Homework 3 - Lukasz Grzybek

1.

```
#reads breast_cancer_updated
library(readr)
breast_cancer_updated <- read_csv("rr/breast_cancer_updated.csv")
```

```
## Rows: 699 Columns: 11
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (1): Class
## dbl (10): IDNumber, ClumpThickness, UniformCellSize, UniformCellShape, Margi...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
View(breast_cancer_updated)
```

```
#removes "IDNumber" column and sets data to a new variable
library(dbplyr)
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.3.6      ✓ dplyr   1.0.10
## ✓ tibble  3.1.8      ✓ stringr 1.4.1
## ✓ tidyr   1.2.1      ✓ forcats 0.5.2
## ✓ purrr   0.3.4
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::ident()  masks dbplyr::ident()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ dplyr::sql()    masks dbplyr::sql()
```

```
df <- breast_cancer_updated %>% select(-c( "IDNumber"))
summary(df)
```

```
##  ClumpThickness   UniformCellSize  UniformCellShape MarginalAdhesion
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000   1st Qu.: 1.000
##  Median : 4.000   Median : 1.000   Median : 1.000   Median : 1.000
##  Mean   : 4.418   Mean   : 3.134   Mean   : 3.207   Mean   : 2.807
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 5.000   3rd Qu.: 4.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##
##  EpithelialCellSize   BareNuclei     BlandChromatin   NormalNucleoli
##  Min.   : 1.000     Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 2.000     1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000
##  Median : 2.000     Median : 1.000   Median : 3.000   Median : 1.000
##  Mean   : 3.216     Mean   : 3.545   Mean   : 3.438   Mean   : 2.867
##  3rd Qu.: 4.000     3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 4.000
##  Max.   :10.000     Max.   :10.000   Max.   :10.000   Max.   :10.000
##                     NA's   :16
##     Mitoses          Class
##  Min.   : 1.000   Length:699
##  1st Qu.: 1.000   Class :character
##  Median : 1.000   Mode  :character
##  Mean   : 1.589
##  3rd Qu.: 1.000
##  Max.   :10.000
##
```

```
#checks each column for missing values
summary(df$ClumpThickness)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   4.000   4.418   6.000  10.000
```

```
summary(df$UniformCellSize)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   3.134   5.000  10.000
```

```
summary(df$UniformCellShape)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   3.207   5.000  10.000
```

```
summary(df$MarginalAdhesion)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   2.807   4.000  10.000
```

```
summary(df$EpithelialCellSize)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   3.216   4.000  10.000
```

```
summary(df$BareNuclei) #missing
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   1.000   1.000   3.545   6.000  10.000      16
```

```
summary(df$BlandChromatin)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   3.438   5.000  10.000
```

```
summary(df$NormalNucleoli)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   2.867   4.000  10.000
```

```
summary(df$Mitoses)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.589   1.000  10.000
```

```
summary(df$Class)
```

```
##    Length     Class      Mode
##       699 character character
```

```
#removing missing values
df <- df %>% drop_na(BareNuclei)
summary(df$BareNuclei)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   3.545   6.000  10.000
```

```
#using 10-fold cross validation to train the data
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(lattice)
train_control = trainControl(method = "cv", number = 10)
tree <- train(Class ~., data = df, method = "rpart", trControl = train_control)
tree
```

```
## CART
##
## 683 samples
##   9 predictor
##   2 classes: 'benign', 'malignant'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 614, 616, 614, 614, 615, 615, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.02510460  0.9354821  0.8592118
##   0.05439331  0.9193044  0.8235750
##   0.79079498  0.8246539  0.5443795
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.0251046.
```

