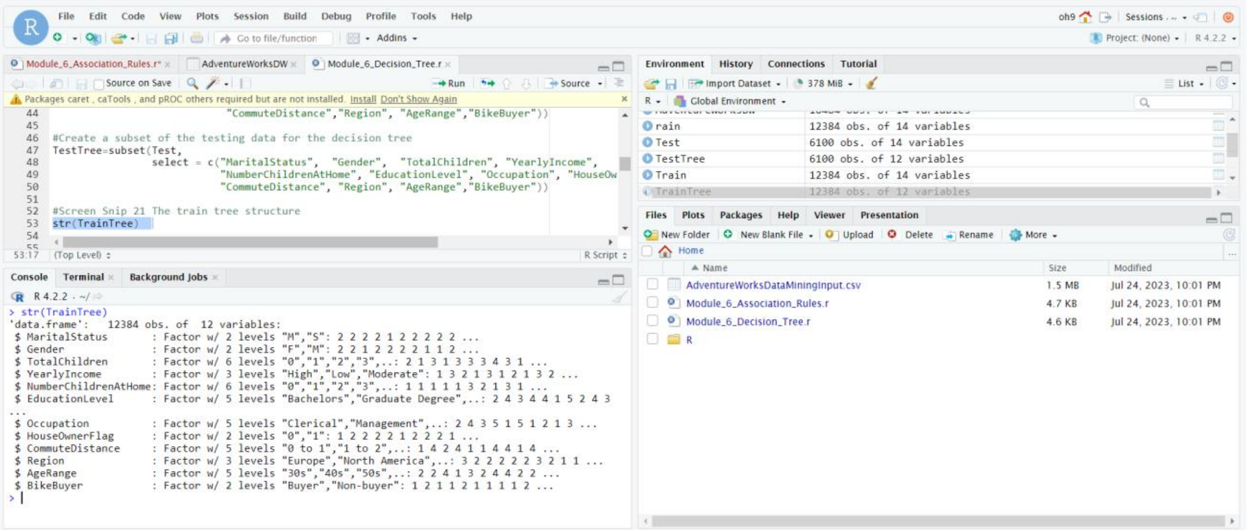
DECISION TREE AND RANDOM FOREST FOR THE ADVENTUREWORKDW DATASET

Name: Oscar Hernandez Mata

In this project I wanted to predict whether a costumer would be classified as a potential bike buyer or a no-bike buyer. Hence, I used the AdventureWorkDW dataset from MySQL, which I modified by selecting the necessary columns for this project.

We converted some of the integer variables to factors in order to create a training set that contains 67% of the data and a training set that contains the rest 33% by using the hold out method.

Next, we created a subset of the training data (TrainTree) to limit the number of columns in the decision tree. Then we repeated the process for the testing data (See Figure 1).



