|  |  |  |
| --- | --- | --- |
| A picture containing text, sign, clipart  Description automatically generated | **BOSTON**  **UNIVERSITY** | **METROPOLITAN COLLEGE** |

**AD 699 DATA MINING FOR BUSINESS ANALYTICS**

**ASSIGNMENT 4**

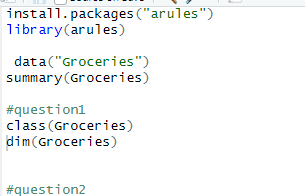
**APRIL 14, 2023**

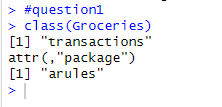
**Aravind Hanumantha Rao**

**BU ID - U55859882**

1. **Describe “Groceries” by answering following questions:**

**1) What is the class of “Groceries”?**

****

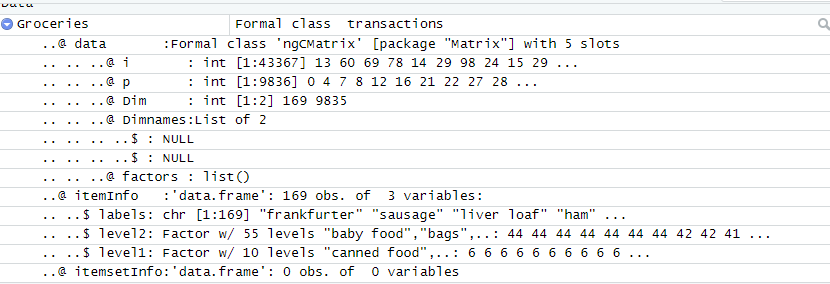
****

Class of groceries as seen from the code is a series of transactions

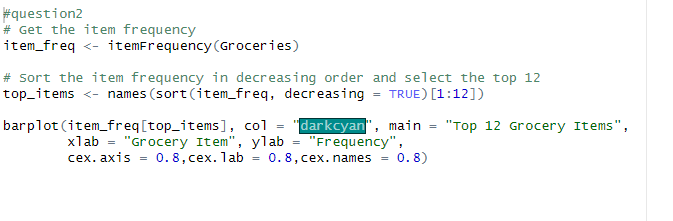
**2)How many rows and columns does Groceries contain?**

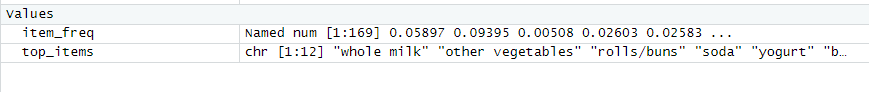
****

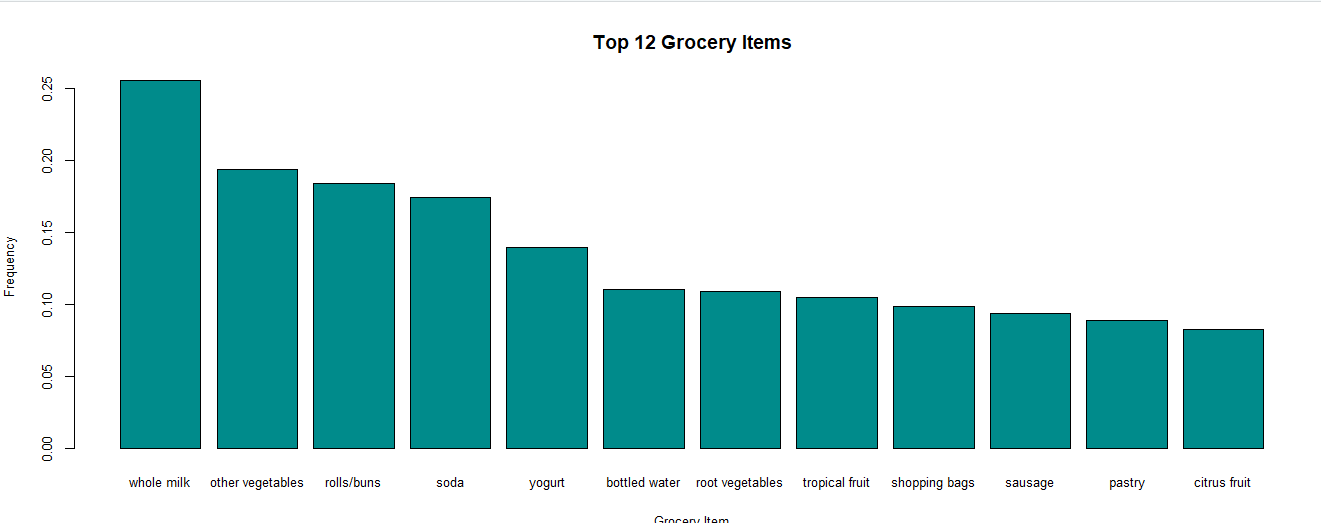
The Groceries object has dimensions of 9835 rows and 169 columns. This suggests that there are 9835 transactions in the dataset, and 169 different items that could have been purchased in those transactions.

****

1. **Generate an item frequency barplot for the grocery items that depicts the 12 most common grocery items from the dataset. Include a screenshot of your results, along with the code you used to do this. Fill the bars with any color of your choice. This plot should be oriented vertically (the default way)?**

****

****

****

1. **Now, create a subset of rules that contain your grocery item (you can find your item in the spreadsheet in Blackboard). Select any one rule with your item on the left-hand side, and any one rule with your item on the right-hand side, and explain them in the way you would explain them to your roommate (I’m assuming your roommate is a smart person who is unfamiliar with data mining). Remember, every rule has four components: support, coverage, confidence, and lift?**

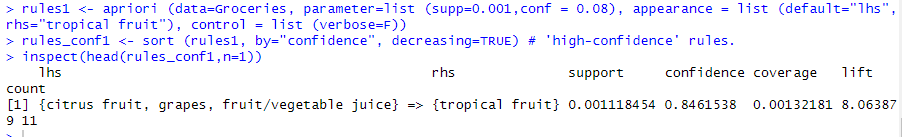
**RHS-Tropical fruit**

**rules1 <- apriori (data=Groceries, parameter=list (supp=0.001,conf = 0.08), appearance = list (default="lhs",rhs="tropical fruit"), control = list (verbose=F))**

**rules\_conf1 <- sort (rules1, by="confidence", decreasing=TRUE) # 'high-confidence' rules.**

**inspect(head(rules\_conf1,n=1))**

****

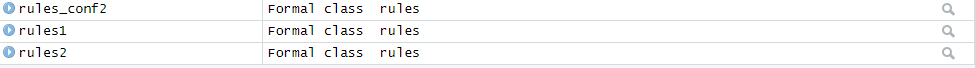
****

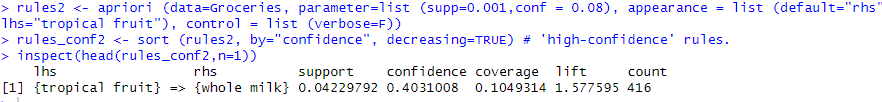
**LHS- Tropical fruit**

**rules2 <- apriori (data=Groceries, parameter=list (supp=0.001,conf = 0.08), appearance = list (default="rhs",lhs="tropical fruit"), control = list (verbose=F))**

**rules\_conf2 <- sort (rules2, by="confidence", decreasing=TRUE) # 'high-confidence' rules.**

**inspect(head(rules\_conf2,n=1))**

****

****

**Explanation in a way to a roommate -**

**Explanation for RHS –**

Overall, this rule suggests that customers who buy citrus fruit, grapes, and fruit/vegetable juice are likely to also buy tropical fruit, with a high level of confidence. This information can be used by retailers to optimize their product placements and promotions, or to suggest additional items to customers during checkout.

