Exercise 10 Advanced Methods for Regression and Classification

Stefan Merdian

2025-01-04

```r
data("Caravan", package="ISLR")
head(Caravan)
```

```
##   MOSTYPE MAANTHUI MGEMOMV MGEMLEEF MOSHOOFD MGODRK MGODPR MGODOV MGODGE MRELGE
## 1      33        1       3        2        8      0      5      1      3      7
## 2      37        1       2        2        8      1      4      1      4      6
## 3      37        1       2        2        8      0      4      2      4      3
## 4       9        1       3        3        3      2      3      2      4      5
## 5      40        1       4        2       10      1      4      1      4      7
## 6      23        1       2        1        5      0      5      0      5      0
##   MRELSA MRELOV MFALLEEN MFGEKIND MFWEKIND MOPLHOOG MOPLMIDD MOPLLAAG MBERHOOG
## 1      0      2        1        2        6        1        2        7        1
## 2      2      2        0        4        5        0        5        4        0
## 3      2      4        4        4        2        0        5        4        0
## 4      2      2        2        3        4        3        4        2        4
## 5      1      2        2        4        4        5        4        0        0
## 6      6      3        3        5        2        0        5        4        2
##   MBERZELF MBERBOER MBERMIDD MBERARBG MBERARBO MSKA MSKB1 MSKB2 MSKC MSKD
## 1        0        1        2        5        2    1     1     2    6    1
## 2        0        0        5        0        4    0     2     3    5    0
## 3        0        0        7        0        2    0     5     0    4    0
## 4        0        0        3        1        2    3     2     1    4    0
## 5        5        4        0        0        0    9     0     0    0    0
## 6        0        0        4        2        2    2     2     2    4    2
##   MHHUUR MHKOOP MAUT1 MAUT2 MAUT0 MZFONDS MZPART MINKM30 MINK3045 MINK4575
## 1      1      8     8     0     1       8      1       0        4        5
## 2      2      7     7     1     2       6      3       2        0        5
## 3      7      2     7     0     2       9      0       4        5        0
## 4      5      4     9     0     0       7      2       1        5        3
## 5      4      5     6     2     1       5      4       0        0        9
## 6      9      0     5     3     3       9      0       5        2        3
##   MINK7512 MINK123M MINKGEM MKOOPKLA PWAPART PWABEDR PWALAND PPERSAUT PBESAUT
## 1        0        0       4        3       0       0       0        6       0
## 2        2        0       5        4       2       0       0        0       0
## 3        0        0       3        4       2       0       0        6       0
## 4        0        0       4        4       0       0       0        6       0
## 5        0        0       6        3       0       0       0        0       0
## 6        0        0       3        3       0       0       0        6       0
##   PMOTSCO PVRAAUT PAANHANG PTRACTOR PWERKT PBROM PLEVEN PPERSONG PGEZONG
## 1       0       0        0        0      0     0      0        0       0
## 2       0       0        0        0      0     0      0        0       0
## 3       0       0        0        0      0     0      0        0       0
## 4       0       0        0        0      0     0      0        0       0
## 5       0       0        0        0      0     0      0        0       0
## 6       0       0        0        0      0     0      0        0       0
##   PWAOREG PBRAND PZEILPL PPLEZIER PFIETS PINBOED PBYSTAND AWAPART AWABEDR
## 1       0      5       0        0      0       0        0       0       0
## 2       0      2       0        0      0       0        0       2       0
## 3       0      2       0        0      0       0        0       1       0
## 4       0      2       0        0      0       0        0       0       0
## 5       0      6       0        0      0       0        0       0       0
## 6       0      0       0        0      0       0        0       0       0
##   AWALAND APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT ABROM
## 1       0        1       0       0       0        0        0      0     0
## 2       0        0       0       0       0        0        0      0     0
## 3       0        1       0       0       0        0        0      0     0
## 4       0        1       0       0       0        0        0      0     0
## 5       0        0       0       0       0        0        0      0     0
## 6       0        1       0       0       0        0        0      0     0
##   ALEVEN APERSONG AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED
## 1      0        0       0       0      1       0        0      0       0
## 2      0        0       0       0      1       0        0      0       0
## 3      0        0       0       0      1       0        0      0       0
## 4      0        0       0       0      1       0        0      0       0
## 5      0        0       0       0      1       0        0      0       0
## 6      0        0       0       0      0       0        0      0       0
##   ABYSTAND Purchase
## 1        0       No
## 2        0       No
## 3        0       No
## 4        0       No
## 5        0       No
## 6        0       No
```

```
dim(Caravan)
```

```
## [1] 5822   86
```