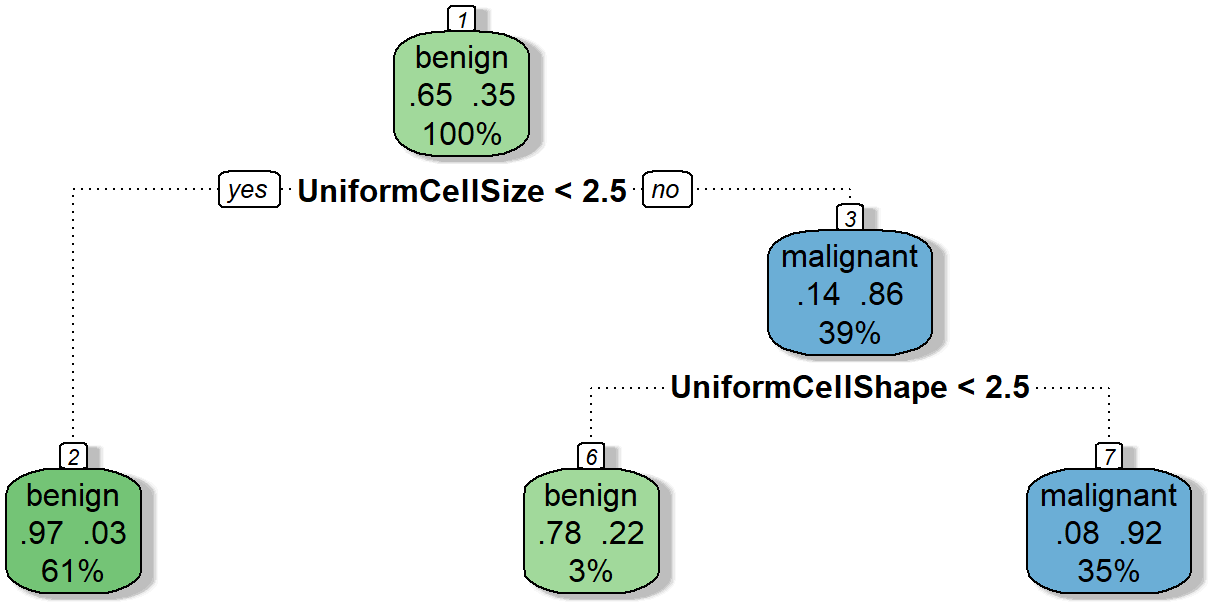
Accuracy using 10-fold cross validation of breast cancer malignancy was 0.9444587 with a cp of 0.02510460 and a kappa of 0.8771488.

```
#visualizes a decision tree
library(rattle)
```

```
## Loading required package: bitops
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
fancyRpartPlot(tree$finalModel, caption = "")
```




```
#rules that correspond to the above decision tree
if ("UniformCellSize"<2.5){
  df$Class == "benign"

} else{
  df$Class == "malignant"
  if ("UniformCellShape"<2.5){
  df$Class == "benign"
}else{
  df$Class == "malignant"
}
}
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13]  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
##  [25]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE
##  [37] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE
##  [49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [61]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE
##  [73]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
##  [85]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
## [109] FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [121]  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [145] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE
## [157] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
## [169]  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE
## [181]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [193] FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
## [205]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE
## [217]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
## [229] FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [241]  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE
## [253]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
## [265]  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
## [277]  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
## [289]  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
## [301]  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
## [313]  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE
## [325] FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE
## [337] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [349] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [361] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [373]  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [385] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [397] FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
## [409] FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [421]  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [433] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
## [445] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE
## [457] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
## [469]  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## [481] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [493] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
## [505]  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [517] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [529] FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [541] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## [553] FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [565] FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE
## [577]  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [589]  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## [601] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
```

```
## [613] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
## [625] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [637] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## [649] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
## [661] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## [673] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

2.

```
library(dplyr)
summary(storms)
```

```
##      name                year          month             day
##  Length:11859       Min.   :1975   Min.   : 1.000   Min.   : 1.00
##  Class :character   1st Qu.:1992   1st Qu.: 8.000   1st Qu.: 8.00
##  Mode  :character   Median :2002   Median : 9.000   Median :16.00
##                     Mean   :2001   Mean   : 8.785   Mean   :15.83
##                     3rd Qu.:2011   3rd Qu.: 9.000   3rd Qu.:24.00
##                     Max.   :2020   Max.   :12.000   Max.   :31.00
##
##       hour             lat             long            status
##  Min.   : 0.000   Min.   : 7.20   Min.   :-109.30   Length:11859
##  1st Qu.: 6.000   1st Qu.:17.50   1st Qu.: -80.70   Class :character
##  Median :12.000   Median :24.60   Median : -64.40   Mode  :character
##  Mean   : 9.117   Mean   :24.76   Mean   : -64.09
##  3rd Qu.:18.000   3rd Qu.:31.30   3rd Qu.: -48.40
##  Max.   :23.000   Max.   :51.90   Max.   :  -6.00
##
##  category       wind           pressure    tropicalstorm_force_diameter
##  -1:2898   Min.   : 10.00   Min.   : 882   Min.   :  0.0
##  0 :5347   1st Qu.: 35.00   1st Qu.: 985   1st Qu.: 60.0
##  1 :1934   Median : 45.00   Median : 999   Median :120.0
##  2 : 749   Mean   : 53.64   Mean   : 992   Mean   :145.3
##  3 : 434   3rd Qu.: 65.00   3rd Qu.:1006   3rd Qu.:210.0
##  4 : 411   Max.   :160.00   Max.   :1022   Max.   :870.0
##  5 :  86                                   NA's   :6509
##  hurricane_force_diameter
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 18.15
##  3rd Qu.: 25.00
##  Max.   :300.00
##  NA's   :6509
```