**Explanation for LHS –**

this rule suggests that customers who buy tropical fruit are more likely to also buy whole milk, compared to customers who do not buy tropical fruit. If you see someone buying tropical fruit in the store, there's a good chance they might also be interested in buying whole milk.

**4)In a sentence or two, explain what meaning these rules might have for a store like Star Market. What could it do with this information?**

The association rules generated from market basket analysis can provide valuable insights for stores like Star Market regarding which products are often purchased together. This information can be used to optimize product placements, design targeted promotions, and suggest additional items to customers during checkout. For example, if Star Market observes that customers who buy tropical fruits are also likely to buy whole milk, they could place these items in close proximity or run promotions that offer discounts for purchasing both items together to encourage customers to buy them.

**5)Using the plot() function in the arulesViz package, generate a scatter plot of any three rules involving your grocery item. Include a screenshot of your plot, along with the code you used to generate the plot. Describe your results in a sentence or two.**

**install.packages("arulesViz")**

**library(arulesViz)**

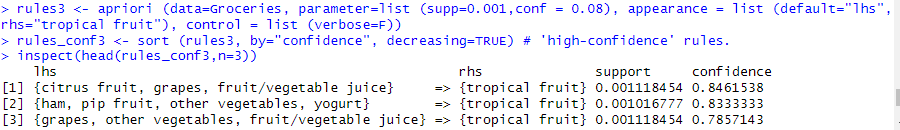
**rules3 <- apriori (data=Groceries, parameter=list (supp=0.001,conf = 0.08), appearance = list (default="lhs",rhs="tropical fruit"), control = list (verbose=F))**

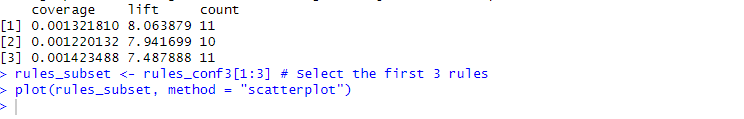
**rules\_conf3 <- sort (rules3, by="confidence", decreasing=TRUE) # 'high-confidence' rules.**

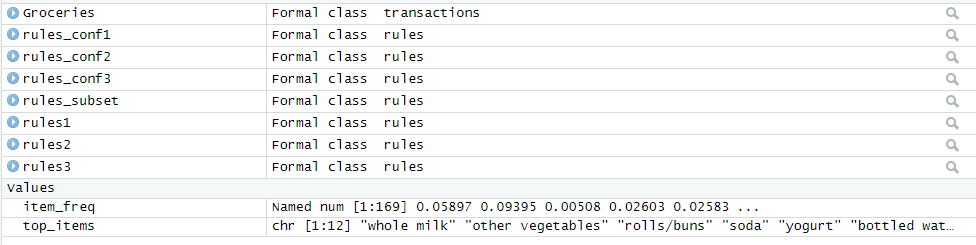
**inspect(head(rules\_conf3,n=3))**

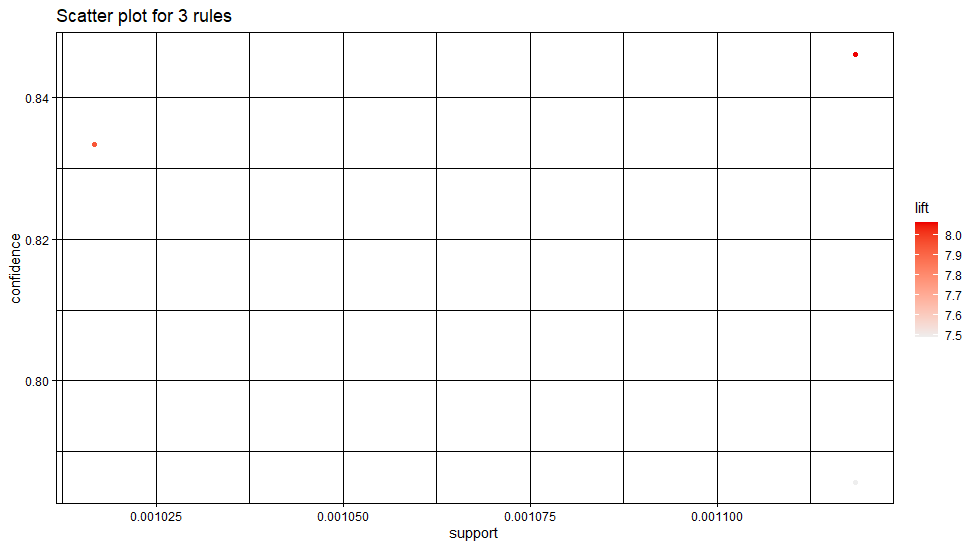
**rules\_subset <- rules\_conf3[1:3] # Select the first 3 rules**

**plot(rules\_subset, method = "scatterplot")**

****

****

****

****

**Description of this scatterplot –**

The scatterplot shows the distribution of the support and confidence values for the selected association rules. Each point in the plot represents an association rule, with the x-axis indicating the support value and the y-axis indicating the confidence value. The size of each point indicates the lift value, with larger points indicating higher lift values.

The scatterplot can be used to identify interesting and potentially useful association rules. In this case, we can see that all three rules have relatively low support values, indicating that these itemsets occur relatively infrequently in the transactions. However, they have high confidence values and moderate to high lift values, suggesting strong associations between the antecedent and consequent. This information can be used to guide marketing strategies, such as creating targeted promotions or product bundles that include these items together.

