

Overview of the Wine Quality Dataset:

The Wine Quality dataset contains various physicochemical properties and sensory quality ratings of white wines. It is often used in machine learning and data analysis to predict the quality of wines based on their chemical composition. The dataset consists of 11 physicochemical attributes and one target variable representing the wine quality rating.

Key Features:

Number of Instances: The dataset contains 4,898 instances of white wine samples.

Number of Attributes: There are 11 input attributes (predictors) representing different chemical properties of the wines.

Target Variable: The dataset includes one target variable, "quality," which represents the sensory quality rating of wines on a scale from 0 to 10.

Attribute Information: The attributes include:

Fixed Acidity: The nonvolatile acids in the wine.

Volatile Acidity: The amount of acetic acid in the wine, which contributes to vinegar taste.

Citric Acid: The presence of citric acid, which adds freshness to the wine.

Residual Sugar: The amount of sugar remaining after fermentation.

Chlorides: The amount of salt in the wine.

Free Sulfur Dioxide: The amount of free SO₂, which prevents microbial growth.

Total Sulfur Dioxide: The total SO₂ present, including the bound and free forms.

Density: The density of the wine, relative to water.

pH: The acidity level of the wine.

Sulphates: The amount of sulfur dioxide in the wine.

Alcohol: The alcohol content in the wine.

Code:

```
# Load required libraries
install.packages("rpart")
library(rpart)
library(rpart.plot)

# Load the Wine Quality dataset
wine_data <- read.csv("D:/Bsc life/10th semester/Data
Mining/Final Project/winequality-white.csv", sep = ";")
```

[illegible]

```

# Fit the decision tree model with Gini Index
decision_tree_gini <- rpart(formula(paste(target_var, "~",
paste(features, collapse = "+"))),
                           data = train_data,
                           method = "class",
                           parms = list(split = "gini"))

# Fit the decision tree model with Gain Ratio
decision_tree_gain_ratio <- rpart(formula(paste(target_var, "~",
paste(features, collapse = "+"))),
                                  data = train_data,
                                  method = "class",
                                  parms = list(split = "gainratio"))

# Make predictions on the test data for each criterion
predictions_info_gain <- predict(decision_tree_info_gain,
test_data, type = "class")
predictions_gini <- predict(decision_tree_gini, test_data, type =
"class")
predictions_gain_ratio <- predict(decision_tree_gain_ratio,
test_data, type = "class")

# Calculate accuracy for each criterion
accuracy_info_gain[i] <- mean(predictions_info_gain ==
test_data$quality)
accuracy_gini[i] <- mean(predictions_gini == test_data$quality)
accuracy_gain_ratio[i] <- mean(predictions_gain_ratio ==
test_data$quality)

# Create confusion matrix for each criterion
confusion_matrices_info_gain[[i]] <- table(Actual =
test_data$quality, Predicted = predictions_info_gain)
confusion_matrices_gini[[i]] <- table(Actual = test_data$quality,
Predicted = predictions_gini)
confusion_matrices_gain_ratio[[i]] <- table(Actual =
test_data$quality, Predicted = predictions_gain_ratio)
}

```

```

# Calculate the average accuracy for each criterion
average_accuracy_info_gain <- mean(accuracy_info_gain)
average_accuracy_gini <- mean(accuracy_gini)
average_accuracy_gain_ratio <- mean(accuracy_gain_ratio)

# Print the average accuracy for each criterion
cat("Average Accuracy with Information Gain:",
    average_accuracy_info_gain, "\n")
cat("Average Accuracy with Gini Index:", average_accuracy_gini,
    "\n")
cat("Average Accuracy with Gain Ratio:",
    average_accuracy_gain_ratio, "\n")

# Print confusion matrices for each criterion
for (i in 1:k) {
    cat("Confusion Matrix for Information Gain (Fold", i, "):\n")
    print(confusion_matrices_info_gain[[i]])

    cat("Confusion Matrix for Gini Index (Fold", i, "):\n")
    print(confusion_matrices_gini[[i]])

    cat("Confusion Matrix for Gain Ratio (Fold", i, "):\n")
    print(confusion_matrices_gain_ratio[[i]])
}

# Visualize decision trees using rpart.plot
#options(repr.plot.width = 1000, repr.plot.height = 500)
# Visualize decision trees using rpart.plot with adjusted plot
dimensions
options(repr.plot.width = 800, repr.plot.height = 400)
print("Decision Tree with Information Gain")
rpart.plot(decision_tree_info_gain)

print("Decision Tree with Gini Index:")
rpart.plot(decision_tree_gini)

print("Decision Tree with Gain Ratio:")
rpart.plot(decision_tree_gain_ratio)

```

```
print("Decision Tree with Information Gain")  
rpart.plot(decision_tree_info_gain)
```