Packages caret , caTools , and pROC others required but are not installed. Install Don't Show Again

```
                                   "CommuteDistance","Region", "AgeRange","BikeBuyer"))

#Create a subset of the testing data for the decision tree
TestTree=subset(Test,
                select = c("MaritalStatus",  "Gender",  "TotalChildren", "YearlyIncome",
                           "NumberChildrenAtHome", "EducationLevel", "Occupation", "HouseOw
                           "CommuteDistance", "Region", "AgeRange","BikeBuyer"))

#Screen Snip 21 The train tree structure
str(TrainTree)
```

```
> str(TrainTree)
'data.frame':	12384 obs. of  12 variables:
 $ MaritalStatus       : Factor w/ 2 levels "M","S": 2 2 2 2 1 2 2 2 2 2 ...
 $ Gender              : Factor w/ 2 levels "F","M": 2 2 1 2 2 2 2 1 1 2 ...
 $ TotalChildren       : Factor w/ 6 levels "0","1","2","3",..: 2 1 3 1 3 3 3 4 3 1 ...
 $ YearlyIncome        : Factor w/ 3 levels "High","Low","Moderate": 1 3 2 1 3 1 2 1 3 2 ...
 $ NumberChildrenAtHome: Factor w/ 6 levels "0","1","2","3",..: 1 1 1 1 1 3 2 1 3 1 ...
 $ EducationLevel      : Factor w/ 5 levels "Bachelors","Graduate Degree",..: 2 4 3 4 4 1 5 2 4 3
...
 $ Occupation          : Factor w/ 5 levels "Clerical","Management",..: 2 4 3 5 1 5 1 2 1 3 ...
 $ HouseOwnerFlag      : Factor w/ 2 levels "0","1": 1 2 2 2 2 1 2 2 2 1 ...
 $ CommuteDistance     : Factor w/ 5 levels "0 to 1","1 to 2",..: 1 4 2 4 1 1 4 4 1 4 ...
 $ Region              : Factor w/ 3 levels "Europe","North America",..: 3 2 2 2 2 2 3 2 1 1 ...
 $ AgeRange            : Factor w/ 5 levels "30s","40s","50s",..: 2 2 4 1 3 2 4 4 2 2 ...
 $ BikeBuyer           : Factor w/ 2 levels "Buyer","Non-buyer": 1 2 1 1 2 1 1 1 1 2 ...
> |
```

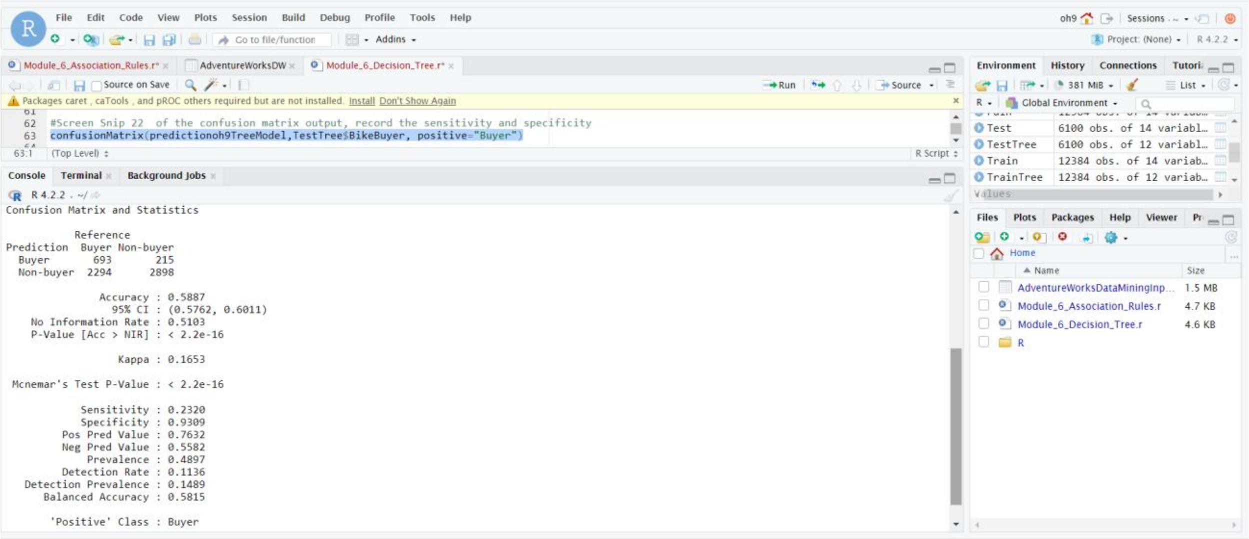
rain	12384 obs. of 14 variables
Test	6100 obs. of 14 variables
TestTree	6100 obs. of 12 variables
Train	12384 obs. of 14 variables
TrainTree	12384 obs. of 12 variables

Name	Size	Modified
AdventureWorksDataMiningInput.csv	1.5 MB	Jul 24, 2023, 10:01 PM
Module_6_Association_Rules.r	4.7 KB	Jul 24, 2023, 10:01 PM
Module_6_Decision_Tree.r	4.6 KB	Jul 24, 2023, 10:01 PM
R		

Figure 1

The subsequent phase in the classification procedure involved constructing the decision tree model using the training subset, with "Bike Buyer" designated as the class label. Following this, the model was evaluated by presenting it with the testing data.