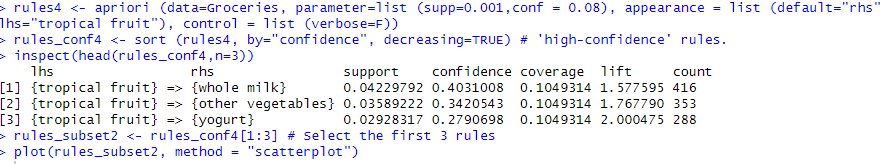
**rules4 <- apriori (data=Groceries, parameter=list (supp=0.001,conf = 0.08), appearance = list (default="rhs",lhs="tropical fruit"), control = list (verbose=F))**

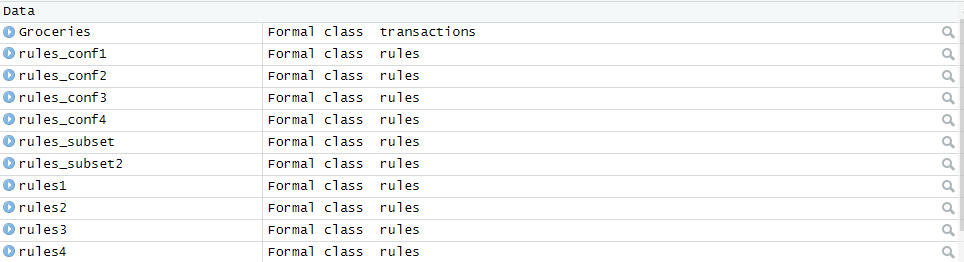
**rules\_conf4 <- sort (rules4, by="confidence", decreasing=TRUE) # 'high-confidence' rules.**

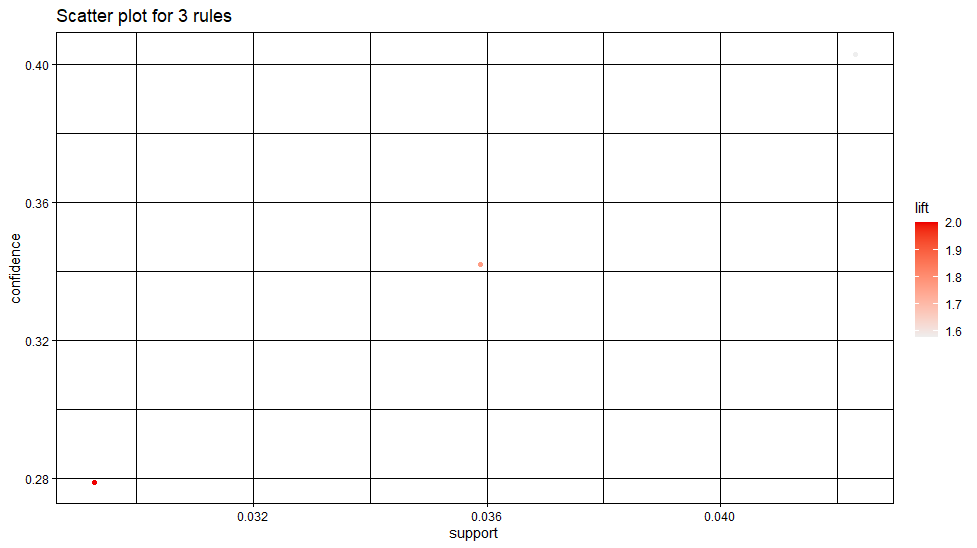
**inspect(head(rules\_conf4,n=3))**

**rules\_subset2 <- rules\_conf4[1:3] # Select the first 3 rules**

**plot(rules\_subset2, method = "scatterplot")**







**Description of this scatter plot**

The scatterplot shows the distribution of the support and confidence values for the selected association rules that have a "tropical fruit" in their antecedent. Each point in the plot represents an association rule, with the x-axis indicating the support value and the y-axis indicating the confidence value. The size of each point indicates the lift value, with larger points indicating higher lift values.

From the plot, we can see that all three rules have relatively high support values, indicating that these itemsets occur frequently in the transactions. They also have high confidence values, suggesting strong associations between the antecedent and consequent. The rule with "whole milk" in the consequent has the highest support value and moderate lift value. The rule with "yogurt" in the consequent has the highest lift value and moderate support value.

**6)Again using the plot() function in the arulesViz package, generate a plot for any three of your rules. This time, add two more arguments to the function: method="graph", engine="htmlwidget". What do you see now? Include a screenshot of your plot, along with the code you used to generate the plot. Describe your results in a sentence or two?**

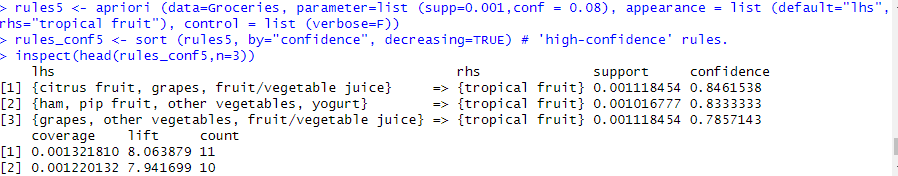
**rules5 <- apriori (data=Groceries, parameter=list (supp=0.001,conf = 0.08), appearance = list (default="lhs",rhs="tropical fruit"), control = list (verbose=F))**

**rules\_conf5 <- sort (rules5, by="confidence", decreasing=TRUE) # 'high-confidence' rules.**

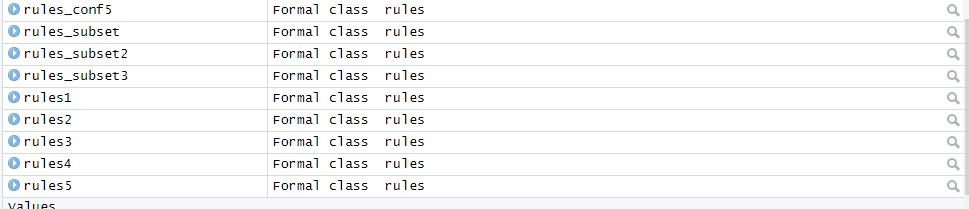
**inspect(head(rules\_conf5,n=3))**

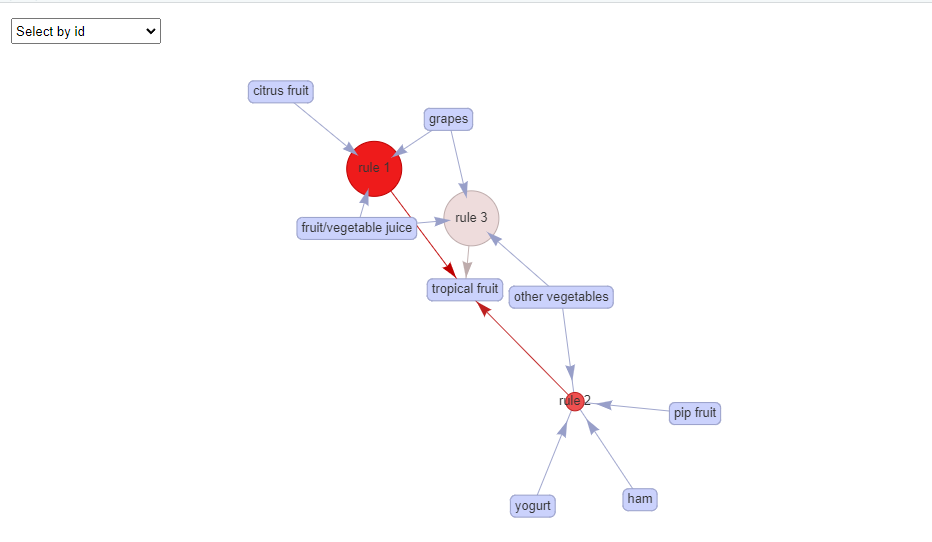
**rules\_subset3 <- rules\_conf5[1:3] # Select the first 3 rules**

**plot(rules\_subset3, method = "graph",engine="htmlwidget")**

****

****

****

****

**Description -**

The plot function is used to visualize the top three rules in a graph format, where the nodes represent the items and the edges indicate the association between the items. The size and color of the nodes and edges reflect the frequency and strength of the association, respectively. The "htmlwidget" engine is used to display the graph in an interactive format that allows for zooming and panning.

