9372 1.00000
```

```
# The first few principal components are selected, so that the cumulative variance interpreta
tion rate reaches 90%
pca_data <- data.frame(pca_model$x[, 1:which(cumsum(pca_model$sdev^2 / sum(pca_model$sdev^2))
>= 0.9)[1]])
```

Principal component 1 and principal component 2 together explained nearly 69% of the variation in the data. While PC3 explains about 24.82% of the variation, its contribution is slightly weaker compared to the first two major components. This principal component may reflect the influence of secondary features, such as fewer important variables or some random pattern of learning behavior.

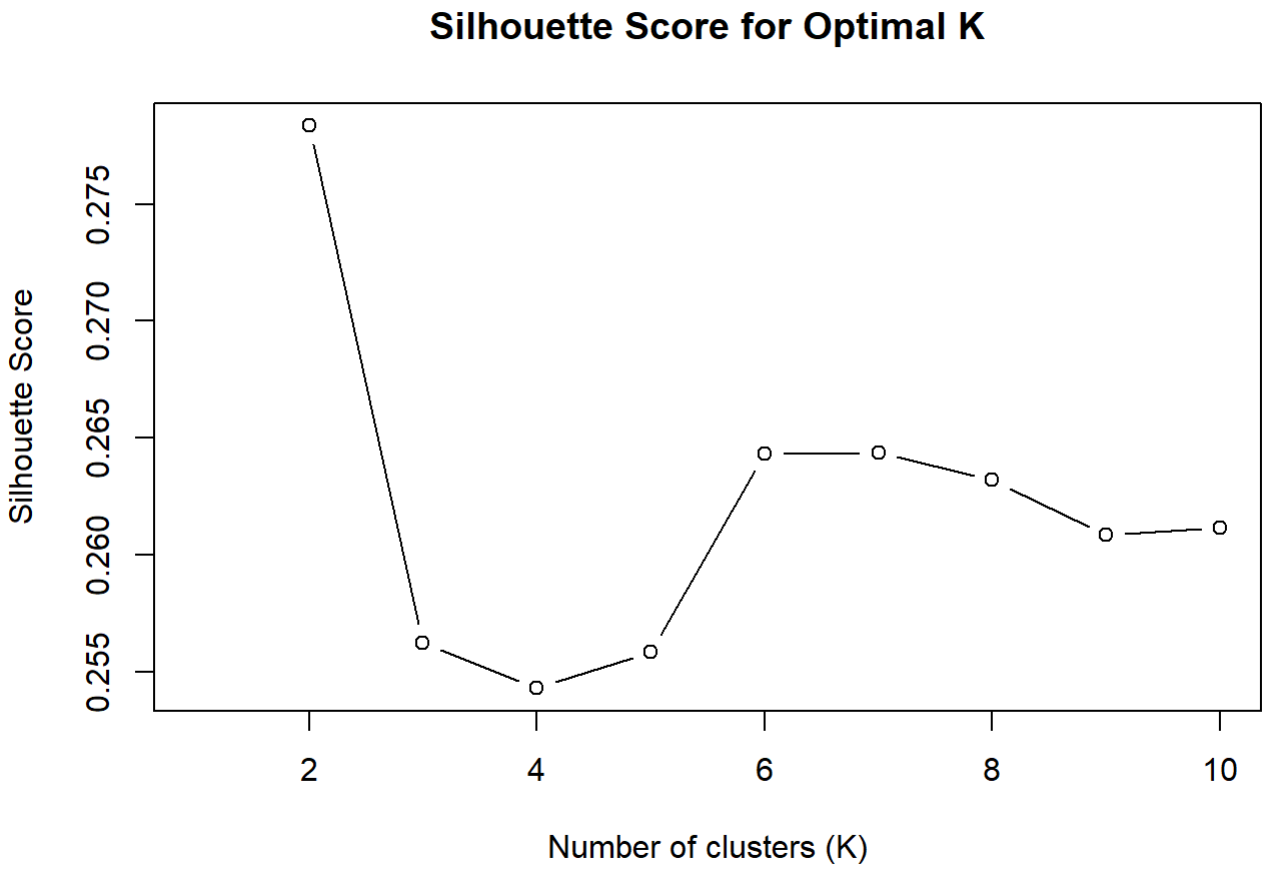
Now we are going to use Elbow method and silhouette score to determine the K value