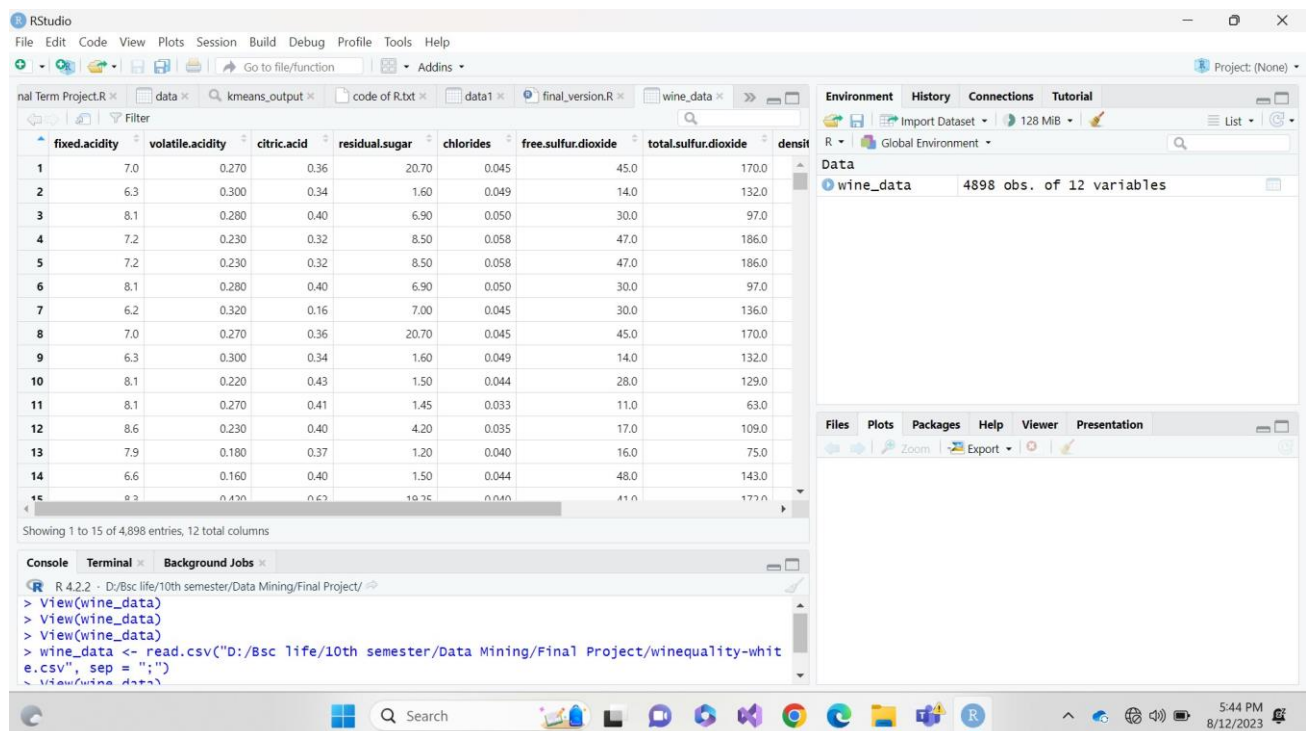
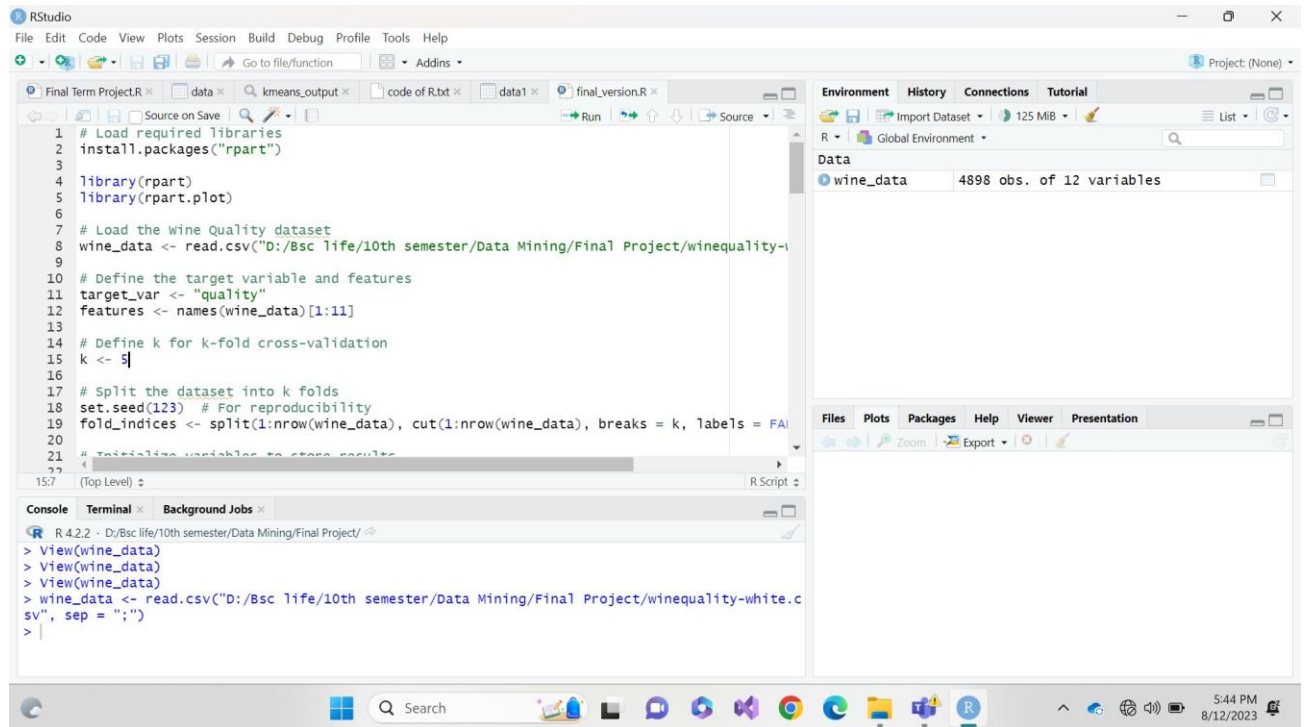
```
print("Decision Tree with Gini Index:")  
rpart.plot(decision_tree_gini)
```

```
print("Decision Tree with Gain Ratio:")  
rpart.plot(decision_tree_gain_ratio)
```