As seen above it is selecting by ID and each rule is connected by its id and shows a graph on which rule is linked with which grocery items for that rule.here as the rhs is tropical fruit and lhs has a lot other items that are linked with tropical fruit . it is a expanded graph.

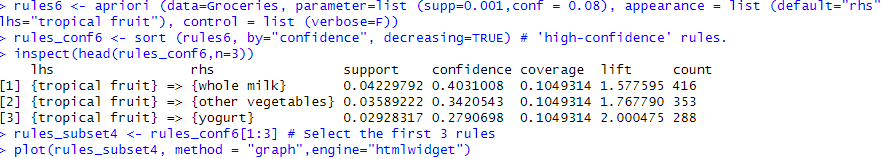
rules6 <- apriori (data=Groceries, parameter=list (supp=0.001,conf = 0.08), appearance = list (default="rhs",lhs="tropical fruit"), control = list (verbose=F))

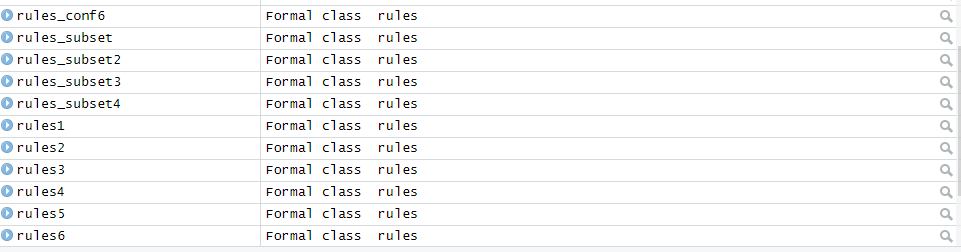
rules\_conf6 <- sort (rules6, by="confidence", decreasing=TRUE) # 'high-confidence' rules.

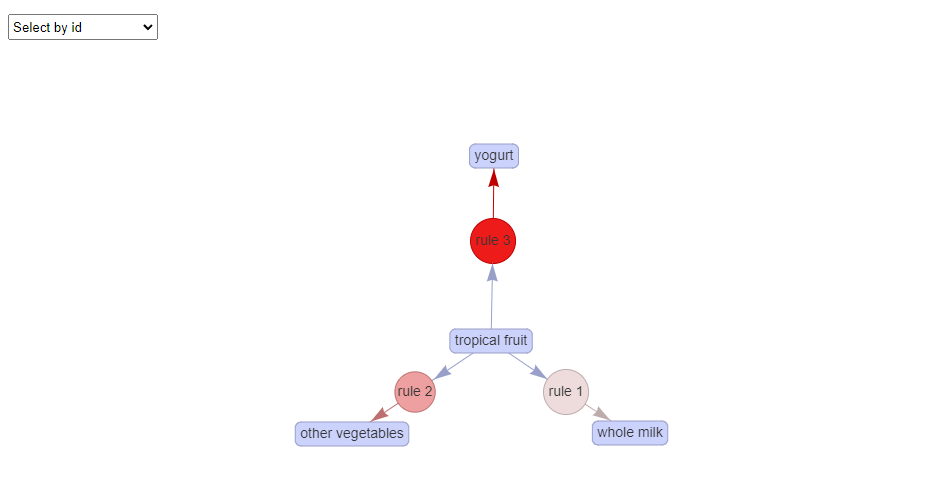
inspect(head(rules\_conf6,n=3))

rules\_subset4 <- rules\_conf6[1:3] # Select the first 3 rules

plot(rules\_subset4, method = "graph",engine="htmlwidget")







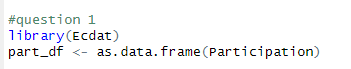
**Description –**

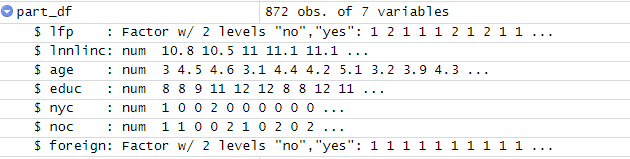
The plot function is used to visualize the top three rules in a graph format, where the nodes represent the items and the edges indicate the association between the items. The size and color of the nodes and edges reflect the frequency and strength of the association, respectively. The "htmlwidget" engine is used to display the graph in an interactive format that allows for zooming and panning.

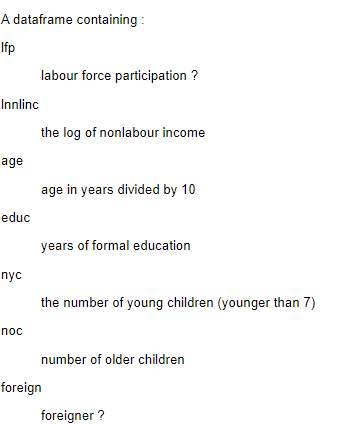
As seen above it is selecting by ID and each rule is connected by its id and shows a graph on which rule is linked with which grocery items for that rule. The only change is lhs is tropical fruit and rhs is just linked with one of the product, therefore the graph here is much shorter compared to the one above. Each rule shows the relation of how tropical fruit will entice to buy any of these 3 products.

**Task 2: Classification Tree**

1. **Bring the dataset Participation from the Ecdat package into your R environment. Use the ? or help() function to learn more about its variables. What does lfp mean?**

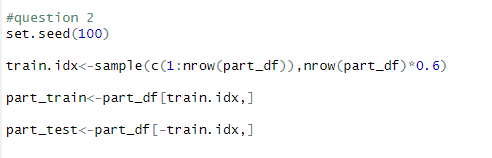






LFP stands for Labor Force Participation. It is a measure that calculates the proportion of the population that is either employed or actively seeking employment. It is an important economic indicator that is used to gauge the health of the labor market and overall economic conditions. A high LFP rate is generally considered to be a positive sign, as it suggests that a large percentage of the population is engaged in productive work, while a low LFP rate can indicate weak labor market conditions and potential economic difficulties.