Preprocessing

```
threshold <- 0.7

numeric_vars <- sapply(Caravan, is.numeric)
numeric_data <- Caravan[, numeric_vars]

cor_matrix <- cor(numeric_data)


high_cor_pairs <- which(abs(cor_matrix) > threshold & lower.tri(cor_matrix), arr.ind = TRUE)

print("Highly correlated variable pairs:")
```

```
## [1] "Highly correlated variable pairs:"
```

```r
print(high_cor_pairs)
```

```
##          row col
## MOSHOOFD   5   1
## MFWEKIND  15   3
## MGODGE     9   7
## MRELOV    12  10
## MFALLEEN  13  12
## MOPLLAAG  18  17
## MHKOOP    31  30
## MAUTO     34  32
## MZPART    36  35
## AWAPART   65  44
## AWABEDR   66  45
## AWALAND   67  46
## APERSAUT  68  47
## ABESAUT   69  48
## AMOTSCO   70  49
## AVRAAUT   71  50
## AAANHANG  72  51
## ATRACTOR  73  52
## AWERKT    74  53
## ABROM     75  54
## ALEVEN    76  55
## APERSONG  77  56
## AGEZONG   78  57
## AWAOREG   79  58
## ABRAND    80  59
## AZEILPL   81  60
## APLEZIER  82  61
## AFIETS    83  62
## AINBOED   84  63
## ABYSTAND  85  64
```

```r
exclude_vars <- unique(colnames(cor_matrix)[high_cor_pairs[, 2]])

print("Variables to exclude:")
```

```
## [1] "Variables to exclude:"
```

```r
print(exclude_vars)
```

```
##  [1] "MOSTYPE"  "MGEMOMV"  "MGODPR"   "MRELGE"   "MRELOV"   "MOPLMIDD"
##  [7] "MHHUUR"   "MAUT1"    "MZFONDS"  "PWAPART"  "PWABEDR"  "PWALAND"
## [13] "PPERSAUT" "PBESAUT"  "PMOTSCO"  "PVRAAUT"  "PAANHANG" "PTRACTOR"
## [19] "PWERKT"   "PBROM"    "PLEVEN"   "PPERSONG" "PGEZONG"  "PWAOREG"
## [25] "PBRAND"   "PZEILPL"  "PPLEZIER" "PFIETS"   "PINBOED"  "PBYSTAND"
```

```r
cleaned_data <- Caravan[, !(colnames(Caravan) %in% exclude_vars)]

print("Cleaned dataset (variables with high correlation removed):")
```

```
## [1] "Cleaned dataset (variables with high correlation removed):"
```

```r
head(cleaned_data)
```

```
##   MAANTHUI MGEMLEEF MOSHOOFD MGODRK MGODOV MGODGE MRELSA MFALLEEN MFGEKIND
## 1        1        2        8      0      1      3      0        1        2
## 2        1        2        8      1      1      4      2        0        4
## 3        1        2        8      0      2      4      2        4        4
## 4        1        3        3      2      2      4      2        2        3
## 5        1        2       10      1      1      4      1        2        4
## 6        1        1        5      0      0      5      6        3        5
##   MFWEKIND MOPLHOOG MOPLLAAG MBERHOOG MBERZELF MBERBOER MBERMIDD MBERARBG
## 1        6        1        7        1        0        1        2        5
## 2        5        0        4        0        0        0        5        0
## 3        2        0        4        0        0        0        7        0
## 4        4        3        2        4        0        0        3        1
## 5        4        5        0        0        5        4        0        0
## 6        2        0        4        2        0        0        4        2
##   MBERARBO MSKA MSKB1 MSKB2 MSKC MSKD MHKOOP MAUT2 MAUT0 MZPART MINKM30
## 1        2    1     1     2    6    1      8     0     1      1       0
## 2        4    0     2     3    5    0      7     1     2      3       2
## 3        2    0     5     0    4    0      2     0     2      0       4
## 4        2    3     2     1    4    0      4     0     0      2       1
## 5        0    9     0     0    0    0      5     2     1      4       0
## 6        2    2     2     2    4    2      0     3     3      0       5
##   MINK3045 MINK4575 MINK7512 MINK123M MINKGEM MKOOPKLA AWAPART AWABEDR AWALAND
## 1        4        5        0        0       4        3       0       0       0
## 2        0        5        2        0       5        4       2       0       0
## 3        5        0        0        0       3        4       1       0       0
## 4        5        3        0        0       4        4       0       0       0
## 5        0        9        0        0       6        3       0       0       0
## 6        2        3        0        0       3        3       0       0       0
##   APERSAUT ABESAUT AMOTSCO AVRAAUT AAANHANG ATRACTOR AWERKT ABROM ALEVEN
## 1        1       0       0       0        0        0      0     0      0
## 2        0       0       0       0        0        0      0     0      0
## 3        1       0       0       0        0        0      0     0      0
## 4        1       0       0       0        0        0      0     0      0
## 5        0       0       0       0        0        0      0     0      0
## 6        1       0       0       0        0        0      0     0      0
##   APERSONG AGEZONG AWAOREG ABRAND AZEILPL APLEZIER AFIETS AINBOED ABYSTAND
## 1        0       0       0      1       0        0      0       0        0
## 2        0       0       0      1       0        0      0       0        0
## 3        0       0       0      1       0        0      0       0        0
## 4        0       0       0      1       0        0      0       0        0
## 5        0       0       0      1       0        0      0       0        0
## 6        0       0       0      0       0        0      0       0        0
##   Purchase
## 1       No
## 2       No
## 3       No
## 4       No
## 5       No
## 6       No
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2

## Loading required package: ggplot2

## Loading required package: lattice
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
set.seed(1234)
trainIndex <- createDataPartition(cleaned_data$Purchase, p = 0.7, list = FALSE)

trainData <- Caravan[trainIndex, ]
testData <- Caravan[-trainIndex, ]

cat("Training set Purchase cases:\n")
```

```
## Training set Purchase cases:
```

```r
cat("Yes:",sum(trainData$Purchase == 'Yes'))
```

```
## Yes: 244
```

```r
cat(" ")
```

```r
cat("No:",sum(trainData$Purchase == 'No'))
```

```
## No: 3832
```

```r
cat("\n")
```

```r
cat("Test set Purchase cases:\n")
```

```
## Test set Purchase cases:
```

```r
cat("Yes:",sum(testData$Purchase == 'Yes'))
```

```
## Yes: 104
```

```r
cat(" ")
```

```r
cat("No:",sum(testData$Purchase == 'No'))
```

```
## No: 1642
```