To assess the effectiveness of our model, a confusion matrix was generated. This matrix provides a detailed breakdown of the true positives, true negatives, false positives, and false negatives. Additionally, measures of sensitivity and specificity were calculated. The confusion matrix is crucial as it offers insights into the robustness and accuracy of our model, as illustrated in Figure 2.



```
#Screen Snip 22  of the confusion matrix output, record the sensitivity and specificity
confusionMatrix(predictionoh9TreeModel,TestTree$BikeBuyer, positive="Buyer")
```

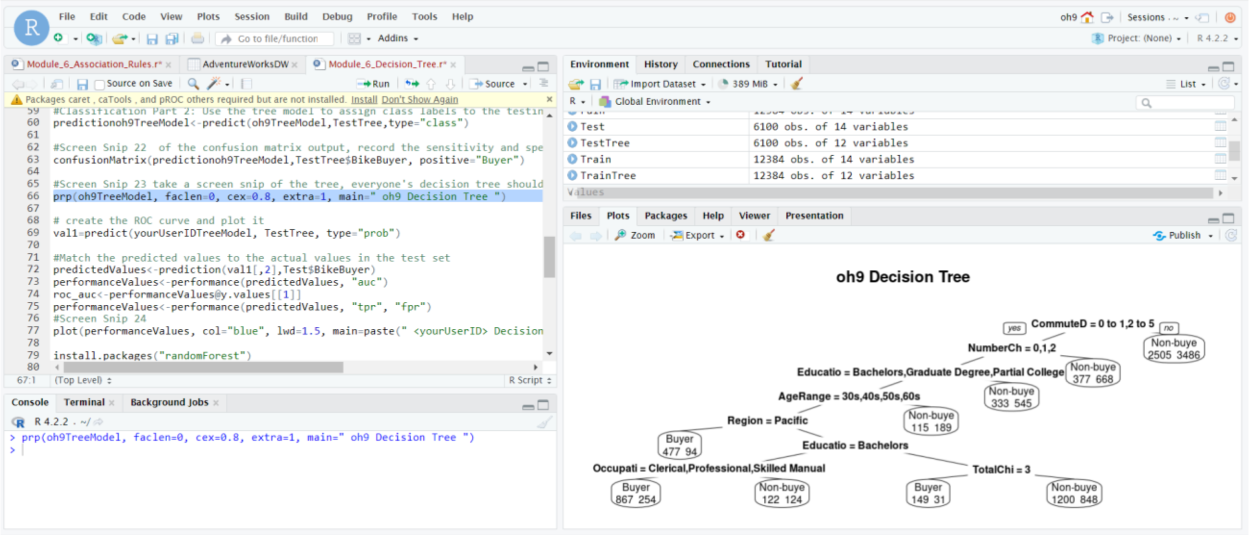
```
Confusion Matrix and Statistics

           Reference
Prediction  Buyer Non-buyer
  Buyer       693       215
  Non-buyer  2294      2898

               Accuracy : 0.5887
                 95% CI : (0.5762, 0.6011)
    No Information Rate : 0.5103
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.1653

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.2320
            Specificity : 0.9309
         Pos Pred Value : 0.7632
         Neg Pred Value : 0.5582
             Prevalence : 0.4897
         Detection Rate : 0.1136
   Detection Prevalence : 0.1489
      Balanced Accuracy : 0.5815

       'Positive' Class : Buyer
```

Test	6100 obs. of 14 variabl…
TestTree	6100 obs. of 12 variabl…
Train	12384 obs. of 14 variab…
TrainTree	12384 obs. of 12 variab…

Name	Size
AdventureWorksDataMiningInp…	1.5 MB
Module_6_Association_Rules.r	4.7 KB
Module_6_Decision_Tree.r	4.6 KB
R	

Figure 2

The analysis indicated that the decision tree model demonstrated a sensitivity of 0.23 and a specificity of 0.93. These results suggest that while the model is highly effective at identifying true negatives, it struggles considerably with accurately predicting true positives, as detailed in Figures 2 and 3.