```
wss <- function(data, k) {
  kmeans(data, k, nstart = 10)$tot.withinss
}

# Try different K values and calculate WSS
k_values <- 1:10
wss_values <- map_dbl(k_values, wss, data = pca_data)

plot(k_values, wss_values, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of K",
     ylab = "Total Within-Cluster Sum of Squares (WSS)",
     main = "Elbow Method for Optimal K")
```



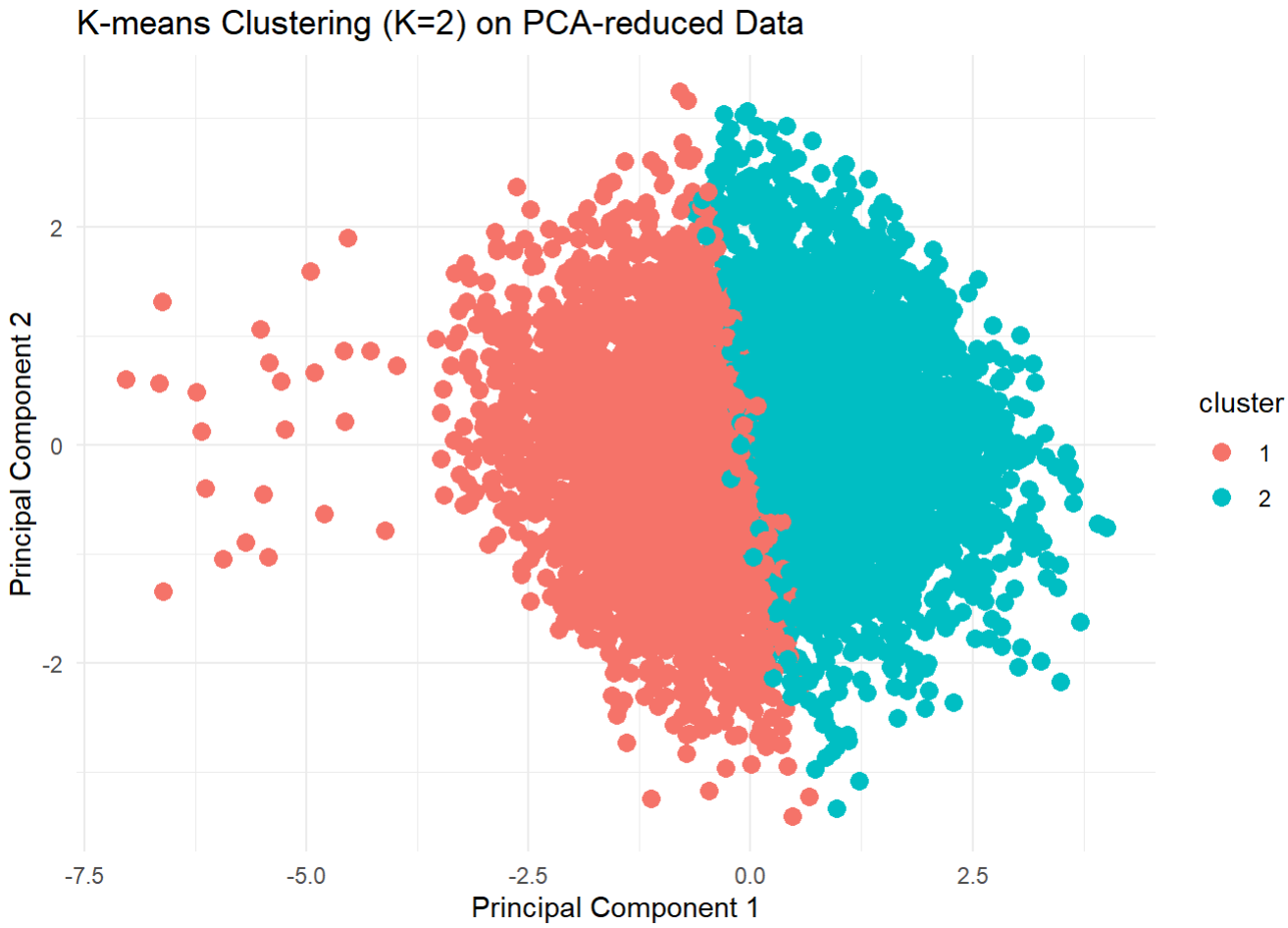
In this graph, it can be seen that WSS decreases the most when K=2 to K=3, and then decreases gradually with the increase of K value. Therefore, the inflection point in the graph occurs around K=3, which means that K in the range of 2 to 3 May be a more ideal number of clusters.