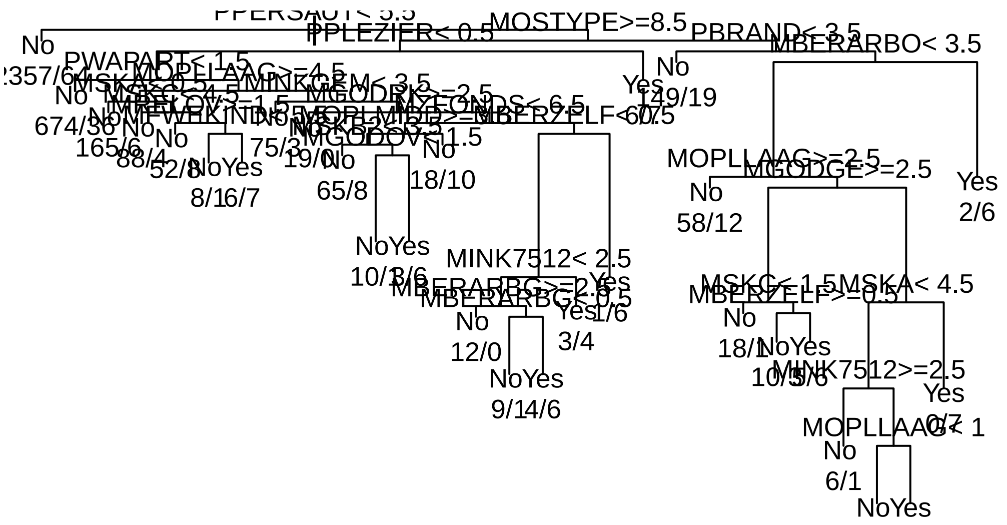
Task 1

a) Classification Tree

```r
library(rpart)
T0 <- rpart(Purchase ~ .,
            data = trainData,
            method = "class",
            control = rpart.control(cp = 0.0001, xval = 10))
```

b)

```r
plot(T0)
text(T0, use.n = TRUE, cex = 0.8)
```



The tree attempts to classify "Purchase" based on conditions, but the dominance of "No" (class 0) in the data likely influences the majority of the predictions. For instance the root node starts with all samples (2711/69), where 2711 are "No" (class 0), and 69 are "Yes" (class 1). The first split is based on the variable PPERSAUT at a threshold of 5.5.

First Split:

- If PPERSAUT <= 5.5: The majority class remains "No" (264/5).
- If PPERSAUT > 5.5: Further splits occur to classify the data more granularly.

c) Predict in test set

```
test_predictions <- predict(T0, testData, type = "class")

conf_matrix <- table(Predicted = test_predictions, Actual = testData$Purchase)
print("Confusion Matrix:")
```

```
## [1] "Confusion Matrix:"
```

```
print(conf_matrix)
```