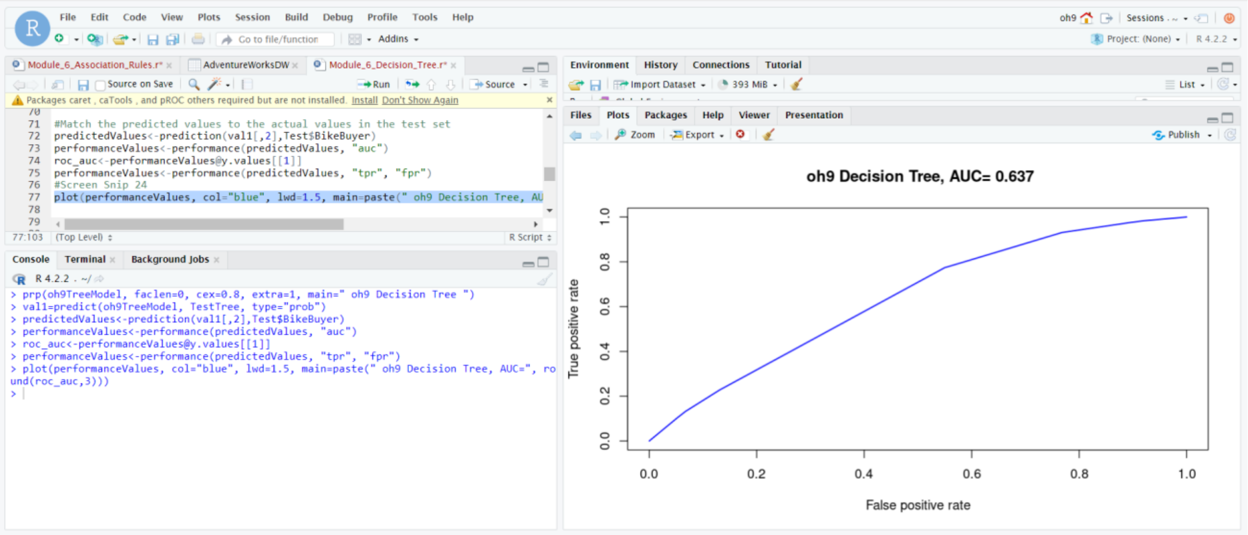
```
#Classification Part 2: Use the tree model to assign class labels to the testin
predictionoh9TreeModel<-predict(oh9TreeModel,TestTree,type="class")

#Screen Snip 22  of the confusion matrix output, record the sensitivity and spe
confusionMatrix(predictionoh9TreeModel,TestTree$BikeBuyer, positive="Buyer")

#Screen Snip 23 take a screen snip of the tree, everyone's decision tree should
prp(oh9TreeModel, faclen=0, cex=0.8, extra=1, main=" oh9 Decision Tree ")

# create the ROC curve and plot it
val1=predict(yourUserIDTreeModel, TestTree, type="prob")

#Match the predicted values to the actual values in the test set
predictedValues<-prediction(val1[,2],Test$BikeBuyer)
performanceValues<-performance(predictedValues, "auc")
roc_auc<-performanceValues@y.values[[1]]
performanceValues<-performance(predictedValues, "tpr", "fpr")
#Screen Snip 24
plot(performanceValues, col="blue", lwd=1.5, main=paste(" <yourUserID> Decision

install.packages("randomForest")
```

```
> prp(oh9TreeModel, faclen=0, cex=0.8, extra=1, main=" oh9 Decision Tree ")
> 
```

Test	6100 obs. of 14 variables
TestTree	6100 obs. of 12 variables
Train	12384 obs. of 14 variables
TrainTree	12384 obs. of 12 variables

Figure 3

The ROC curve graphic provides an assessment of the model's predictive accuracy by comparing predicted values against actual outcomes. The AUC, or Area Under the Curve, for the decision tree model is 0.637, as shown in Figure 4. This value indicates that the model's performance in distinguishing between positive and negative instances is suboptimal, given that the AUC is significantly lower than the ideal value of 1.