```
#sets storms data as data frame
dfb <- as.data.frame(storms)
```

```
#changes the "category" column from a character to a factor
dfb$category <- as.factor(dfb$category)
```

```
#checks for missing values
summary(dfb$category)
```

```
##   -1    0    1    2    3    4    5
## 2898 5347 1934  749  434  411   86
```

```
sum(is.na(dfb$category))
```

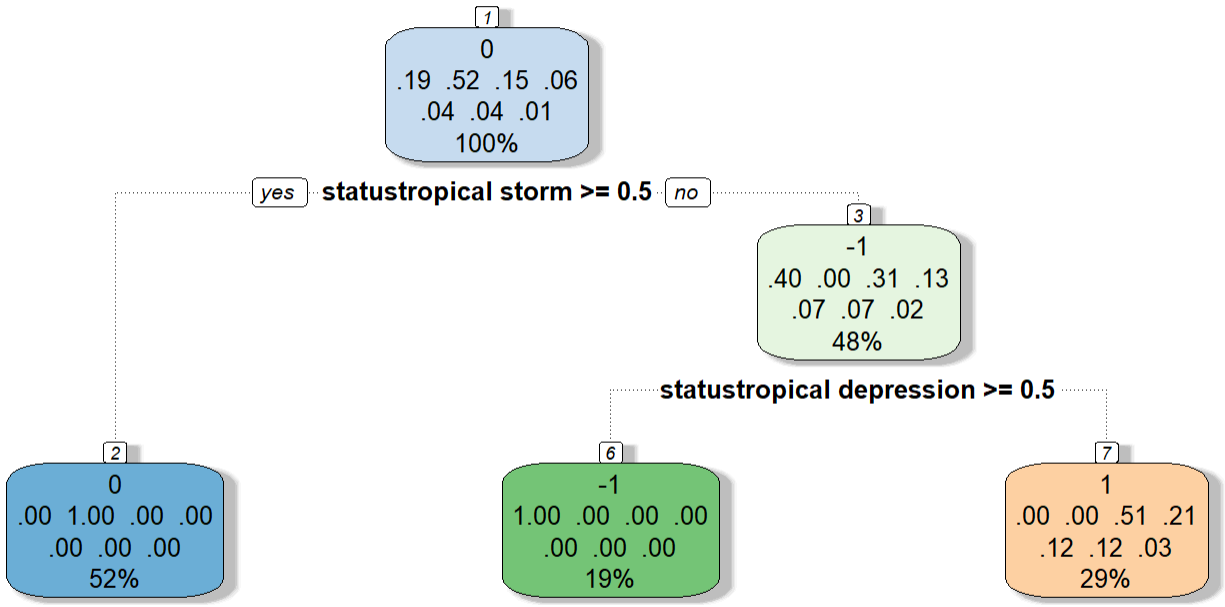
```
## [1] 0
```

```
#uses 10-fold cross validation to train the data

library(rpart)
library(tidyverse)
library(caret)

train_control = trainControl(method = "cv", number = 10)

#missing values (NA) excluded from the training
#hyperparameters of maxdepth=2, minsplit=5 and minbucket=3 set
#training performed on the "category" variable using rpart1SE
tree2 <- train(category ~., data = dfb, control = rpart.control(minsplit =  5, maxdepth = 2,
minbucket = 3), trControl = train_control, method = "rpart1SE", na.action=na.exclude)
tree2
```

```
## CART
##
## 11859 samples
##    12 predictor
##     7 classes: '-1', '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4815, 4814, 4816, 4815, 4814, 4816, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8594412  0.7842289
```

Accuracy scored returned using cross validation is 0.8594397 with a kappa of 0.7842258