Outputs:

1. Load the Wine Quality dataset

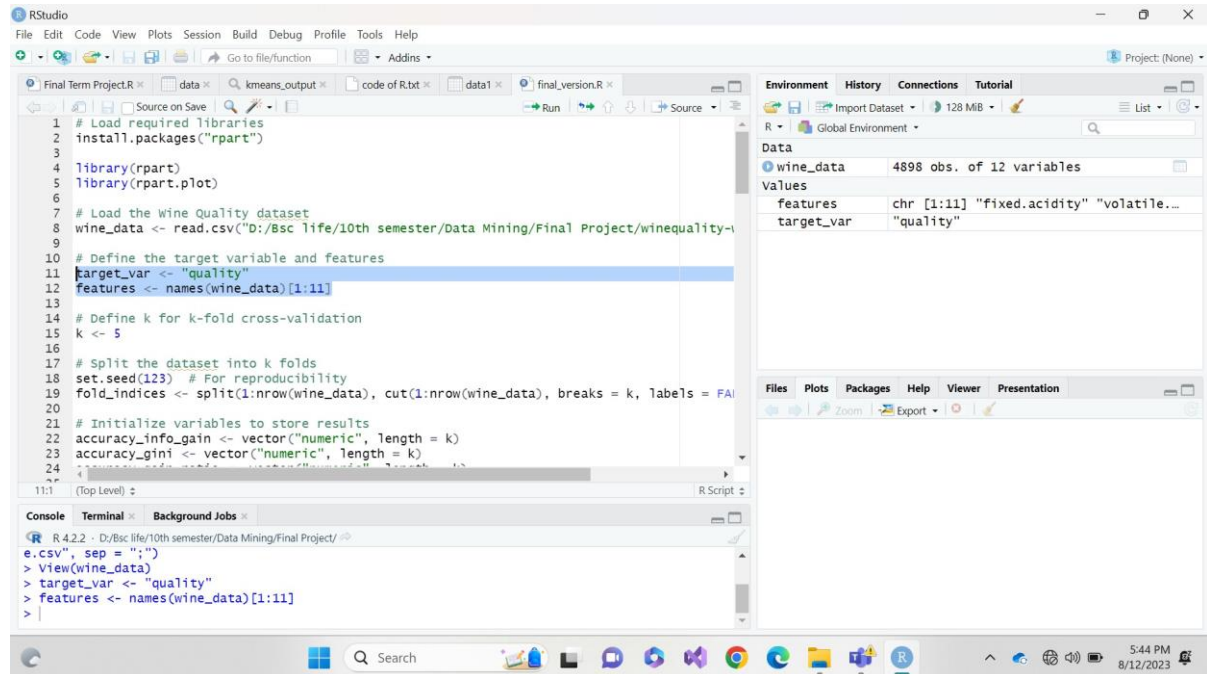
```
wine_data <- read.csv("D:/Bsc life/10th semester/Data Mining/Final Project/winequality-white.csv", sep = ";")
```



2. Define the target variable and features

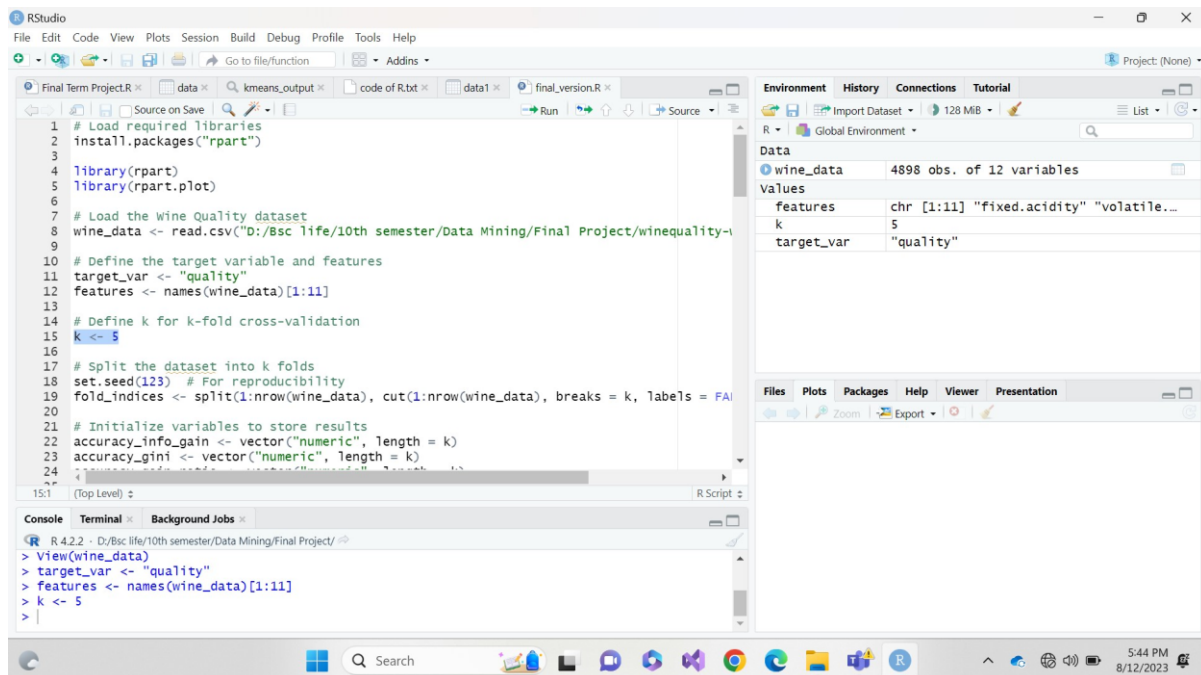
```
target_var <- "quality"
```

```
features <- names(wine_data)[1:11]
```