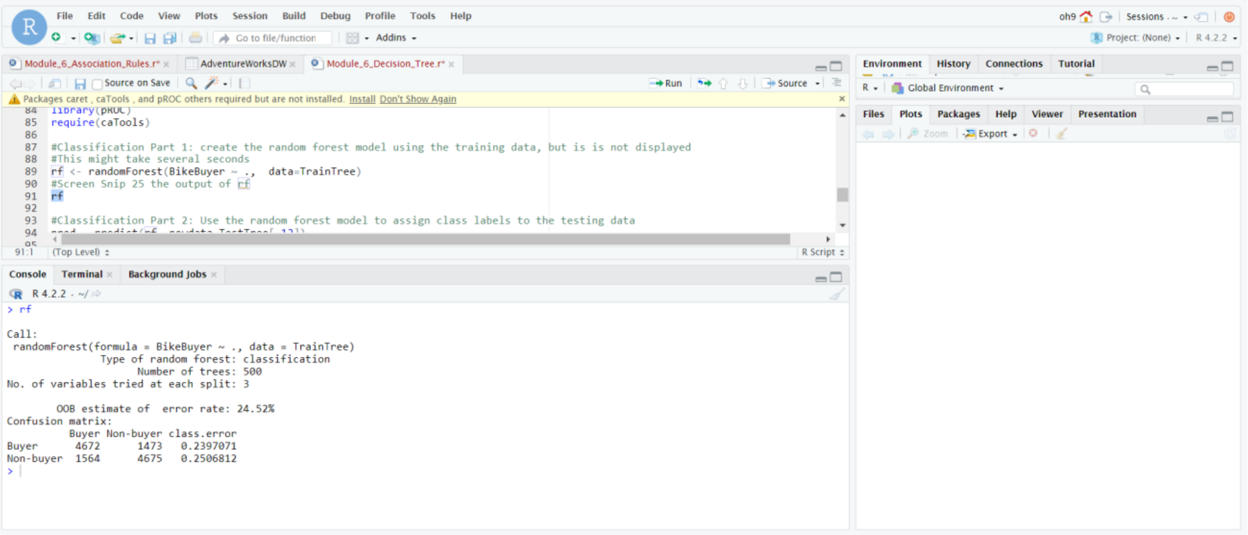
⚠ Packages caret , caTools , and pROC others required but are not installed. Install Don't Show Again

```
#Match the predicted values to the actual values in the test set
predictedValues<-prediction(val1[,2],Test$BikeBuyer)
performanceValues<-performance(predictedValues, "auc")
roc_auc<-performanceValues@y.values[[1]]
performanceValues<-performance(predictedValues, "tpr", "fpr")
#Screen Snip 24
plot(performanceValues, col="blue", lwd=1.5, main=paste(" oh9 Decision Tree, AU
```

```
> prp(oh9TreeModel, faclen=0, cex=0.8, extra=1, main=" oh9 Decision Tree ")
> val1=predict(oh9TreeModel, TestTree, type="prob")
> predictedValues<-prediction(val1[,2],Test$BikeBuyer)
> performanceValues<-performance(predictedValues, "auc")
> roc_auc<-performanceValues@y.values[[1]]
> performanceValues<-performance(predictedValues, "tpr", "fpr")
> plot(performanceValues, col="blue", lwd=1.5, main=paste(" oh9 Decision Tree, AUC=", ro
und(roc_auc,3)))
> 
```

Figure 4

Since the decision tree did not accurately predict the true positives, I first installed the randomForest and caTools packages, then created a random forest model using the randomForest command for the training data (See Figure 5). Next, I used the random forest model to assign class labels to the testing data: **predict(rf, newdata=TestTree[-12]) and able(TestTree[,12], pred)**.



