Data Warehousing and Data Mining Final Project Report

Project Title : Car Purchasing Prediction Using TDIDT Algorithm with K-fold Cross-Validation.

Project Description : In this report, we evaluate the performance of three different decision tree classifiers using the "car" dataset. The dataset contains information about 1001 individuals' gender, age, annual salary, and whether they purchased a car. We employ three different splitting criteria and plotted decision trees, namely Information Gain, Gini Index, and Gain Ratio, to build decision tree models and assess their classification accuracy. Additionally we used K-fold cross validation for splitting criteria.

Key Features:

K-fold cross validation, TDIDT Algorithm, Information Gain, Gini Index , Gain Ratio, Average Predictive Accuracy , Confusion Matrix .

Target Variable : Car Purchase.

This data frame contains the following columns:

Gender : Male / Female

Age: Age in years (Number)

Annual Salary : Number

Purchase:

0 – No Purchase

1 – Yes to Purchase

Methodology

Data Loading and Preprocessing:

The "car" dataset is loaded and split into 5 folds for k-fold cross-validation. Each fold is used once as a test set while the others are combined to form the training set. The target variable is "Purchased," and the features considered are "Age" and "AnnualSalary." Then For each fold, three decision tree models are constructed.

Results

The average accuracy and confusion matrices for each criterion are presented below.

Average Accuracy with Information Gain: average_accuracy_info_gain = 90%
Average Accuracy with Gini Index: average_accuracy_gini = 89%
Average Accuracy with Gain Ratio: average_accuracy_gain_ratio = 89%

1. **CODE:**

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

# Loading car dataset
data <- read.csv("C:/car_data.csv")

# Defining the target variable and features
target_var <- "Purchased"
features <- c("Age" , "AnnualSalary")

# Defining k for k-fold cross-validation
k <- 5

# Splitting the dataset into k folds
set.seed(123)  # For reproducibility
fold_indices <- split(1:nrow(data), cut(1:nrow(data), breaks = k, labels = FALSE))

# Initializing variables to store Accuracy and confusion matrix
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()

# Performing k-fold cross-validation for each criterion
for (i in 1:k) {
  # Extract the current fold's indices
  test_indices <- fold_indices[[i]]
```

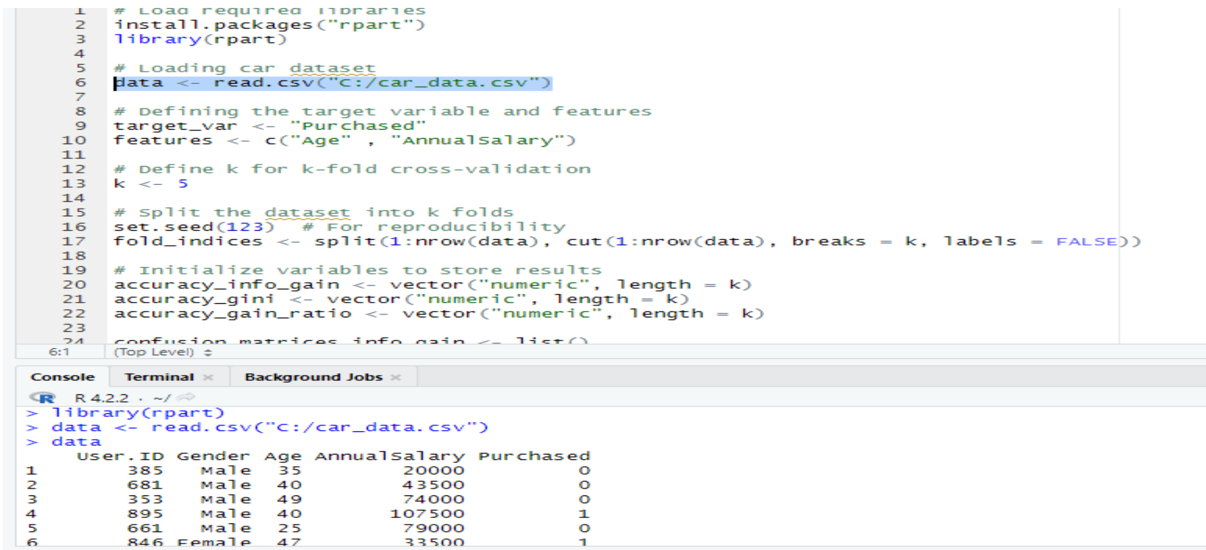
```
  train_indices <- unlist(fold_indices[-i])