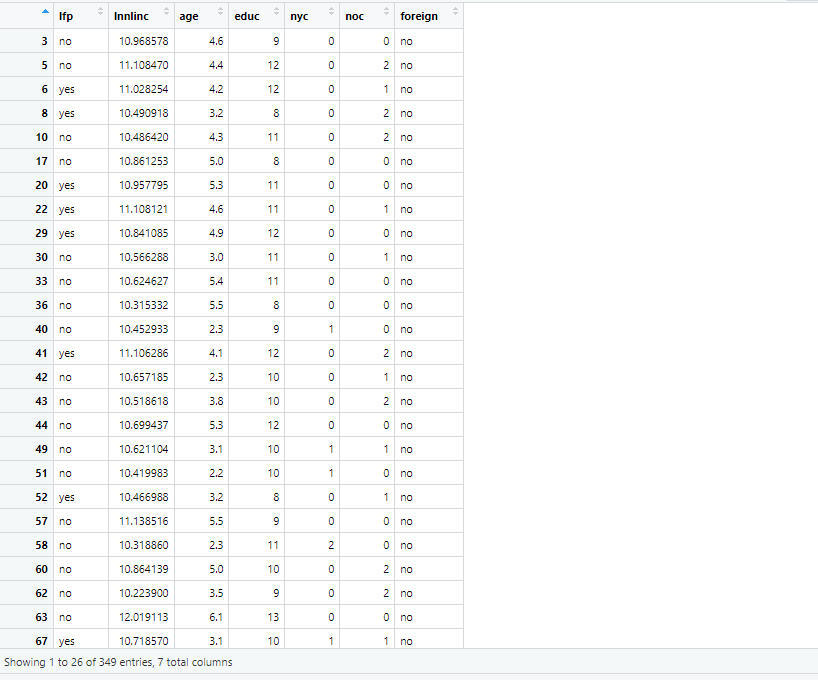
**2) Using your assigned seed value (from Assignment 2), partition your data into training (60%) and validation (40%) sets. Show the step(s) that you used to do this?**

****

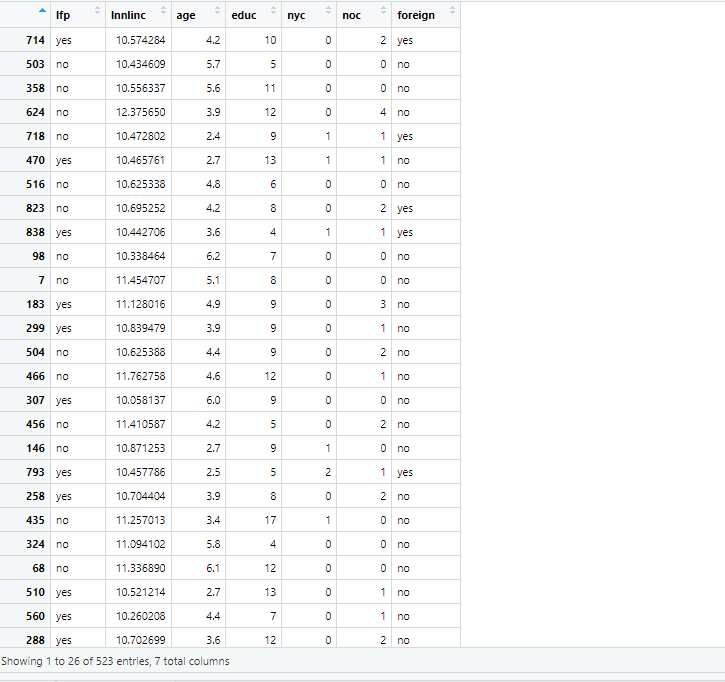
****

****

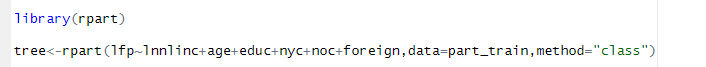
**Part\_test**

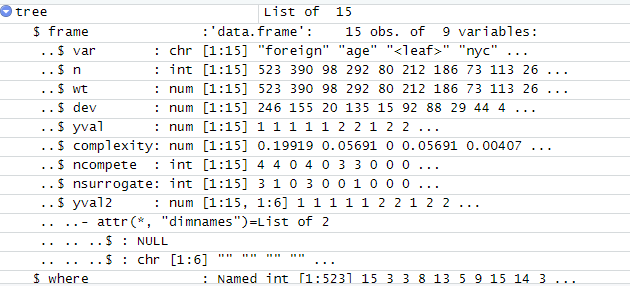
****

**Part\_train**

****

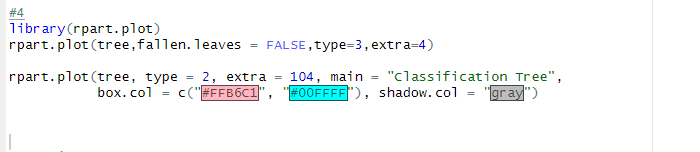
**3)Build a tree model with this dataset, using lfp as your outcome variable.**

****

****

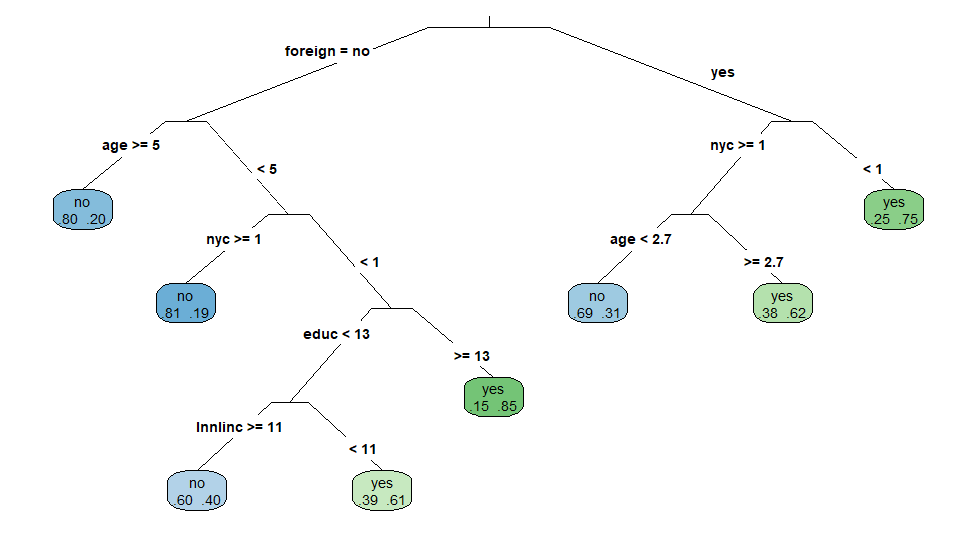
Use rpart.plot to display a classification tree that depicts your model. (If the tree model is hard to see or read, that’s okay...but if you wish to change it, you can try changing some settings, such as cex, fallen.leaves, and varlen).