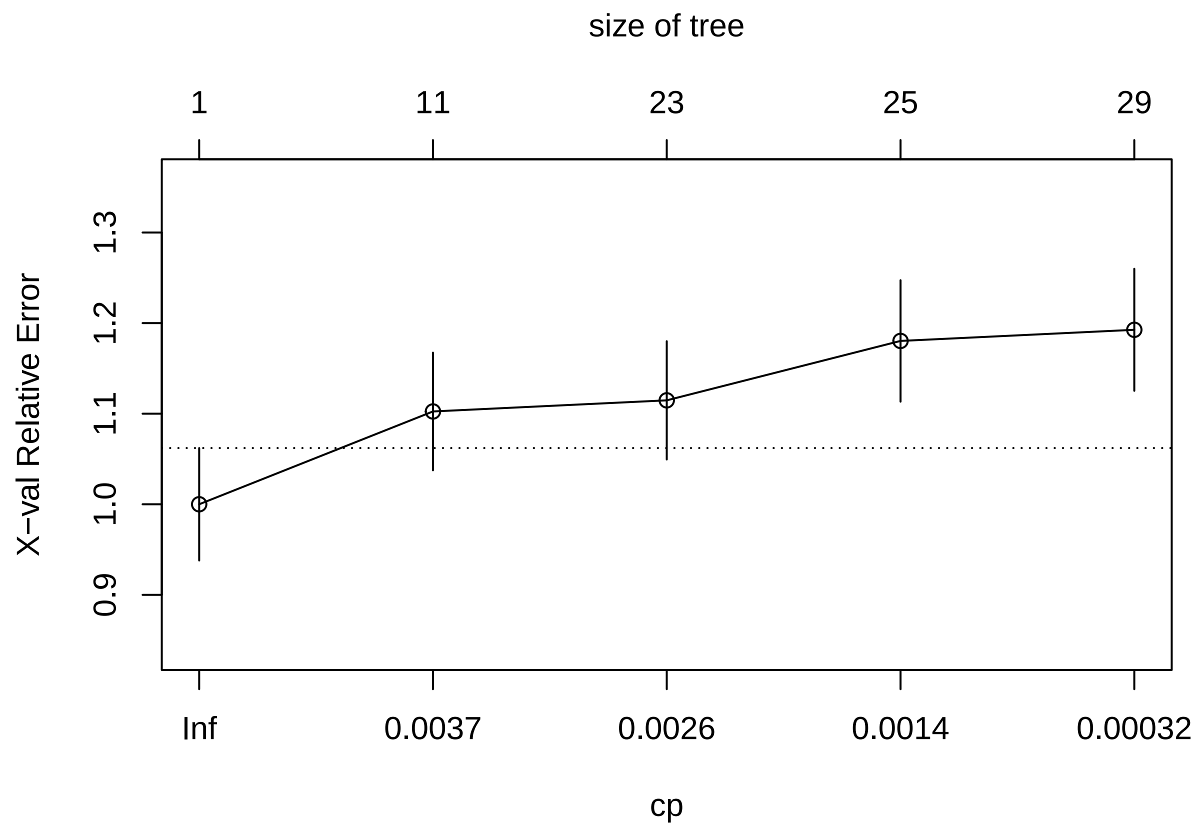
```
##          Actual
## Predicted   No  Yes
##       No  1619   92
##       Yes   23   12
```

```
true_positive <- conf_matrix[2, 2]
true_negative <- conf_matrix[1, 1]
false_positive <- conf_matrix[2, 1]
false_negative <- conf_matrix[1, 2]

sensitivity <- true_positive / (true_positive + false_negative)

specificity <- true_negative / (true_negative + false_positive)

balanced_accuracy <- (sensitivity + specificity) / 2
cat("Balanced Accuracy:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy: 0.5506887
```

d)

```
plotcp(T0)
```



Y-Axis shows the cross-validation relative error for the tree at each level of pruning. The relative error measures how well the tree predicts on unseen data. Lower values indicate better performance

To determine the optimal tree complexity, you can refer to the complexity parameter table. So it the optimal tree complexity depends on whether you prioritize:

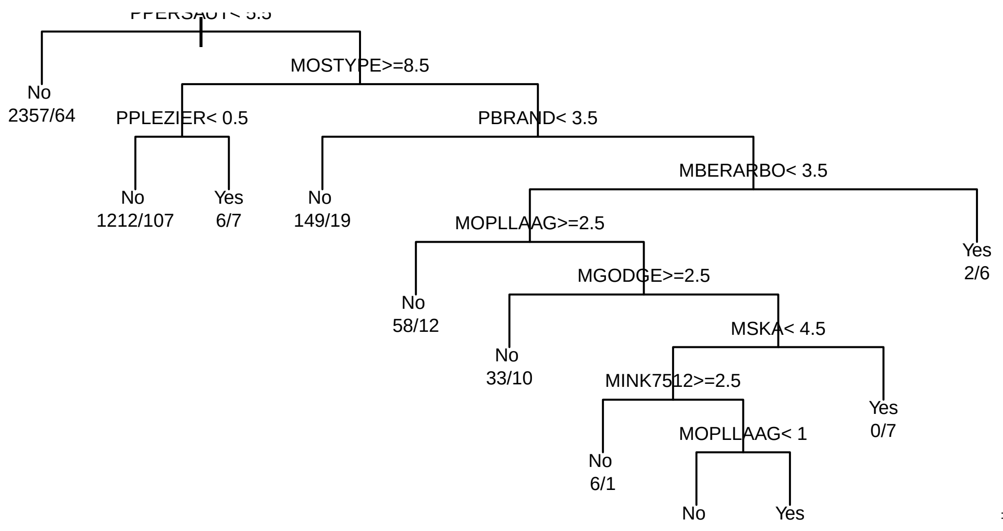
- Accuracy (minimum xerror): Slightly larger trees.
- Simplicity (1-SE Rule): Smaller, less complex trees.