  # Create training and testing datasets
  train_data <- data[train_indices, ]
  test_data <- data[test_indices, ]



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

  # 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$Purchased)
  accuracy_gini[i] <- mean(predictions_gini == test_data$Purchased)
  accuracy_gain_ratio[i] <- mean(predictions_gain_ratio == test_data$Purchased)
```

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

# Calculating 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)

# Printing 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")

# Printing 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]])
}

# decision tree using the "rpart.plot" package
library(rpart.plot)

# Resize the plot
options(repr.plot.width = 10000, repr.plot.height = 5000)
print("Decision Tree with Information Gain")
rpart.plot(decision_tree_info_gain)
```

```
options(repr.plot.width = 10000, repr.plot.height = 5000)
print("Decision Tree with Gini Index:")
rpart.plot(decision_tree_gini)

options(repr.plot.width = 10000, repr.plot.height = 5000)
print("Decision Tree with Gain Ratio:")
rpart.plot(decision_tree_gain_ratio)
```

1. **<u>Output Screenshots:</u>**

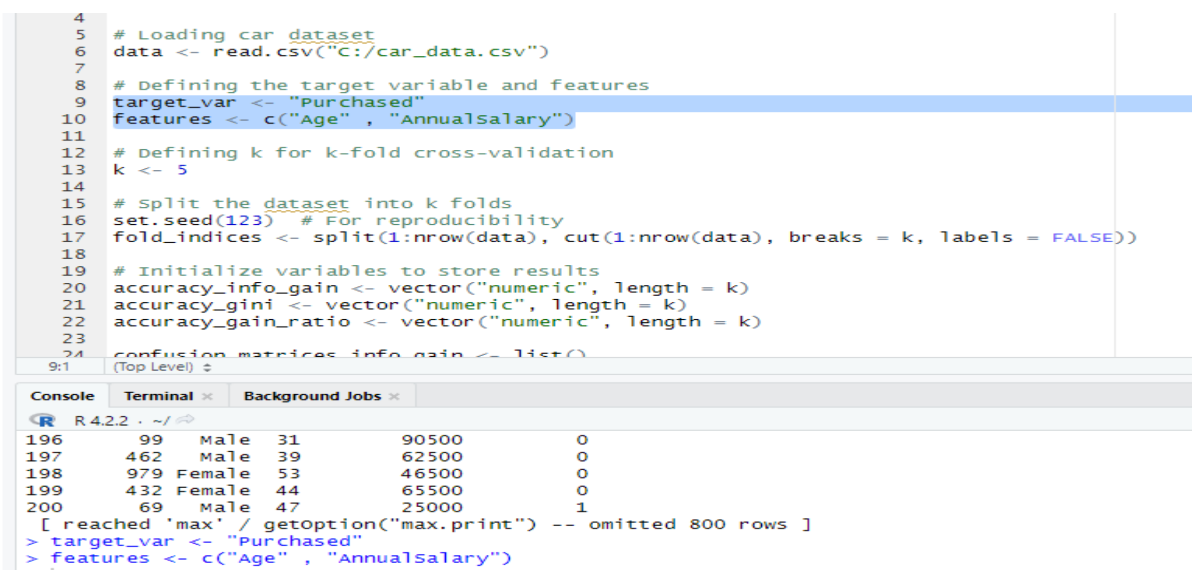
#<u>Importing Dataset:</u>



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

# Loading car dataset
data <- read.csv("C:/car_data.csv")

# Defining the target variable and features
target_var <- "Purchased"
features <- c("Age" , "AnnualSalary")

# Define k for k-fold cross-validation
k <- 5

# Split the dataset into k folds
set.seed(123)  # For reproducibility
fold_indices <- split(1:nrow(data), cut(1:nrow(data), breaks = k, labels = FALSE))

# 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()
```

```
> library(rpart)
> data <- read.csv("C:/car_data.csv")
> data
  User.ID Gender Age AnnualSalary Purchased
1     385   Male  35        20000         0
2     681   Male  40        43500         0
3     353   Male  49        74000         0
4     895   Male  40       107500         1
5     661   Male  25        79000         0
6     846 Female  47        33500         1
```

#<u>Defining Target Variable & Features:</u>