1. Then, adjust the way your model looks. Don’t change anything about the model itself, but use a new combination of values for ‘type’ and/or ‘extra’ in rpart.plot to change the appearance of the tree

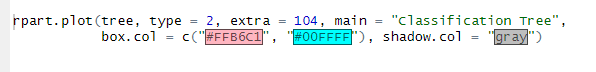
****

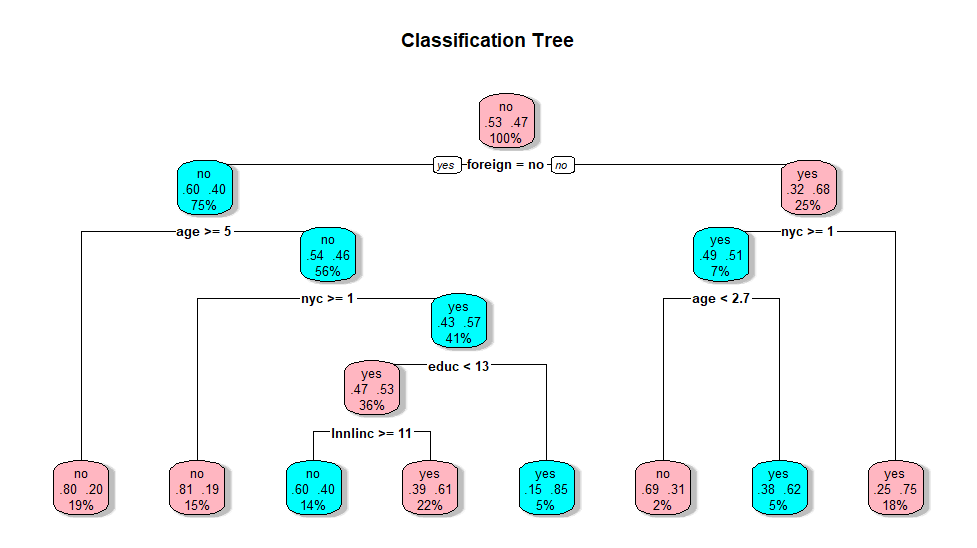
**4 a)**

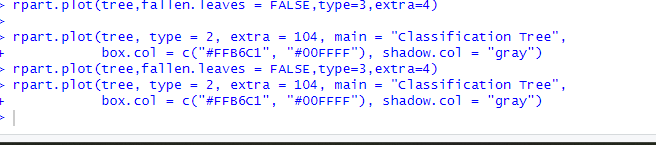
****

****

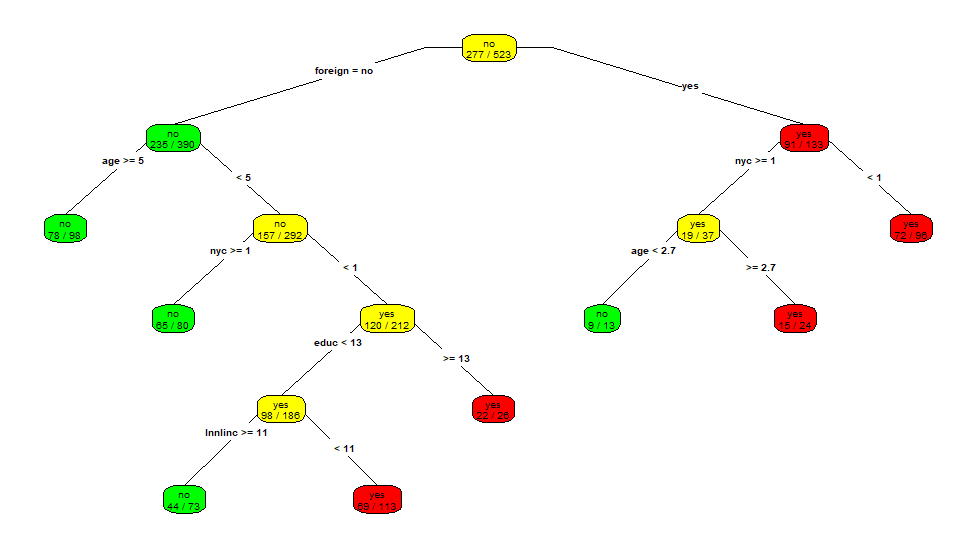
1. Try yet another alternative way of viewing your model. Show your results.

****

****

****

****

****

1. **Now, write a couple of sentences about what you saw with each of the three graphical versions of your model. Which one do you like best, and why?**

Among the three graphical representations, I find the second one to be the most effective. Unlike the first and the final graphs, the second one not only presents the probabilities associated with each node but also displays them in percentages, making it easier to understand. With this format, the tree becomes more comprehensible and user-friendly, allowing for better interpretation and analysis of the data.

**6. Describe the split that’s created at your tree’s root node (what variable did it split on, and what rule did it use?). Why is the root node significant?**

The root node in a decision tree is the first node that is split upon, and it sets the foundation for the entire tree. In this case, the root node is "foreigner" which means that the decision tree algorithm determined that this variable was the most important predictor of labour force participation. This is significant because it suggests that being a foreigner is a strong factor in determining whether someone will participate in the labour force or not, and all subsequent splits in the tree will be based on this variable. Understanding the split at the root node is crucial for interpreting the rest of the decision tree and making informed decisions based on the insights it provides.

**7. Did all the input variables from the dataset appear in your model diagram? If not, why not?**

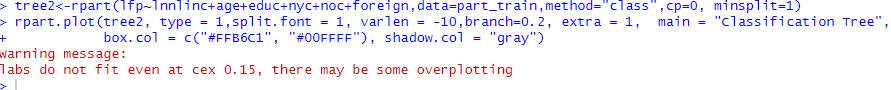
**NOC- Number of older children**

The decision tree algorithm may have determined that the "noc" variable was not as significant in predicting the outcome variable (lfp) as the other variables in the dataset. Therefore, the algorithm may have excluded it from the tree. Alternatively, the algorithm may have determined that including the "noc" variable did not improve the predictive power of the tree and therefore it was not included in the final model.