```
#visualizes a decision tree
library(rattle)
fancyRpartPlot(tree2$finalModel, caption = "")
```




```
#missing values (NA) excluded and saved as new variable
#new data partition created, 70% for train set, 30% for test set
#confusion matrix for test set created

dfc <- na.exclude(dfb)
index = createDataPartition(y=dfc$category, p=0.7, list=FALSE)
train_set = dfc[index,]
test_set = dfc[-index,]
pred_tree <- predict(tree2, test_set)
confusionMatrix(test_set$category, pred_tree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  -1   0   1   2   3   4   5
##         -1 309   0   0   0   0   0   0
##         0    0 831   0   0   0   0   0
##         1    0   1 238   0   0   0   0
##         2    0   0  96   0   0   0   0
##         3    0   0  57   0   0   0   0
##         4    0   0  57   0   0   0   0
##         5    0   0  14   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.8596
##                  95% CI : (0.8417, 0.8763)
##     No Information Rate : 0.519
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7843
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: -1 Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             1.0000   0.9988   0.5152       NA       NA       NA
## Specificity             1.0000   1.0000   0.9991  0.94011  0.96444  0.96444
## Pos Pred Value          1.0000   1.0000   0.9958       NA       NA       NA
## Neg Pred Value          1.0000   0.9987   0.8358       NA       NA       NA
## Prevalence              0.1928   0.5190   0.2882  0.00000  0.00000  0.00000
## Detection Rate          0.1928   0.5184   0.1485  0.00000  0.00000  0.00000
## Detection Prevalence    0.1928   0.5184   0.1491  0.05989  0.03556  0.03556
## Balanced Accuracy       1.0000   0.9994   0.7571       NA       NA       NA
##                      Class: 5
## Sensitivity                NA
## Specificity          0.991266
## Pos Pred Value             NA
## Neg Pred Value             NA
## Prevalence           0.000000
## Detection Rate       0.000000
## Detection Prevalence 0.008734
## Balanced Accuracy          NA
```

```
#confusion matrix for train set created
pred_tree2 <- predict(tree2, train_set)
confusionMatrix(train_set$category, pred_tree2)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   -1    0    1    2    3    4    5
##         -1  722    0    0    0    0    0    0
##         0     0 1940    0    0    0    0    0
##         1     0    0  558    0    0    0    0
##         2     0    0  227    0    0    0    0
##         3     0    0  133    0    0    0    0
##         4     0    0  134    0    0    0    0
##         5     0    0   33    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.8594
##                  95% CI : (0.8478, 0.8703)
##     No Information Rate : 0.5177
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7842
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: -1 Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             1.0000   1.0000   0.5143       NA       NA       NA
## Specificity             1.0000   1.0000   1.0000  0.93942   0.9645  0.96424
## Pos Pred Value          1.0000   1.0000   1.0000       NA       NA       NA
## Neg Pred Value          1.0000   1.0000   0.8347       NA       NA       NA
## Prevalence              0.1927   0.5177   0.2896  0.00000   0.0000  0.00000
## Detection Rate          0.1927   0.5177   0.1489  0.00000   0.0000  0.00000
## Detection Prevalence    0.1927   0.5177   0.1489  0.06058   0.0355  0.03576
## Balanced Accuracy       1.0000   1.0000   0.7571       NA       NA       NA
##                      Class: 5
## Sensitivity                NA
## Specificity          0.991193
## Pos Pred Value             NA
## Neg Pred Value             NA
## Prevalence           0.000000
## Detection Rate       0.000000
## Detection Prevalence 0.008807
## Balanced Accuracy          NA
```

The test set resulted in a slightly higher accuracy of 0.8603 than compared with the accuracy of the train set (0.8591). There are problems with classifying classes 2,3,4, and 5 because of missing values. Because the accuracy scores of both the traing and test sets are very close,this suggests there might not be any overfitting in the data.

3.