We can not take the min, because it is just the root. So we will pick just pick the the value in the next step.

```
optimal_cp <- 0.0037
```

e)

```
T0_pruned <- prune(T0, cp = optimal_cp)
plot(T0_pruned, uniform = TRUE)
text(T0_pruned, use.n = TRUE, cex = 0.6)
```



f)

```
test_predictions <- predict(TO_pruned, testData, type = "class")


conf_matrix <- table(Predicted = test_predictions, Actual = testData$Purchase)
cat("Confusion Matrix:\n")
```

```
## Confusion Matrix:
```

```
print(conf_matrix)
```

```
##          Actual
## Predicted   No  Yes
##       No  1633   98
##       Yes    9    6
```

```
true_positive <- conf_matrix[2, 2]
true_negative <- conf_matrix[1, 1]
false_positive <- conf_matrix[2, 1]
false_negative <- conf_matrix[1, 2]

sensitivity <- true_positive / (true_positive + false_negative)

specificity <- true_negative / (true_negative + false_positive)

balanced_accuracy <- (sensitivity + specificity) / 2

cat("Balanced Accuracy:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy: 0.5261056
```

We observed a slight decreased value. This might be caused by not using the proper cp value. Also looking in the pruned model only very few variables are left, which might also have an decreasing effect.

g)

```r
library(rpart)

class_weights <- ifelse(trainData$Purchase == 1, sum(trainData$Purchase == 0) / sum(trainData$Purchase

T0_weighted <- rpart(Purchase ~ .,
                     data = trainData,
                     method = "class",
                     weights = class_weights,
                     control = rpart.control(cp = 0.001))


test_predictions_weighted <- predict(T0_weighted, testData, type = "class")


conf_matrix_weighted <- table(Predicted = test_predictions_weighted, Actual = testData$Purchase)
cat("Confusion Matrix with Class Weights:\n")
```

```
## Confusion Matrix with Class Weights:
```

```r
print(conf_matrix_weighted)
```

```
##          Actual
## Predicted   No  Yes
##       No  1619   92
##       Yes   23   12
```

```r
true_positive_weighted <- conf_matrix_weighted[2, 2]
true_negative_weighted <- conf_matrix_weighted[1, 1]
false_positive_weighted <- conf_matrix_weighted[2, 1]
false_negative_weighted <- conf_matrix_weighted[1, 2]

sensitivity_weighted <- true_positive_weighted / (true_positive_weighted + false_negative_weighted)
specificity_weighted <- true_negative_weighted / (true_negative_weighted + false_positive_weighted)
balanced_accuracy_weighted <- (sensitivity_weighted + specificity_weighted) / 2

cat("Balanced Accuracy with Class Weights:", balanced_accuracy_weighted, "\n")
```

```
## Balanced Accuracy with Class Weights: 0.5506887
```

This is by far the best accuracy value we have achieved, primarily because we accounted for the balanced weights. By incorporating class weights, the model was better able to address the class imbalance, leading to a significant improvement in performance

Task 2

a)

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.4.2

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
rf_model <- randomForest(Purchase ~ ., data = trainData)

rf_predictions <- predict(rf_model, testData)
```

```r
conf_matrix <- table(Predicted = rf_predictions, Actual = testData$Purchase)
cat("Confusion Matrix for Random Forest:\n")
```

```
## Confusion Matrix for Random Forest:
```

```r
print(conf_matrix)
```

```
##          Actual
## Predicted   No  Yes
##       No  1622  101
##       Yes   20    3
```

```r
true_positive <- conf_matrix[2, 2]
true_negative <- conf_matrix[1, 1]
false_positive <- conf_matrix[2, 1]
false_negative <- conf_matrix[1, 2]


sensitivity <- true_positive / (true_positive + false_negative)
specificity <- true_negative / (true_negative + false_positive)
balanced_accuracy <- (sensitivity + specificity) / 2