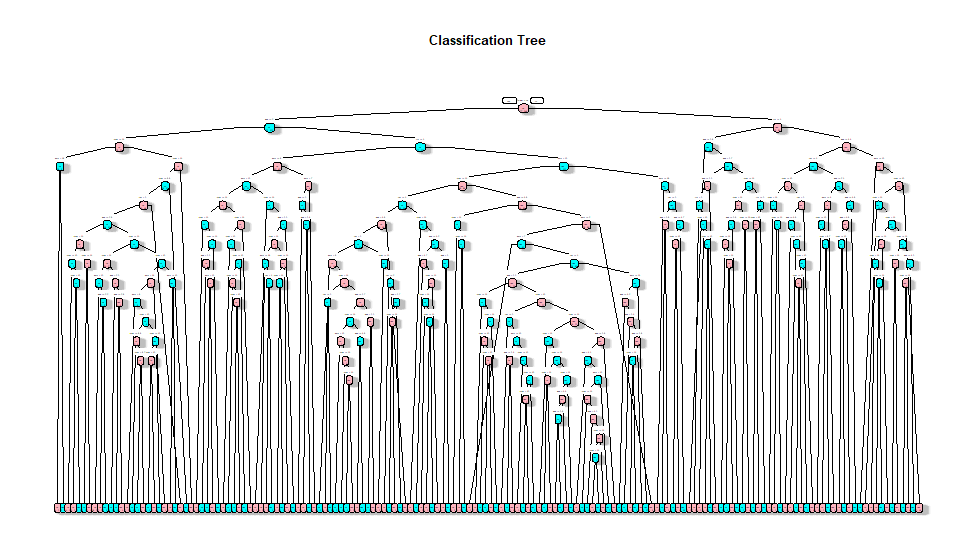
**8. Describe any one rule that your tree generates regarding whether a worker in Switzerland will participate in the labor force. To describe a rule, just trace any path along your tree from the root node to a terminal node.**

**Tracing -** If a person is a foreigner, aged greater than 50, has more than one young child, and has an education greater than 13, there is a 15% chance that they are in the labour force. This is the decision path traced by the tree.

**9. Now, build another tree model. This time, set a complexity parameter of 0, and use minsplit =2, to make the tree as large as possible. Show what your overfit tree looks like, using rpart.plot. Don’t worry about interpreting this tree – just show it?**

****

****

****

**10. Using five-fold cross-validation, determine the optimal complexity parameter (cp) for a tree model built with your training data. Demonstrate this by showing your cptable and stating which cp value you chose.**

> set.seed(100)

> cv.ct<-rpart(lfp~lnnlinc+age+educ+nyc+noc+foreign,data=part\_train,method="class",minsplit=2,cp=0,xval=5)

> printcp(cv.ct)

Classification tree:

rpart(formula = lfp ~ lnnlinc + age + educ + nyc + noc + foreign,

data = part\_train, method = "class", minsplit = 2, cp = 0,

xval = 5)

Variables actually used in tree construction:

[1] age educ foreign lnnlinc noc nyc

Root node error: 246/523 = 0.47036

n= 523

CP nsplit rel error xerror xstd

1 0.1991870 0 1.00000 1.00000 0.046400

2 0.0569106 1 0.80081 0.76016 0.044556

3 0.0304878 3 0.68699 0.78049 0.044810

4 0.0162602 5 0.62602 0.78049 0.044810

5 0.0108401 6 0.60976 0.78455 0.044859

6 0.0101626 9 0.57724 0.77236 0.044711

7 0.0091463 11 0.55691 0.80488 0.045091

8 0.0081301 18 0.49187 0.84553 0.045499

9 0.0060976 23 0.45122 0.86179 0.045642

10 0.0054201 33 0.38211 0.87805 0.045773

11 0.0040650 37 0.35772 0.92276 0.046076

12 0.0030488 78 0.19106 0.92683 0.046099

13 0.0027100 82 0.17886 0.96341 0.046278

14 0.0020325 103 0.11789 0.97154 0.046310

15 0.0000000 161 0.00000 0.97561 0.046325

****

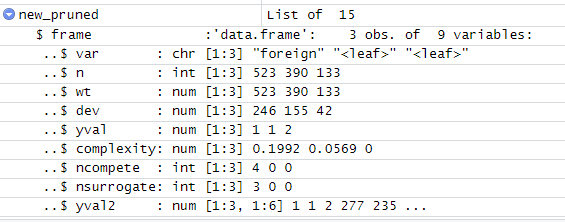
****

**The one with the lowest xerror will the cp value**

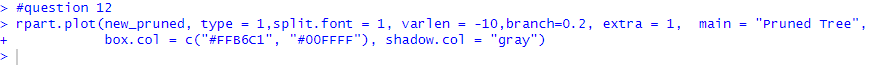
**11. Generate a new tree model, with the cp value that you found previously**

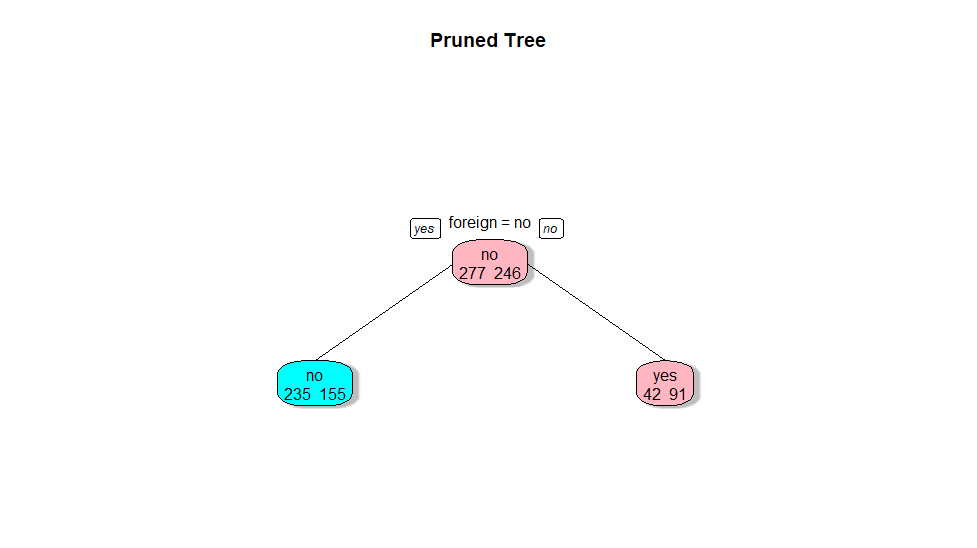
****

****

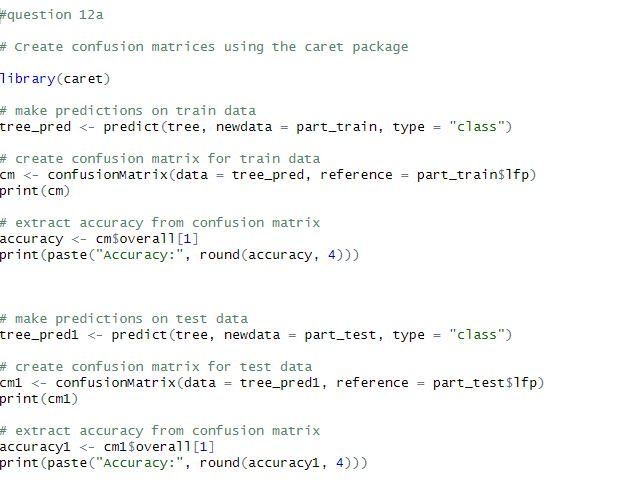
****

**12. Use rpart.plot to show your new tree model (the pruned tree). Show this with your preferred “type” and “extra” settings in rpart.plot.**