```
silhouette_scores <- numeric(length(k_values))

# Do k-means clustering for each K value and calculate the Silhouette Score
for (i in 1:length(k_values)) {
  k <- k_values[i]

  kmeans_model <- kmeans(pca_data, centers = k, nstart = 25)

  dist_matrix <- dist(pca_data)

  silhouette_result <- silhouette(kmeans_model$cluster, dist_matrix)

  if (length(dim(silhouette_result)) == 2) {
    silhouette_scores[i] <- mean(silhouette_result[, 3])
  } else {
    silhouette_scores[i] <- NA
  }
}

# Draw the Silhouette Score for each K value

plot(k_values, silhouette_scores, type = "b",
     xlab = "Number of clusters (K)", ylab = "Silhouette Score",
     main = "Silhouette Score for Optimal K")
```



In this diagram, you can see that the Silhouette Score is at its maximum from K=2. According to the Silhouette Score and Elbow Method, K = 2 is probably the best value for K.

```r
k = 2
kmeans_result <- kmeans(pca_data, centers = k, nstart = 10)

# Visualize the clustering results
pca_data$cluster <- as.factor(kmeans_result$cluster)

ggplot(pca_data, aes(x = PC1, y = PC2, color = cluster)) +
  geom_point(size = 3) +
  labs(title = paste("K-means Clustering (K=", k, ") on PCA-reduced Data", sep=""),
       x = "Principal Component 1", y = "Principal Component 2") +
  theme_minimal()
```




```r
# Combine raw data with clustering results
clustered_data <- cbind(dataset_cleaned , cluster = kmeans_result$cluster)

# The average value of each feature is calculated according to the clustering results
cluster_summary <- clustered_data %>%
  group_by(cluster) %>%
  summarise(across(c(Hours_Studied, Attendance, Tutoring_Sessions), mean, na.rm = TRUE))

# Print the average of each cluster
print(cluster_summary)
```

```
## # A tibble: 2 × 4
##   cluster Hours_Studied Attendance Tutoring_Sessions
##     <int>         <dbl>      <dbl>              <dbl>
## 1       1          21.6       89.0               1.39
## 2       2          18.5       71.4               1.20
```

Through K-means clustering analysis, students can be divided into two distinct groups with the following main characteristics:

- Study Time: Students in Cluster 1 have a higher average study time compared to those in Cluster 2, indicating that this group tends to dedicate more time to studying. This suggests that spending more study time may effectively improve academic performance.
- Attendance Rate: The attendance rate of students in Cluster 1 is significantly higher than that of Cluster 2. This suggests that higher attendance is often associated with better study habits and improved academic performance.
- Participation in Tutoring Sessions: Although the difference in tutoring session participation between the two groups is not substantial, students in Cluster 1 participate slightly more. This indicates that this group may rely more on additional tutoring to enhance their academic performance.

3.Conclusion

In conclusion, factors such as study time, attendance and tutoring sessions play an important role in determining student achievement. Both decision trees and naive Bayes models provide valuable insights, with naive Bayes slightly superior on some metrics. Cluster analysis further reveals different groups of students divided according to learning behavior, which can be used to tailor teaching programs to different groups of students.