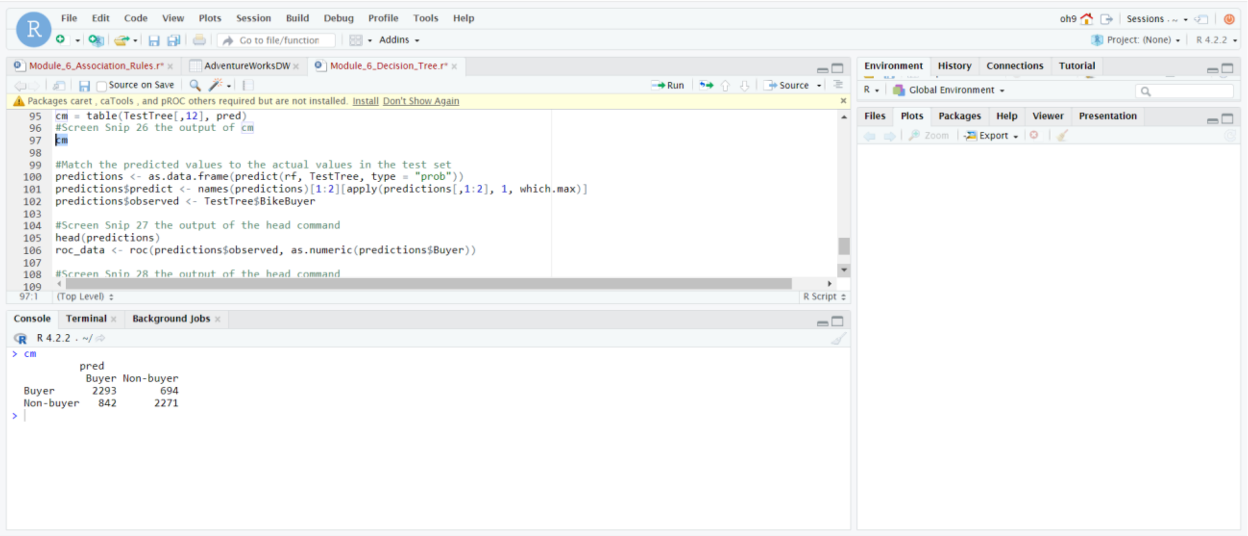
⚠ Packages caret , caTools , and pROC others required but are not installed. Install Don't Show Again

```
library(pROC)
require(caTools)

#Classification Part 1: create the random forest model using the training data, but is is not displayed
#This might take several seconds
rf <- randomForest(BikeBuyer ~ .,  data=TrainTree)
#Screen Snip 25 the output of rf
rf

#Classification Part 2: Use the random forest model to assign class labels to the testing data
```

```
> rf

Call:
 randomForest(formula = BikeBuyer ~ ., data = TrainTree)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 24.52%
Confusion matrix:
          Buyer Non-buyer class.error
Buyer      4672      1473   0.2397071
Non-buyer  1564      4675   0.2506812
> 
```

Figure 5

Finally I matched the predicted values to the actual values in the test set using the follwing commands: **predictions <- as.data.frame(predict(rf, TestTree, type = "prob")), predictions$predict <- names(predictions)[1:2][apply(predictions[,1:2], 1, which.max)], predictions$observed <- TestTree$BikeBuyer** (See figure 6).