****

****

**12a. Create confusion matrices in R to assess the performance of your huge tree against your training and validation sets. How did it perform?**

****

Train data

> print(cm)

Confusion Matrix and Statistics

Reference

Prediction no yes

no 196 68

yes 81 178

Accuracy : 0.7151

95% CI : (0.6743, 0.7534)

No Information Rate : 0.5296

P-Value [Acc > NIR] : <2e-16

Kappa : 0.4299

Mcnemar's Test P-Value : 0.3256

Sensitivity : 0.7076

Specificity : 0.7236

Pos Pred Value : 0.7424

Neg Pred Value : 0.6873

Prevalence : 0.5296

Detection Rate : 0.3748

Detection Prevalence : 0.5048

Balanced Accuracy : 0.7156

'Positive' Class : no

|  |
| --- |
| > # extract accuracy from confusion matrix  > accuracy <- cm$overall[1]  > print(paste("Accuracy:", round(accuracy, 4)))  [1] "Accuracy: 0.7151" |
|  |
| |  | | --- | | > | |

**Test data –**

print(cm1)

Confusion Matrix and Statistics

Reference

Prediction no yes

no 132 59

yes 62 96

Accuracy : 0.6533

95% CI : (0.6008, 0.7032)

No Information Rate : 0.5559

P-Value [Acc > NIR] : 0.0001339

Kappa : 0.2992

Mcnemar's Test P-Value : 0.8557254

Sensitivity : 0.6804

Specificity : 0.6194

Pos Pred Value : 0.6911

Neg Pred Value : 0.6076

Prevalence : 0.5559

Detection Rate : 0.3782

Detection Prevalence : 0.5473

Balanced Accuracy : 0.6499

'Positive' Class : no

> # extract accuracy from confusion matrix

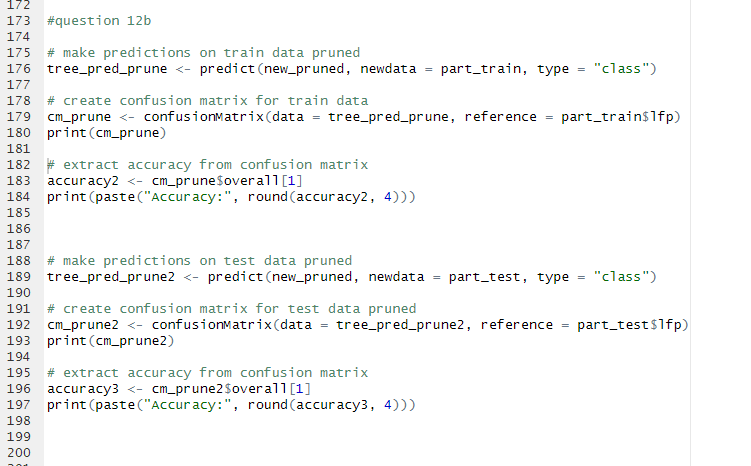
> accuracy1 <- cm1$overall[1]

> print(paste("Accuracy:", round(accuracy1, 4)))

[1] "Accuracy: 0.6533"

**Now, create confusion matrices to assess your optimally-sized tree model (the one that you built after cross-validation). How was this optimally-sized model’s performance against the training and validation sets? What happened to the difference between the two accuracy values as you went from the huge tree to the optimal one?**

Pruned tree training and testing confusion matrix



Training set prune

> print(cm\_prune)

Confusion Matrix and Statistics

Reference

Prediction no yes

no 235 155

yes 42 91

Accuracy : 0.6233

95% CI : (0.5802, 0.665)

No Information Rate : 0.5296

P-Value [Acc > NIR] : 9.520e-06

Kappa : 0.2241

Mcnemar's Test P-Value : 1.467e-15

Sensitivity : 0.8484

Specificity : 0.3699

Pos Pred Value : 0.6026

Neg Pred Value : 0.6842

Prevalence : 0.5296

Detection Rate : 0.4493

Detection Prevalence : 0.7457

Balanced Accuracy : 0.6091

'Positive' Class : no

> # extract accuracy from confusion matrix

> accuracy2 <- cm\_prune$overall[1]

> print(paste("Accuracy:", round(accuracy2, 4)))

[1] "Accuracy: 0.6233"

**Testing set prune**

> print(cm\_prune2)

Confusion Matrix and Statistics

Reference

Prediction no yes

no 167 99

yes 27 56

Accuracy : 0.639

95% CI : (0.5861, 0.6894)

No Information Rate : 0.5559

P-Value [Acc > NIR] : 0.0009872

Kappa : 0.233

Mcnemar's Test P-Value : 2.529e-10

Sensitivity : 0.8608

Specificity : 0.3613

Pos Pred Value : 0.6278

Neg Pred Value : 0.6747

Prevalence : 0.5559

Detection Rate : 0.4785

Detection Prevalence : 0.7622

Balanced Accuracy : 0.6111

'Positive' Class : no

> # extract accuracy from confusion matrix

> accuracy3 <- cm\_prune2$overall[1]

> print(paste("Accuracy:", round(accuracy3, 4)))

[1] "Accuracy: 0.639"

**What happened to the difference between the two accuracy values as you went from the huge tree to the optimal one?**

The training and testing accuracy may decrease from the huge trees to the pruned ones because as we prune a decision tree, we remove some of its branches and make it simpler. This process may result in a reduction of the model's ability to capture complex relationships in the training data, leading to a decrease in training accuracy. However, pruning may also help to reduce overfitting and improve the model's ability to generalize to new, unseen data, which can increase the testing accuracy.

It is also possible that the decrease in accuracy from the huge trees to the pruned ones is due to the pruning technique used or the hyperparameters chosen for the decision tree. For example, if the pruning technique is too aggressive, it may remove important branches from the tree and decrease accuracy. Similarly, if the hyperparameters are not optimized correctly, the model may not be able to capture the underlying patterns in the data, leading to a decrease in accuracy.

**Why would it be reasonable to expect that the difference between training set accuracy**

**and validation set accuracy would decrease when using a pruned tree?**

It is reasonable to expect that the difference between training set accuracy and validation set accuracy would decrease when using a pruned tree because pruning helps to reduce overfitting of the model to the training data. Overfitting occurs when a model is too complex and fits the training data too closely, resulting in poor generalization performance on new, unseen data. Pruning a decision tree removes some of its branches and simplifies it, making it less likely to overfit.

When a decision tree overfits, it can fit the training data too well, resulting in a larger difference between the training set accuracy and validation set accuracy. This is because the model is essentially memorizing the training data rather than learning the underlying patterns that can generalize to new data.