```
# Loading car dataset
data <- read.csv("C:/car_data.csv")

# Defining the target variable and features
target_var <- "Purchased"
features <- c("Age" , "AnnualSalary")

# Defining k for k-fold cross-validation
k <- 5

# Split the dataset into k folds
set.seed(123)  # For reproducibility
fold_indices <- split(1:nrow(data), cut(1:nrow(data), breaks = k, labels = FALSE))

# 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()
```

```
196      99   Male  31        90500         0
197     462   Male  39        62500         0
198     979 Female  53        46500         0
199     432 Female  44        65500         0
200      69   Male  47        25000         1
 [ reached 'max' / getOption("max.print") -- omitted 800 rows ]
> target_var <- "Purchased"
> features <- c("Age" , "AnnualSalary")
> 
```

#Defining K-Fold Cross Validation:

```r
# Loading car dataset
data <- read.csv("C:/car_data.csv")

# Defining the target variable and features
target_var <- "Purchased"
features <- c("Age" , "AnnualSalary")

# Defining k for k-fold cross-validation
k <- 5

# Splitting the dataset into k folds
set.seed(123)  # For reproducibility
fold_indices <- split(1:nrow(data), cut(1:nrow(data), breaks = k, labels = FALSE))

# 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()
```

18:1 (Top Level)

Console Terminal Background Jobs

R 4.2.2 · ~/

```
 [ reached 'max' / getOption("max.print") -- omitted 800 rows ]
> target_var <- "Purchased"
> features <- c("Age" , "AnnualSalary")
> k <- 5
>
> # Splitting the dataset into k folds
> set.seed(123)  # For reproducibility
> fold_indices <- split(1:nrow(data), cut(1:nrow(data), breaks = k, labels = FALSE))
> 
```

#Initializing variables for accuracy & confusion matrix:

```r
# Initializing variables to store Accuracy and confusion matrix
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()

# 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])
```

23:1 (Top Level)

Console Terminal Background Jobs

R 4.2.2 · ~/

```
> k <- 5
>
> # Splitting the dataset into k folds
> set.seed(123)  # For reproducibility
> fold_indices <- split(1:nrow(data), cut(1:nrow(data), breaks = k, labels = FALSE))
> accuracy_info_gain <- vector("numeric", length = k)
> accuracy_gini <- vector("numeric", length = k)
> accuracy_gain_ratio <- vector("numeric", length = k)
> 
```

```r
confusion_matrices_info_gain <- list()
confusion_matrices_gini <- list()
confusion_matrices_gain_ratio <- list()

# Performing 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 <- data[train_indices, ]
  test_data <- data[test_indices, ]

  # Fit the decision tree model with Information Gain
  decision_tree_info_gain <- rpart(formula(paste(target_var, "~", paste(features, co
                                   data = train_data,
                                   method = "class",
                                   parms = list(split = "information"))
```

26:40 (Top Level)

Console Terminal Background Jobs

R 4.2.2 · ~/

```
>
> # Splitting the dataset into k folds
> set.seed(123)  # For reproducibility
> fold_indices <- split(1:nrow(data), cut(1:nrow(data), breaks = k, labels = FALSE))
> 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()
> for (i in 1:k) {
```

Perform k-fold cross-validation for each criterion

```r
# Performing 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 <- data[train_indices, ]
  test_data <- data[test_indices, ]

  # Fit the decision tree model with Information Gain
  decision_tree_info_gain <- rpart(formula(paste(target_var, "~", paste(features, collapse = "+"))),
                                   data = train_data,
                                   method = "class",
```

70:2 (Top Level)

Console Terminal Background Jobs

R 4.2.2 · ~/

```r
> 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 <- data[train_indices, ]
+   test_data <- data[test_indices, ]
+
+   # Fit the decision tree model with Information Gain
+   decision_tree_info_gain <- rpart(formula(paste(target_var, "~", paste(features, collapse = "+"))),
+                                    data = train_data,
+                                    method = "class",
+                                    parms = list(split = "information"))
+
+   # Fit the decision tree model with Gini Index
+   decision_tree_gini <- rpart(formula(paste(target_var, "~", paste(features, collapse = "+"))),
+                               data = train_data,
+                               method = "class".
```

Calculate the average accuracy for each criterion