3. Define k for k-fold cross-validation

```
k <- 5
```

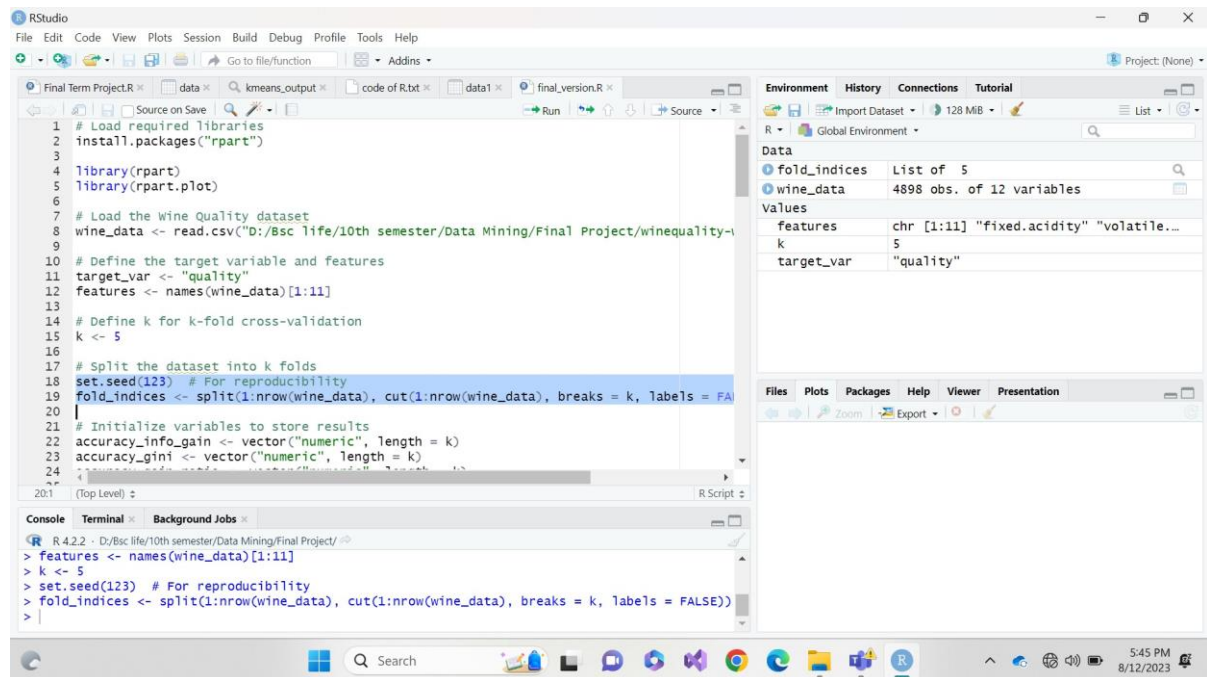


4. Split the dataset into k folds

```
set.seed(123) # For reproducibility
```

```
fold_indices <- split(1:nrow(wine_data), cut(1:nrow(wine_data),
```

```
breaks = k, labels = FALSE))
```