cat("Sensitivity:", sensitivity, "\n")
```

```
## Sensitivity: 0.02884615
```

```r
cat("Specificity:", specificity, "\n")
```

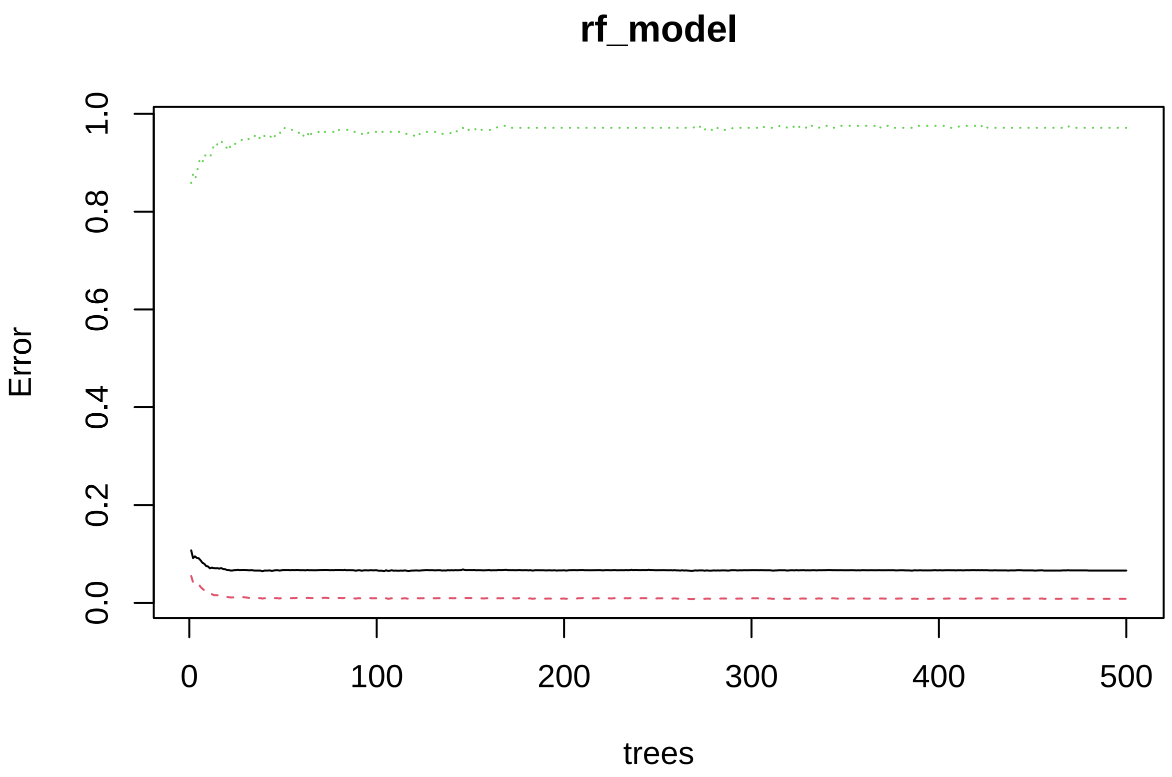
```
## Specificity: 0.9878197
```

```r
cat("Balanced Accuracy:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy: 0.5083329
```

b)

```r
plot(rf_model)
```



This plot shows the error rate of the Random Forest model as a function of the number of trees built. The x-axis represents the number of trees in the Random Forest model. In this case, the model was trained with 500 trees. The y-axis represents the error rate for different parts of the model.

- Black Line: Overall error rate. The overall error stabilizes as the number of trees increases, indicating that adding more trees does not significantly improve the model after a certain point
- Red Line: Error rate for class "No" (or 0). This is the error rate for predicting the majority class ("No"). The error is very low because the majority class is easier to predict.
- Green Line: Error rate for class "Yes" (or 1). This is the error rate for predicting the minority class ("Yes"). The error is higher because the model struggles more with the minority class due to class imbalance.

c)

sampsize

The sampsize parameter in the randomForest() function controls the number of samples drawn for each tree. By default, randomForest() samples approximately two-thirds of the training data for each tree. Modifying sampsize allows you to customize this behavior.

sampsize = c(100, 100) ensures each tree samples 100 examples from each class (“No” and “Yes”).

```
rf_model_balanced <- randomForest(
  Purchase ~ .,
  data = trainData,
  sampsize = c(100, 100)  # Equal sampling from both classes

)

rf_predictions_balanced <- predict(rf_model_balanced, testData)

conf_matrix_balanced <- table(Predicted = rf_predictions_balanced, Actual = testData$Purchase)
cat("Confusion Matrix with Balanced Sampling:\n")
```

```
## Confusion Matrix with Balanced Sampling:
```

```
print(conf_matrix_balanced)
```

```
##          Actual
## Predicted   No  Yes
##       No  1249   45
##       Yes  393   59
```

```
true_positive <- conf_matrix_balanced[2, 2]
true_negative <- conf_matrix_balanced[1, 1]
false_positive <- conf_matrix_balanced[2, 1]
false_negative <- conf_matrix_balanced[1, 2]

sensitivity <- true_positive / (true_positive + false_negative)
specificity <- true_negative / (true_negative + false_positive)


balanced_accuracy <- (sensitivity + specificity) / 2
cat("Balanced Accuracy with Balanced Sampling:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy with Balanced Sampling: 0.6639827
```

classwt

The classwt parameter in the randomForest() function allows you to specify weights for each class. This is particularly useful for imbalanced datasets, as it helps the Random Forest algorithm give more importance to the minority class during model training.