```
#new data partition created, 80% for train set, 20% for test set
library(rpart)
library(dbplyr)
library(caret)
index = createDataPartition(y=dfc$category, p=0.8, list=FALSE)
train_set80 = dfc[index,]
test_set20 = dfc[-index,]
```

```
# build many decision trees, tuning the parameters to find the best tree model (most accurate
with least complexity) and one that avoids overfitting and creating a table to display accura
cy and complexity of each tree

train_control = trainControl(method = "cv", number = 10)

tree3a <- train(category ~., data = train_set80, control = rpart.control(minsplit =  2, maxde
pth = 1, minbucket = 2), trControl = train_control, method = "rpart1SE")



pred_tree <- predict(tree3a, train_set80)
cfm_train3a <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3a, test_set20)
cfm_test3a <- confusionMatrix(test_set20$category, pred_tree)

train_tree3a <- cfm_train3a$overall[1]
test_tree3a <- cfm_test3a$overall[1]
nodes <- nrow(tree3a$finalModel$frame)

comp_tbl <- data.frame("Nodes" = nodes, "TrainAccuracy" = train_tree3a, "TestAccuracy" = test
_tree3a,"MaxDepth" = 1, "Minsplit" = 2, "Minbucket" = 2)

tree3b <- train(category ~., data = train_set80, control = rpart.control(minsplit =  5, maxde
pth = 3, minbucket = 5), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3b, train_set80)
cfm_train3b <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3b, test_set20)
cfm_test3b <- confusionMatrix(test_set20$category, pred_tree)

train_tree3b <- cfm_train3b$overall[1]
test_tree3b <- cfm_test3b$overall[1]
nodes <- nrow(tree3b$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3b, test_tree3b, 3, 5, 5))

tree3c <- train(category ~., data = train_set80, control = rpart.control(minsplit =  12, maxd
epth = 5, minbucket = 12), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3c, train_set80)
cfm_train3c <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3c, test_set20)
cfm_test3c <- confusionMatrix(test_set20$category, pred_tree)

train_tree3c <- cfm_train3c$overall[1]
test_tree3c <- cfm_test3c$overall[1]
nodes <- nrow(tree3c$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3c, test_tree3c, 5, 12, 12))
```

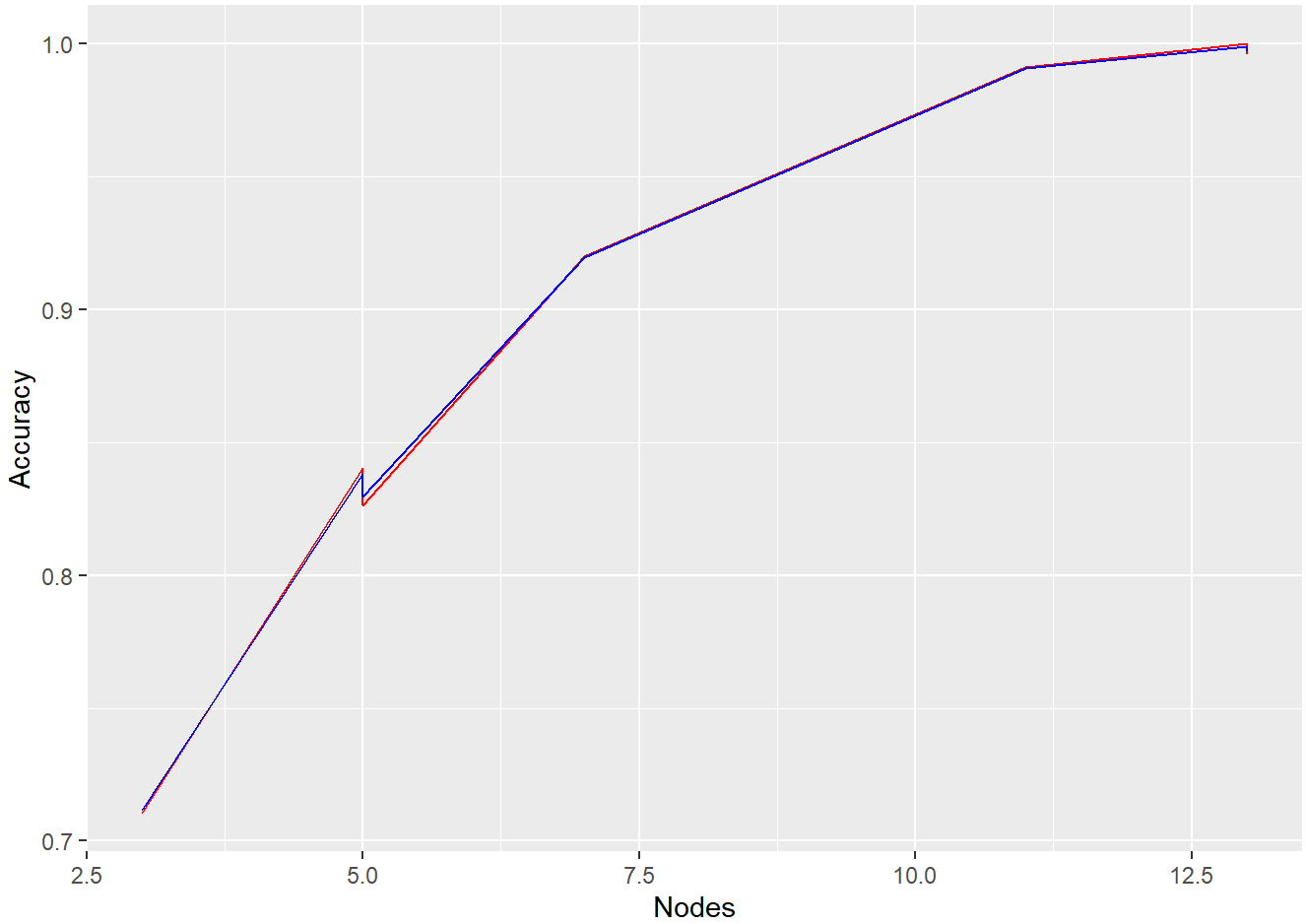
```
tree3d <- train(category ~., data = train_set80, control = rpart.control(minsplit =  30, maxd
epth = 7, minbucket = 30), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3d, train_set80)
cfm_train3d <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3d, test_set20)
cfm_test3d <- confusionMatrix(test_set20$category, pred_tree)