5. Initialize variables to store results

```
accuracy_info_gain <- vector("numeric", length = k)
```

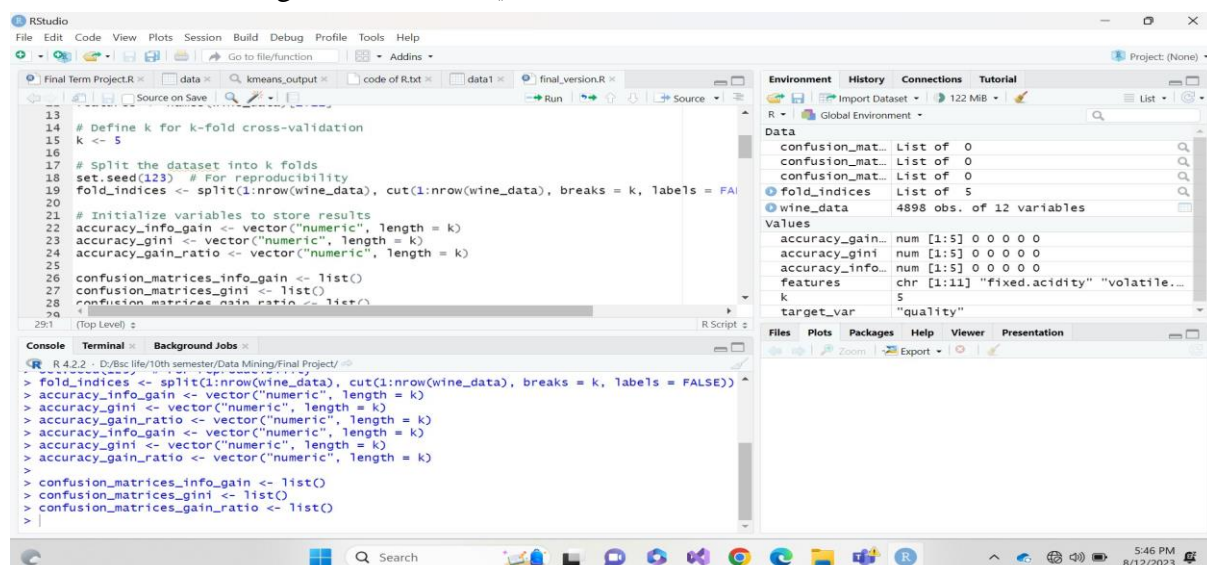
```
accuracy_gini <- vector("numeric", length = k)
```

```
accuracy_gain_ratio <- vector("numeric", length = k)
```

```
confusion_matrices_info_gain <- list()
```

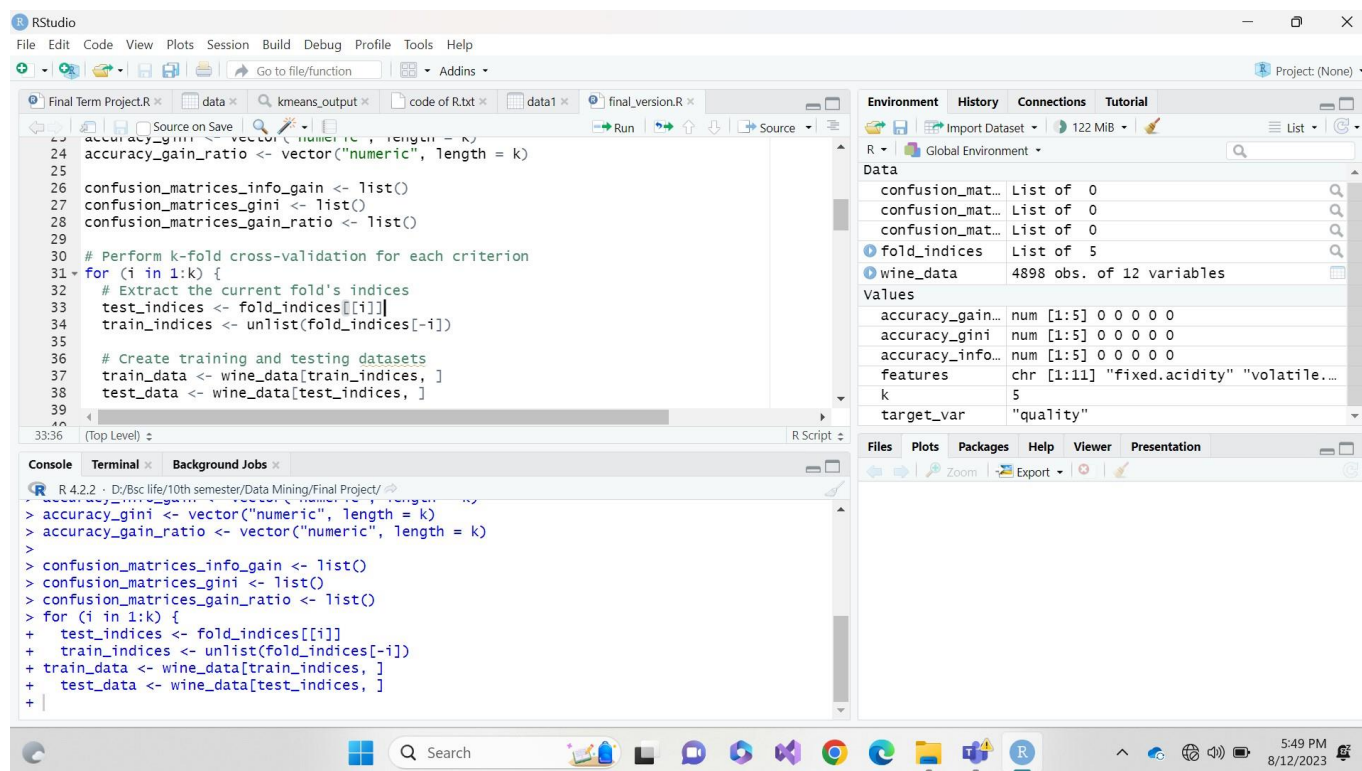
```
confusion_matrices_gini <- list()
```

```
confusion_matrices_gain_ratio <- list()
```