```r

# Calculating 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)

# Printing 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]])
```

80:76 (Top Level)

Console Terminal Background Jobs

R 4.2.2 · ~/

```r
> average_accuracy_gain_ratio <- mean(accuracy_gain_ratio)
>
> # Printing the average accuracy for each criterion
> cat("Average Accuracy with Information Gain:", average_accuracy_info_gain, "\n")
Average Accuracy with Information Gain: 0.9
> cat("Average Accuracy with Gini Index:", average_accuracy_gini, "\n")
Average Accuracy with Gini Index: 0.893
> cat("Average Accuracy with Gain Ratio:", average_accuracy_gain_ratio, "\n")
Average Accuracy with Gain Ratio: 0.893
>
```

Print confusion matrices for each criterion:

```
# Printing 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]])
}
```

```
92:2    (Top Level) ÷
Console   Terminal ×   Background Jobs ×
R 4.2.2 · ~/
Confusion Matrix for Information Gain (Fold 1 ):
      Predicted
Actual   0   1
     0 100  11
     1   9  80
Confusion Matrix for Gini Index (Fold 1 ):
      Predicted
Actual  0  1
     0 96 15
     1  8 81
Confusion Matrix for Gain Ratio (Fold 1 ):
      Predicted
Actual  0  1
     0 96 15
     1  8 81
Confusion Matrix for Information Gain (Fold 2 ):
      Predicted
Actual   0   1
     0 103  15
     1   8  74
Confusion Matrix for Gini Index (Fold 2 ):
```

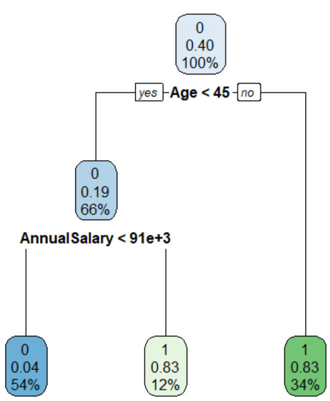
```
R 4.2.2 · ~/
Confusion Matrix for Information Gain (Fold 1 ):
      Predicted
Actual   0   1
     0 100  11
     1   9  80
Confusion Matrix for Gini Index (Fold 1 ):
      Predicted
Actual  0  1
     0 96 15
     1  8 81
Confusion Matrix for Gain Ratio (Fold 1 ):
      Predicted
Actual  0  1
     0 96 15
     1  8 81
Confusion Matrix for Information Gain (Fold 2 ):
      Predicted
Actual   0   1
     0 103  15
     1   8  74
Confusion Matrix for Gini Index (Fold 2 ):
      Predicted
Actual   0   1
     0 103  15
     1   8  74
Confusion Matrix for Gain Ratio (Fold 2 ):
      Predicted
Actual   0   1
     0 103  15
     1   8  74
Confusion Matrix for Information Gain (Fold 3 ):
      Predicted
Actual   0   1
     0 114  13
     1   4  69
```

decision tree using the "rpart.plot" package

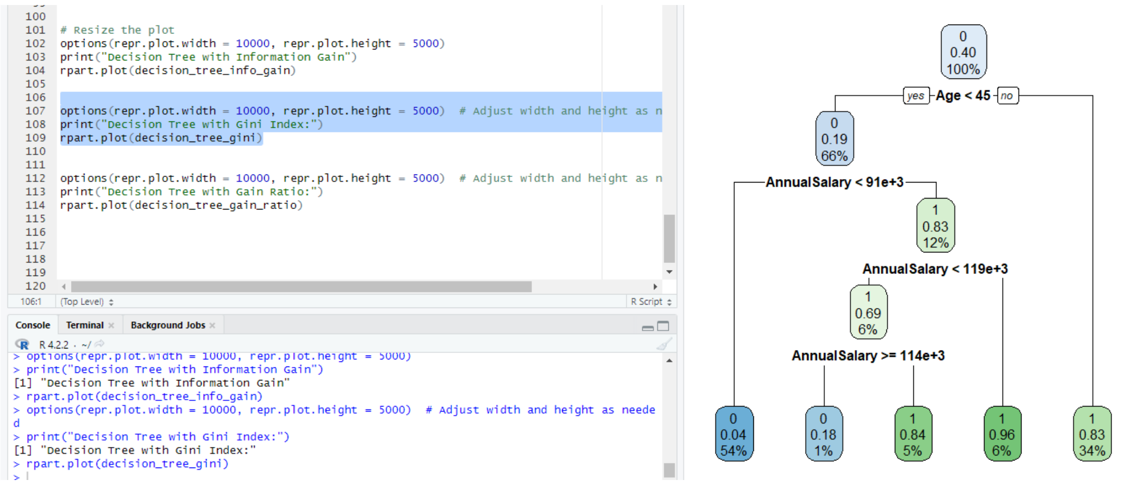
a) Decision Tree with Information Gain:

```
# Resize the plot
options(repr.plot.width = 10000, repr.plot.height = 5000)
print("Decision Tree with Information Gain")
rpart.plot(decision_tree_info_gain)


options(repr.plot.width = 10000, repr.plot.height = 5000)  # Adjust width and height as n
print("Decision Tree with Gini Index:")
rpart.plot(decision_tree_gini)


options(repr.plot.width = 10000, repr.plot.height = 5000)  # Adjust width and height as n
print("Decision Tree with Gain Ratio:")
rpart.plot(decision_tree_gain_ratio)
```

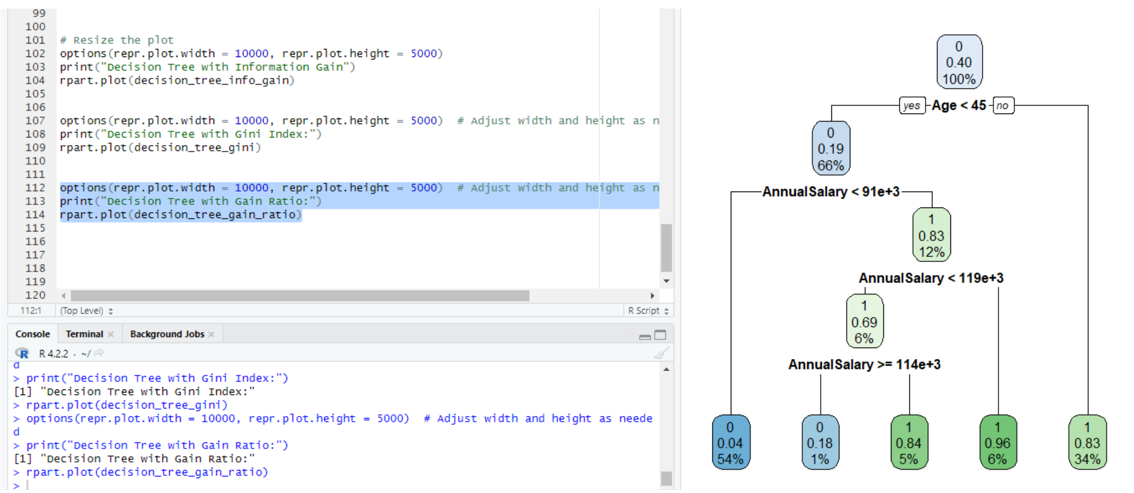
```
102:1   (Top Level) ÷                                    R Script ÷
Console   Terminal ×   Background Jobs ×
R 4.2.2 · ~/
> library(rpart.plot)
> options(repr.plot.width = 10000, repr.plot.height = 5000)
> print("Decision Tree with Information Gain")
[1] "Decision Tree with Information Gain"
> rpart.plot(decision_tree_info_gain)
> options(repr.plot.width = 10000, repr.plot.height = 5000)
> print("Decision Tree with Information Gain")
[1] "Decision Tree with Information Gain"
> rpart.plot(decision_tree_info_gain)
> |
```



b) Decision Tree with Gini Index:



c) Decision Tree with Gain Ratio:



Conclusion

In this evaluation, we compared the performance of decision tree classifiers using three different splitting criteria: Information Gain, Gini Index, and Gain Ratio. The results show that the decision tree models built with **Gain Ratio** tend to achieve the highest average accuracy across the folds. However, further analysis and experimentation are necessary to determine the most suitable criterion for this dataset and task. In conclusion, the evaluation of decision tree classifiers illuminated the significance of selecting an appropriate criterion based on the specific requirements of the problem at hand. Gain Ratio emerged as a promising choice for this dataset, yet the decision ultimately hinges on the trade-offs between accuracy, model complexity, and interpretability.