train_tree3d <- cfm_train3d$overall[1]
test_tree3d <- cfm_test3d$overall[1]
nodes <- nrow(tree3d$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3d, test_tree3d, 7, 30, 30))

tree3e <- train(category ~., data = train_set80, control = rpart.control(minsplit =  50, maxd
epth = 9, minbucket = 50), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3e, train_set80)
cfm_train3e <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3e, test_set20)
cfm_test3e <- confusionMatrix(test_set20$category, pred_tree)

train_tree3e <- cfm_train3e$overall[1]
test_tree3e <- cfm_test3e$overall[1]
nodes <- nrow(tree3e$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3e, test_tree3e, 9, 50, 50))


tree3f <- train(category ~., data = train_set80, control = rpart.control(minsplit =  500, max
depth = 11, minbucket = 500), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3f, train_set80)
cfm_train3f <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3f, test_set20)
cfm_test3f <- confusionMatrix(test_set20$category, pred_tree)

train_tree3f <- cfm_train3f$overall[1]
test_tree3f <- cfm_test3f$overall[1]
nodes <- nrow(tree3f$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3f, test_tree3f, 11, 500, 500))

tree3g <- train(category ~., data = train_set80, control = rpart.control(minsplit =  900, max
depth = 13, minbucket = 900), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3g, train_set80)
cfm_train3g <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3g, test_set20)
cfm_test3g <- confusionMatrix(test_set20$category, pred_tree)
```

```
train_tree3g <- cfm_train3g$overall[1]
test_tree3g <- cfm_test3g$overall[1]
nodes <- nrow(tree3g$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3g, test_tree3g, 13, 900, 900))

tree3h <- train(category ~., data = train_set80, control = rpart.control(minsplit =  2000, ma
xdepth = 15, minbucket = 2000), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3h, train_set80)
cfm_train3h <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3h, test_set20)
cfm_test3h <- confusionMatrix(test_set20$category, pred_tree)

train_tree3h <- cfm_train3h$overall[1]
test_tree3h <- cfm_test3h$overall[1]
nodes <- nrow(tree3h$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3h, test_tree3h, 15, 2000, 2000))

tree3i <- train(category ~., data = train_set80, control = rpart.control(minsplit =  1000, ma
xdepth = 17, minbucket = 1000), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3i, train_set80)
cfm_train3i <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3i, test_set20)
cfm_test3i <- confusionMatrix(test_set20$category, pred_tree)

train_tree3i <- cfm_train3i$overall[1]
test_tree3i <- cfm_test3i$overall[1]
nodes <- nrow(tree3i$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3i, test_tree3i, 17, 1000, 1000))

tree3j <- train(category ~., data = train_set80, control = rpart.control(minsplit =  2000, ma
xdepth = 19, minbucket = 2000), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3j, train_set80)
cfm_train3j <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3j, test_set20)
cfm_test3j <- confusionMatrix(test_set20$category, pred_tree)