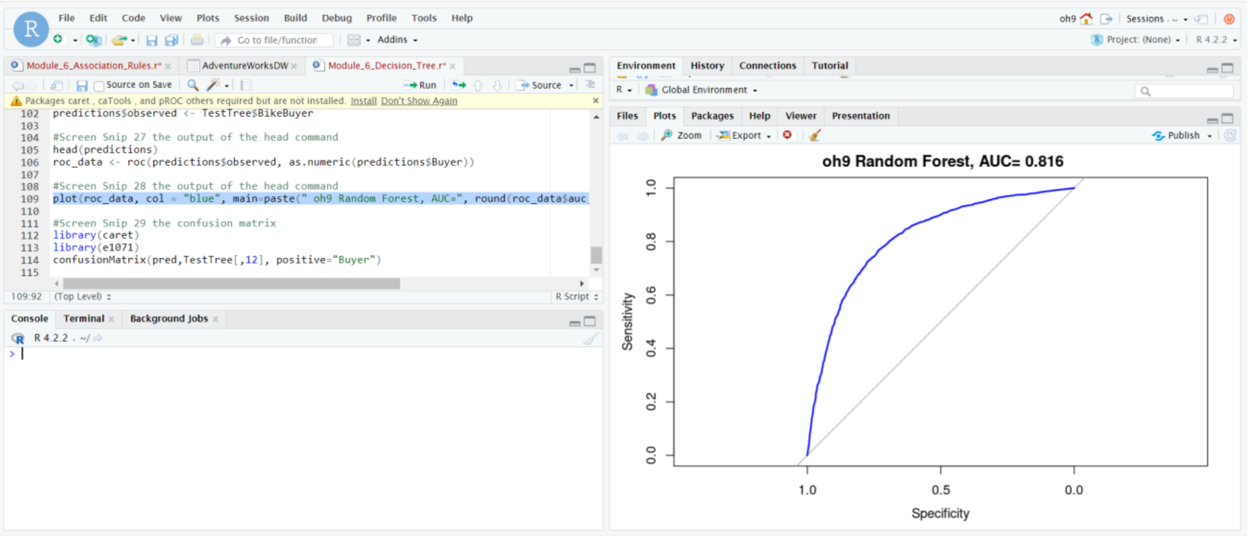
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
oh9
Sessions
Project: (None)
R 4.2.2
Module_6_Association_Rules.r*
AdventureWorksDW
Module_6_Decision_Tree.r*
Source on Save
Run
Source
Environment History Connections Tutorial
Global Environment
Files Plots Packages Help Viewer Presentation
Zoom
Export
97:1
(Top Level)
R Script
Console Terminal Background Jobs
R 4.2.2 . ~/

Packages caret , caTools , and pROC others required but are not installed. Install Don't Show Again

```r
cm = table(TestTree[,12], pred)
#Screen Snip 26 the output of cm
cm

#Match the predicted values to the actual values in the test set
predictions <- as.data.frame(predict(rf, TestTree, type = "prob"))
predictions$predict <- names(predictions)[1:2][apply(predictions[,1:2], 1, which.max)]
predictions$observed <- TestTree$BikeBuyer

#Screen Snip 27 the output of the head command
head(predictions)
roc_data <- roc(predictions$observed, as.numeric(predictions$Buyer))

#Screen Snip 28 the output of the head command
```

```
> cm
          pred
           Buyer Non-buyer
  Buyer     2293       694
  Non-buyer  842      2271
>
```