6. Perform k-fold cross-validation for each criterion

```
for (i in 1:k) {  
  # Extract the current fold's indices  
  test_indices <- fold_indices[[i]]  
  train_indices <- unlist(fold_indices[-i])  
  # Create training and testing datasets  
  train_data <- wine_data[train_indices, ]  
  test_data <- wine_data[test_indices, ]  
}
```



7. Fit the decision tree model with Information Gain, Gini Index, Gain Ratio

```
decision_tree_info_gain <- rpart(formula(paste(target_var, "~",  
paste(features, collapse = "+"))),
```

```
    data = train_data,  
    method = "class",  
    parms = list(split = "information"))
```

```
decision_tree_gini <- rpart(formula(paste(target_var, "~",  
paste(features, collapse = "+"))),
```

```
    data = train_data,  
    method = "class",  
    parms = list(split = "gini"))
```

```
decision_tree_gain_ratio <- rpart(formula(paste(target_var, "~",  
paste(features, collapse = "+"))),
```

```
    data = train_data,  
    method = "class",  
    parms = list(split = "gainratio"))
```

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for fitting three decision tree models using the `rpart` package. The code defines `decision_tree_info_gain`, `decision_tree_gini`, and `decision_tree_gain_ratio` using the `formula` argument to specify the target variable and features.
- Environment Pane:** Shows the loaded data environment. The `wine_data` dataset is listed with 4898 observations and 12 variables. The `values` section shows the structure of several variables: `accuracy_gain...` (numeric, 1:5), `accuracy_gini` (numeric, 1:5), `accuracy_info...` (numeric, 1:5), `features` (character, 1:11), `k` (integer, 5), and `target_var` (character, "quality").
- Console:** Shows the execution of the R code, with the output of the `rpart` function calls visible.
- Terminal:** Displays the R session output, including the path to the R installation and the current working directory.

8. Make predictions on the test data for each criterion

```
predictions_info_gain <- predict(decision_tree_info_gain,  
test_data, type = "class")
```

```
predictions_gini <- predict(decision_tree_gini, test_data, type =  
"class")
```

```
predictions_gain_ratio <- predict(decision_tree_gain_ratio,  
test_data, type = "class")
```

The screenshot displays the RStudio environment with the following components:

- Script Editor:** Contains R code for fitting decision tree models using `rpart` and making predictions on test data.
- Environment:** Lists objects in the global environment, including `confusion_mat...`, `fold_indices`, and `wine_data`.
- Console:** Shows the execution output of the R script, including the fitting and prediction steps.
- Files:** Lists files in the project directory, including `data`, `kmeans_output`, `code of R.txt`, `data1`, and `final_version.R`.

R Code in Script Editor:

```
47 decision_tree_gini <- rpart(formula(paste(target_var, "~", paste(features, collapse = "  
48 data = train_data,  
49 method = "class",  
50 parms = list(split = "gini"))  
51  
52 # Fit the decision tree model with Gain Ratio  
53 decision_tree_gain_ratio <- rpart(formula(paste(target_var, "~", paste(features, collap  
54 data = train_data,  
55 method = "class",  
56 parms = list(split = "gainratio"))  
57  
58 # Make predictions on the test data for each criterion  
59 predictions_info_gain <- predict(decision_tree_info_gain, test_data, type = "class")  
60 predictions_gini <- predict(decision_tree_gini, test_data, type = "class")  
61 predictions_gain_ratio <- predict(decision_tree_gain_ratio, test_data, type = "class")  
62  
63 # Calculate accuracy for each criterion  
64
```

Environment Panel:

Object	Class	Value
confusion_mat...	List of 0	
confusion_mat...	List of 0	
confusion_mat...	List of 0	
fold_indices	List of 5	
wine_data	4898 obs. of 12 variables	

Console Output:

```
R 4.2.2 - D:/Bsc life/10th semester/Data Mining/Final Project/ >  
+ method = "class",  
+ parms = list(split = "gini"))  
+ decision_tree_gain_ratio <- rpart(formula(paste(target_var, "~", paste(features, collapse =  
+))),  
+ data = train_data,  
+ method = "class",  
+ parms = list(split = "gainratio"))  
+ predictions_info_gain <- predict(decision_tree_info_gain, test_data, type = "class")  
+ predictions_gini <- predict(decision_tree_gini, test_data, type = "class")  
+ predictions_gain_ratio <- predict(decision_tree_gain_ratio, test_data, type = "class")  
+
```

9. Calculate accuracy for each criterion

```
accuracy_info_gain[i] <- mean(predictions_info_gain ==  
test_data$quality)
```

```
accuracy_gini[i] <- mean(predictions_gini == test_data$quality)
```

```
accuracy_gain_ratio[i] <- mean(predictions_gain_ratio ==  
test_data$quality)
```