train_tree3j <- cfm_train3j$overall[1]
test_tree3j <- cfm_test3j$overall[1]
nodes <- nrow(tree3j$finalModel$frame)

comp_tbl <- comp_tbl %>% rbind(list(nodes, train_tree3j, test_tree3j, 19, 2000, 2000))

comp_tbl
```

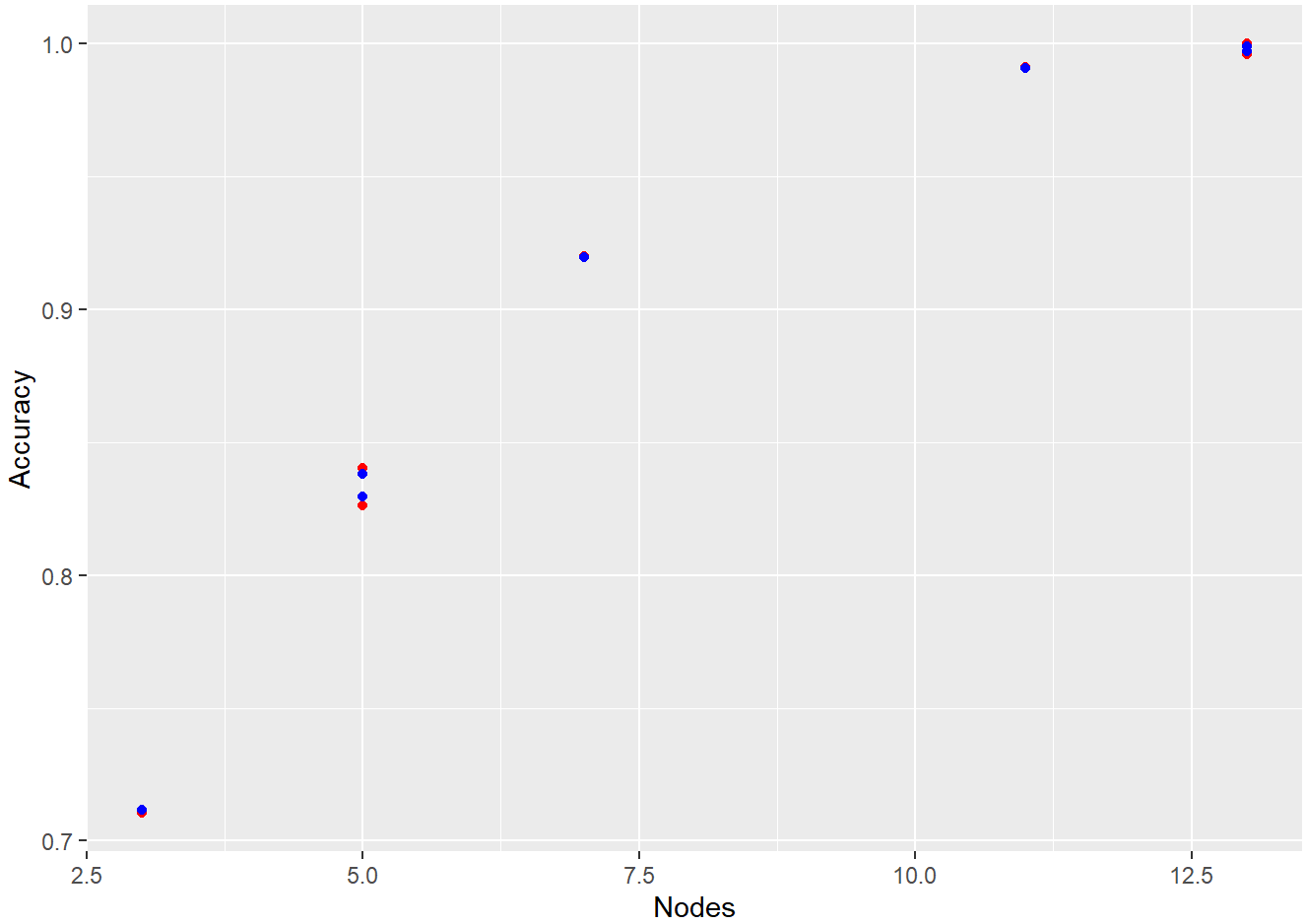
	Nodes <int>	TrainAccuracy <dbl>	TestAccuracy <dbl>	MaxDepth <dbl>	Minsplit <dbl>	Minbucket <dbl>
Accuracy	3	0.7104157	0.7116105	1	2	2
1	7	0.9198972	0.9194757	3	5	5
11	11	0.9911256	0.9906367	5	12	12
12	13	1.0000000	0.9990637	7	30	30
13	13	0.9960299	0.9971910	9	50	50
14	7	0.9198972	0.9194757	11	500	500
15	5	0.8402616	0.8380150	13	900	900
16	3	0.7104157	0.7116105	15	2000	2000
17	5	0.8262494	0.8295880	17	1000	1000
18	3	0.7104157	0.7116105	19	2000	2000

1-10 of 10 rows