Figure 6



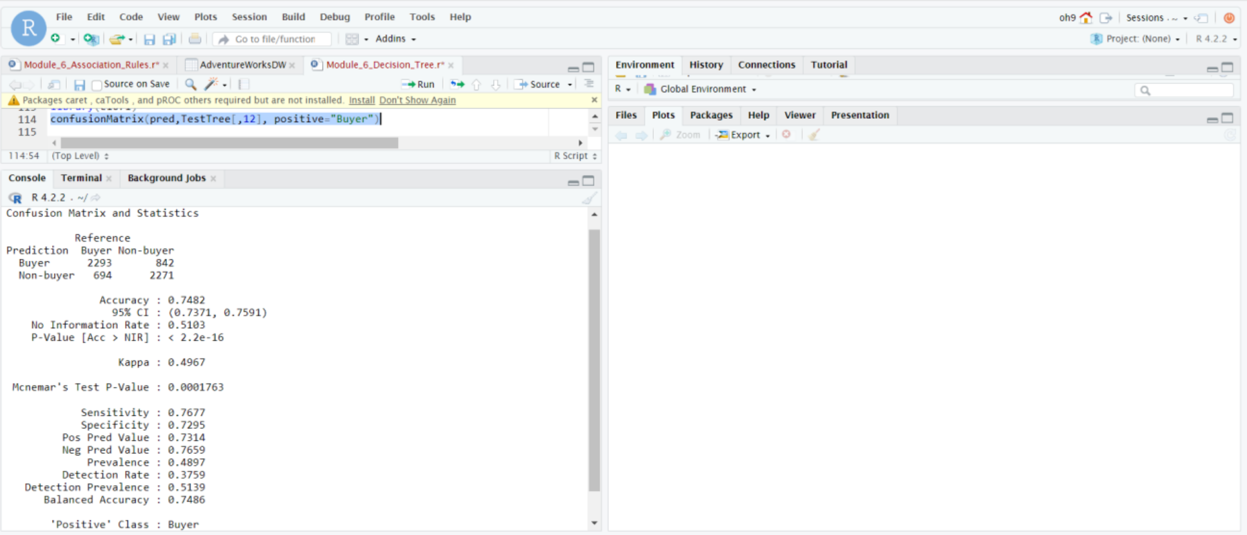
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
oh9
Sessions
Project: (None)
R 4.2.2
Module_6_Association_Rules.r*
AdventureWorksDW
Module_6_Decision_Tree.r*
Source on Save
Run
Source
Environment History Connections Tutorial
Global Environment
Files Plots Packages Help Viewer Presentation
Zoom
Export
Publish
109:92
(Top Level)
R Script
Console Terminal Background Jobs
R 4.2.2 . ~/
oh9 Random Forest, AUC= 0.816
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
Specificity

Packages caret , caTools , and pROC others required but are not installed. Install Don't Show Again

```r
predictions$observed <- TestTree$BikeBuyer

#Screen Snip 27 the output of the head command
head(predictions)
roc_data <- roc(predictions$observed, as.numeric(predictions$Buyer))

#Screen Snip 28 the output of the head command
plot(roc_data, col = "blue", main=paste(" oh9 Random Forest, AUC=", round(roc_data$auc

#Screen Snip 29 the confusion matrix
library(caret)
library(e1071)
confusionMatrix(pred,TestTree[,12], positive="Buyer")

```

```
>
```

Figure 7



Packages caret , caTools , and pROC others required but are not installed. Install Don't Show Again

```
confusionMatrix(pred,TestTree[,12], positive="Buyer")
```

```
Confusion Matrix and Statistics

           Reference
Prediction  Buyer Non-buyer
  Buyer      2293       842
  Non-buyer   694      2271

               Accuracy : 0.7482
                 95% CI : (0.7371, 0.7591)
    No Information Rate : 0.5103
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4967

 Mcnemar's Test P-Value : 0.0001763

            Sensitivity : 0.7677
            Specificity : 0.7295
         Pos Pred Value : 0.7314
         Neg Pred Value : 0.7659
             Prevalence : 0.4897
         Detection Rate : 0.3759
   Detection Prevalence : 0.5139
      Balanced Accuracy : 0.7486

       'Positive' Class : Buyer
```

Figure 8

In the recent analysis, the performance of the random forest model was evaluated using the confusionMatrix function, yielding promising results. The model demonstrated a sensitivity of 0.76 and a specificity of 0.72, indicating a robust ability to accurately identify both true positive and true negative outcomes, as depicted in Figure 9. Further, the area under the Receiver Operating Characteristic (ROC) curve, or AUC, was calculated at 0.816. This value, illustrated in Graphic 7, underscores the model's effectiveness in distinguishing between positive and negative instances, approaching the optimal AUC value of 1.