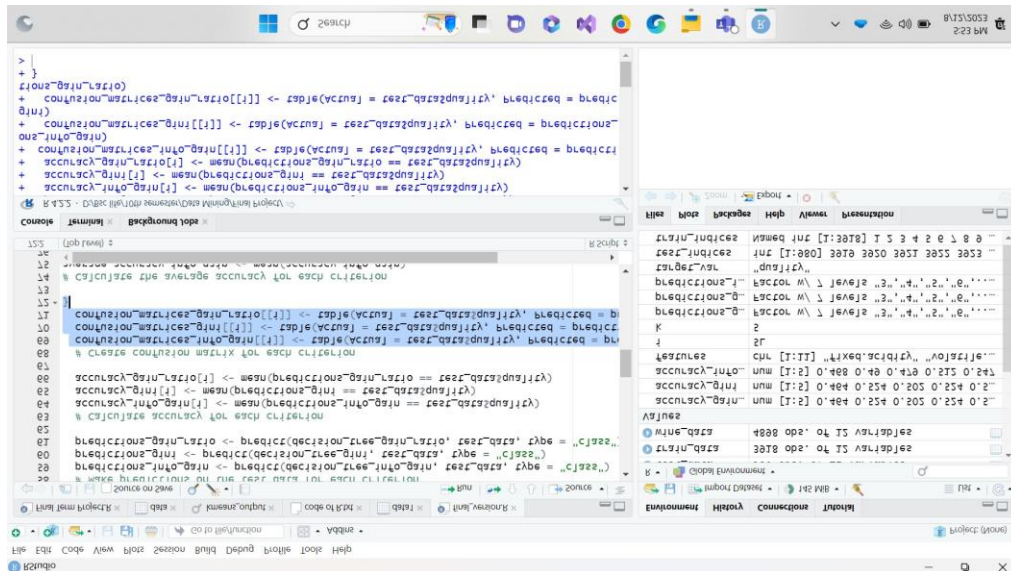
The screenshot shows the RStudio interface with the following components:

- Script Editor:** Contains R code for fitting a decision tree model and calculating accuracy for each criterion. The code is as follows:

```
51  
52 # Fit the decision tree model with Gain Ratio  
53 decision_tree_gain_ratio <- rpart(formula(paste(target_var, "~", paste(features, coll:  
54 data = train_data,  
55 method = "class",  
56 parms = list(split = "gainratio"))  
57  
58 # Make predictions on the test data for each criterion  
59 predictions_info_gain <- predict(decision_tree_info_gain, test_data, type = "class")  
60 predictions_gini <- predict(decision_tree_gini, test_data, type = "class")  
61 predictions_gain_ratio <- predict(decision_tree_gain_ratio, test_data, type = "class")  
62  
63 # Calculate accuracy for each criterion  
64 accuracy_info_gain[i] <- mean(predictions_info_gain == test_data$quality)  
65 accuracy_gini[i] <- mean(predictions_gini == test_data$quality)  
66 accuracy_gain_ratio[i] <- mean(predictions_gain_ratio == test_data$quality)  
67  
68
```
- Environment:** Shows the following objects:
 - confusion_mat... List of 0
 - confusion_mat... List of 0
 - confusion_mat... List of 0
 - fold_indices List of 5
 - wine_data 4898 obs. of 12 variables
- Console:** Shows the output of the R script, including the model fit and the calculation of accuracy for each criterion.

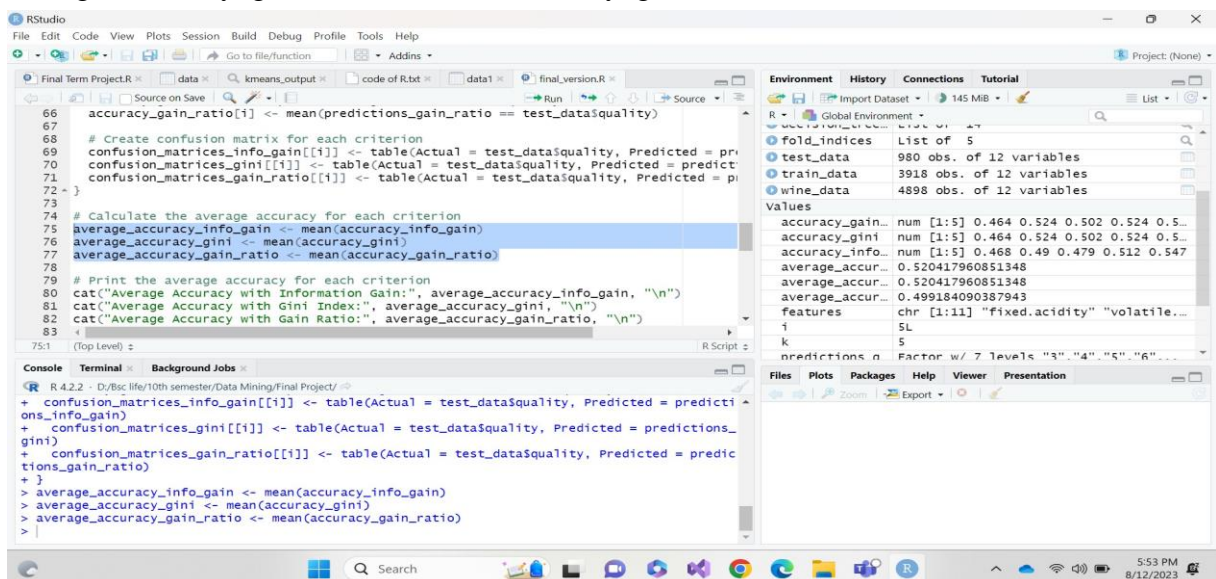
10. Create confusion matrix for each criterion

```
confusion_matrices_info_gain[[i]] <- table(Actual =  
test_data$quality, Predicted = predictions_info_gain)  
confusion_matrices_gini[[i]] <- table(Actual = test_data$quality,  
Predicted = predictions_gini)  
confusion_matrices_gain_ratio[[i]] <- table(Actual =  
test_data$quality, Predicted = predictions_gain_ratio)  
}
```