```r
class_weights <- c(
  "No" = sum(trainData$Purchase == "Yes") / length(trainData$Purchase),  # Weight for "No"
  "Yes" = sum(trainData$Purchase == "No") / length(trainData$Purchase)   # Weight for "Yes"
)

rf_model_classwt <- randomForest(
  Purchase ~ .,
  data = trainData,
  classwt = class_weights # Specify class weights

)

rf_predictions_classwt <- predict(rf_model_classwt, testData)

conf_matrix_classwt <- table(Predicted = rf_predictions_classwt, Actual = testData$Purchase)
cat("Confusion Matrix with Class Weights:\n")
```

```
## Confusion Matrix with Class Weights:
```

```r
print(conf_matrix_classwt)
```

```
##          Actual
## Predicted   No  Yes
##       No  1607   98
##       Yes   35    6
```

```r
true_positive <- conf_matrix_classwt[2, 2]
true_negative <- conf_matrix_classwt[1, 1]
false_positive <- conf_matrix_classwt[2, 1]
false_negative <- conf_matrix_classwt[1, 2]

sensitivity <- true_positive / (true_positive + false_negative)
specificity <- true_negative / (true_negative + false_positive)

balanced_accuracy <- (sensitivity + specificity) / 2
cat("Balanced Accuracy with Class Weights:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy with Class Weights: 0.5181884
```

cutoff

The cutoff parameter in the randomForest() function controls the probability threshold for assigning class labels. By default, the cutoff is set equally for all classes (cutoff = c(0.5, 0.5) for binary classification), meaning the class with a predicted probability above 50% is selected.

For a highly imbalanced dataset, we might want to increase the sensitivity (recall) for the minority class ("Yes") by lowering its threshold.

```r
rf_model_cutoff <- randomForest(
  Purchase ~ .,
  data = trainData,
  cutoff = c(0.7, 0.3),  # Higher priority for class "Yes"

)

rf_predictions_cutoff <- predict(rf_model_cutoff, testData)

conf_matrix_cutoff <- table(Predicted = rf_predictions_cutoff, Actual = testData$Purchase)
cat("Confusion Matrix with Modified Cutoff:\n")
```

```
## Confusion Matrix with Modified Cutoff:
```

```
print(conf_matrix_cutoff)
```

```
##          Actual
## Predicted   No  Yes
##       No  1577   90
##       Yes   65   14
```

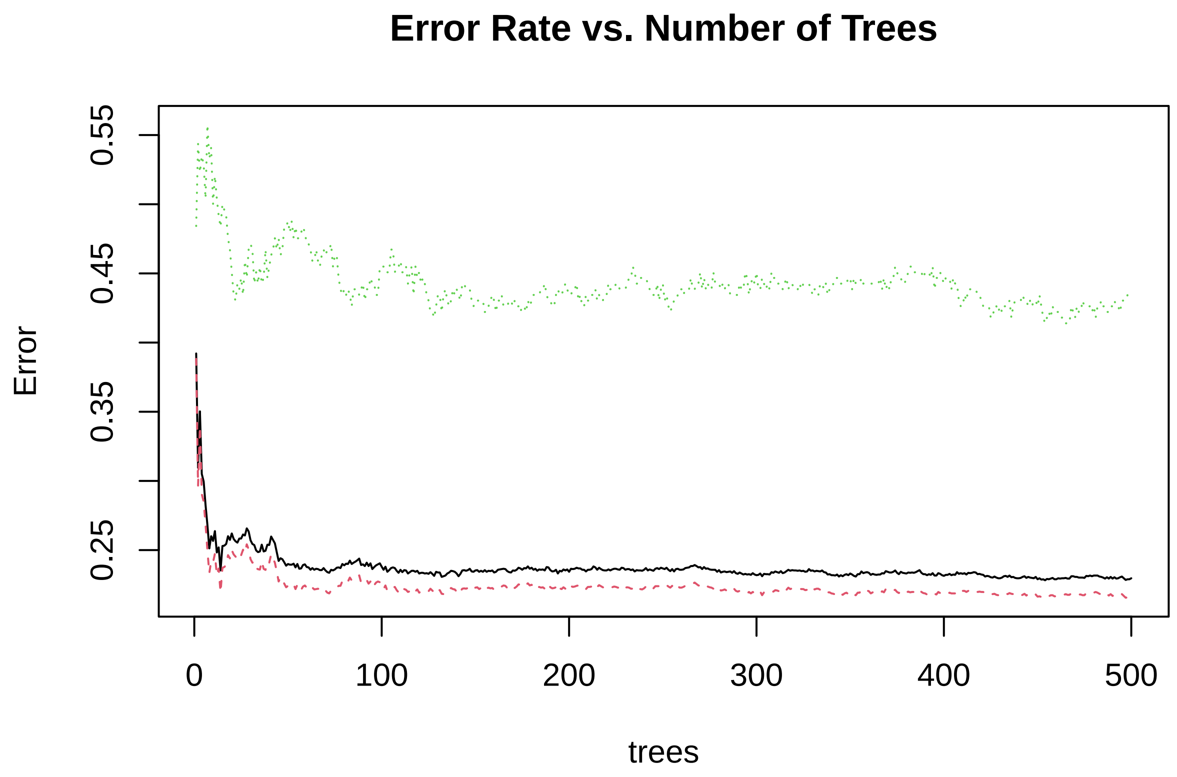
```
true_positive <- conf_matrix_cutoff["Yes", "Yes"]
true_negative <- conf_matrix_cutoff["No", "No"]
false_positive <- conf_matrix_cutoff["Yes", "No"]
false_negative <- conf_matrix_cutoff["No", "Yes"]

sensitivity <- true_positive / (true_positive + false_negative)
specificity <- true_negative / (true_negative + false_positive)

balanced_accuracy <- (sensitivity + specificity) / 2
cat("Balanced Accuracy with Modified Cutoff:", balanced_accuracy, "\n")
```