11. Calculate the average accuracy for each criterion

```
average_accuracy_info_gain <- mean(accuracy_info_gain)  
average_accuracy_gini <- mean(accuracy_gini)  
average_accuracy_gain_ratio <- mean(accuracy_gain_ratio)
```



12. Print the average accuracy for each criterion

```
cat("Average Accuracy with Information Gain:",  
    average_accuracy_info_gain, "\n")  
cat("Average Accuracy with Gini Index:", average_accuracy_gini,  
    "\n")  
cat("Average Accuracy with Gain Ratio:",  
    average_accuracy_gain_ratio, "\n")
```

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for calculating average accuracy for each criterion (Information Gain, Gini Index, Gain Ratio) across 5 folds. The code includes comments and uses `mean()` to calculate the average accuracy for each criterion.
- Environment:** Lists the objects in the global environment: `fold_indices` (List of 5), `test_data` (980 obs. of 12 variables), `train_data` (3918 obs. of 12 variables), and `wine_data` (4898 obs. of 12 variables). It also shows the values for `accuracy_gain...`, `accuracy_gini`, `accuracy_info...`, `average_accu...`, `average_accu...`, `features`, `i`, `k`, and `predictions`.
- Console:** Displays the output of the R code, showing the average accuracy for each criterion across 5 folds. The output is as follows:

```
> }  
> average_accuracy_info_gain <- mean(accuracy_info_gain)  
> average_accuracy_gini <- mean(accuracy_gini)  
> average_accuracy_gain_ratio <- mean(accuracy_gain_ratio)  
> cat("Average Accuracy with Information Gain:", average_accuracy_info_gain, "\n")  
Average Accuracy with Information Gain: 0.4991841  
> cat("Average Accuracy with Gini Index:", average_accuracy_gini, "\n")  
Average Accuracy with Gini Index: 0.520418  
> cat("Average Accuracy with Gain Ratio:", average_accuracy_gain_ratio, "\n")  
Average Accuracy with Gain Ratio: 0.520418  
>
```

13. Print confusion matrices for each criterion

```
for (i in 1:k) {  
  cat("Confusion Matrix for Information Gain (Fold", i, "):\n")  
  print(confusion_matrices_info_gain[[i]])  
  
  cat("Confusion Matrix for Gini Index (Fold", i, "):\n")  
  print(confusion_matrices_gini[[i]])  
  
  cat("Confusion Matrix for Gain Ratio (Fold", i, "):\n")  
  print(confusion_matrices_gain_ratio[[i]])  
}
```

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R code for printing confusion matrices for each criterion across 5 folds.
- Environment:** Lists the objects in the Global Environment: `fold_indices` (List of 5), `test_data` (980 obs. of 12 variables), `train_data` (3918 obs. of 12 variables), and `wine_data` (4898 obs. of 12 variables).
- Console:** Displays the output of the R code for Fold 1.
 - Confusion Matrix for Information Gain (Fold 1):**

	Actual \ Predicted	3	4	5	6	7	8	9
Actual 3	0	0	2	4	0	0	0	0
Actual 4	0	0	16	19	0	0	0	0
Actual 5	0	0	156	155	0	0	0	0
Actual 6	0	0	101	303	0	0	0	0
Actual 7	0	0	11	172	0	0	0	0
Actual 8	0	0	1	36	0	0	0	0
Actual 9	0	0	0	4	0	0	0	0
 - Confusion Matrix for Gini Index (Fold 1):**

	Actual \ Predicted	3	4	5	6	7	8	9
Actual 3	0	0	2	3	1	0	0	0
Actual 4	0	0	21	13	1	0	0	0
Actual 5	0	0	204	106	1	0	0	0
Actual 6	0	0	163	237	4	0	0	0
Actual 7	0	0	30	139	14	0	0	0

14. Visualize decision trees using rpart.plot

```
#options(repr.plot.width = 1000, repr.plot.height = 500)
```

```
# Visualize decision trees using rpart.plot with adjusted plot dimensions
```

```
options(repr.plot.width = 800, repr.plot.height = 400)
```

```
print("Decision Tree with Information Gain")
```

```
rpart.plot(decision_tree_info_gain)
```

```
print("Decision Tree with Gini Index:")
```

```
rpart.plot(decision_tree_gini)
```

```
print("Decision Tree with Gain Ratio:")
```

```
rpart.plot(decision_tree_gain_ratio)
```

```
print("Decision Tree with Information Gain")
```

```
rpart.plot(decision_tree_info_gain)
```

```
print("Decision Tree with Gini Index:")
```

```
rpart.plot(decision_tree_gini)
```

```
print("Decision Tree with Gain Ratio:")
```

```
rpart.plot(decision_tree_gain_ratio)
```