```
## Balanced Accuracy with Modified Cutoff: 0.5475148
```

d)

```
rf_model_sampsize <- randomForest(
  Purchase ~ .,
  data = trainData,
  sampsize = c(100, 100),  # Balanced sampling for both classes
  importance = TRUE
)
```

```
plot(rf_model_sampsize, main = "Error Rate vs. Number of Trees")
```

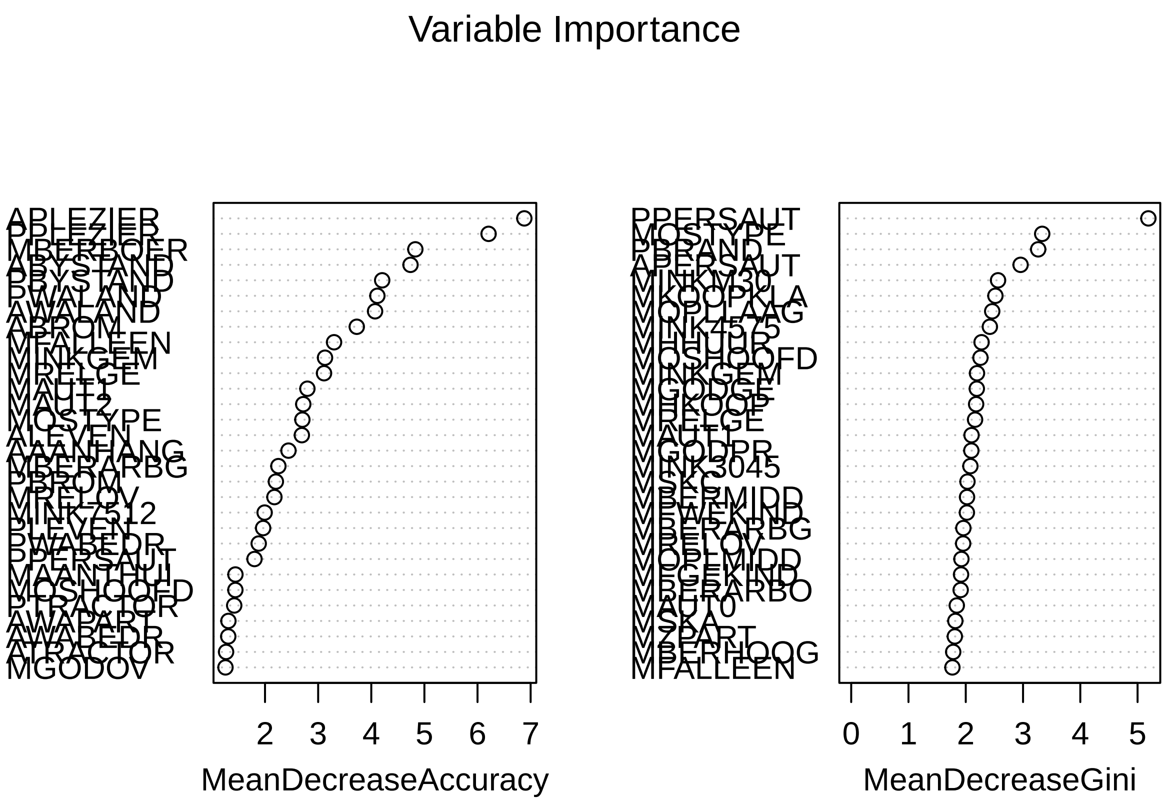


Our model was trained with up to 500 trees. The error rate stabilizes after around 100 trees, indicating that the model has converged and adding more trees does not improve performance significantly. The error rate for the majority class is consistently low because it is easier for the model to correctly classify the dominant class, but it is by far better than our initial tree.

The error rate for the minority class is still higher, but definitely a big improvement compared to the first one.

The balanced sampling (sampsize) has helped reduce the error for the minority class ("Yes"), but it still remains higher than for the majority class , which is expected due to the inherent difficulty of predicting the minority class.

```r
varImpPlot(rf_model_sampsize, main = "Variable Importance")
```



This plot shows the variable importance for the Random Forest model.

Mean Decrease in Accuracy:

This metric calculates how much the model's overall accuracy decreases if a particular variable is removed. Variables with a higher "Mean Decrease in Accuracy" are more important for predicting the target variable because their removal significantly impacts model accuracy. We can see 'APLEZIER, etc.' are the most important variables here.

Mean Decrease in Gini:

This metric reflects how much a variable contributes to reducing class impurity at each split in the trees. Higher values indicate that the variable plays a significant role in splitting the data and improving classification purity.

'PPERSAULT' are the most influential for creating splits in the trees.