```
#plots table

library(ggplot2)
plot1 <- ggplot(comp_tbl, aes(x=Nodes)) +
  geom_line(aes(y = TrainAccuracy), color = "red") +
  geom_line(aes(y = TestAccuracy), color="blue") +
  ylab("Accuracy")

plot2 <- ggplot(comp_tbl, aes(x=Nodes)) +
  geom_point(aes(y = TrainAccuracy), color = "red") +
  geom_point(aes(y = TestAccuracy), color="blue") +
  ylab("Accuracy")

plot1
```




```
plot2
```




```
#tree3g is the final model of choice with a MaxDepth of 13, Minsplit and Minbucket of 900
#tree3g evaluated by applying a confusion matrix to the train and test sets. Then accuracy sc
ore is double checked using 10 fold cross validation

train_control = trainControl(method = "cv", number = 10)

tree3final <- train(category ~., data = dfc, control = rpart.control(minsplit =  900, maxdept
h = 13, minbucket = 900), trControl = train_control, method = "rpart1SE")

pred_tree <- predict(tree3final, train_set80)
cfm_train3finaltrain <- confusionMatrix(train_set80$category, pred_tree)
pred_tree <- predict(tree3final, test_set20)
cfm_test3finaltest <- confusionMatrix(test_set20$category, pred_tree)


cfm_train3finaltrain
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   -1    0    1    2    3    4    5
##         -1  825    0    0    0    0    0    0
##         0     0 2217    0    0    0    0    0
##         1     0    0  638    0    0    0    0
##         2     0    0  259    0    0    0    0
##         3     0    0  152    0    0    0    0
##         4     0    0  153    0    0    0    0
##         5     0    0   38    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.8594
##                  95% CI : (0.8486, 0.8697)
##     No Information Rate : 0.5177
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7843
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: -1 Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             1.0000   1.0000   0.5145       NA       NA       NA
## Specificity             1.0000   1.0000   1.0000  0.93951   0.9645  0.96427
## Pos Pred Value          1.0000   1.0000   1.0000       NA       NA       NA
## Neg Pred Value          1.0000   1.0000   0.8348       NA       NA       NA
## Prevalence              0.1927   0.5177   0.2896  0.00000   0.0000  0.00000
## Detection Rate          0.1927   0.5177   0.1490  0.00000   0.0000  0.00000
## Detection Prevalence    0.1927   0.5177   0.1490  0.06049   0.0355  0.03573
## Balanced Accuracy       1.0000   1.0000   0.7573       NA       NA       NA
##                      Class: 5
## Sensitivity                NA
## Specificity          0.991126
## Pos Pred Value             NA
## Neg Pred Value             NA
## Prevalence           0.000000
## Detection Rate       0.000000
## Detection Prevalence 0.008874
## Balanced Accuracy          NA
```

```
cfm_test3finaltest
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  -1   0   1   2   3   4   5
##         -1 206   0   0   0   0   0   0
##         0    0 554   0   0   0   0   0
##         1    0   1 158   0   0   0   0
##         2    0   0  64   0   0   0   0
##         3    0   0  38   0   0   0   0
##         4    0   0  38   0   0   0   0
##         5    0   0   9   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.8596
##                  95% CI : (0.8373, 0.8798)
##     No Information Rate : 0.5197
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7841
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: -1 Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity             1.0000   0.9982   0.5147       NA       NA       NA
## Specificity             1.0000   1.0000   0.9987  0.94007  0.96442  0.96442
## Pos Pred Value          1.0000   1.0000   0.9937       NA       NA       NA
## Neg Pred Value          1.0000   0.9981   0.8361       NA       NA       NA
## Prevalence              0.1929   0.5197   0.2875  0.00000  0.00000  0.00000
## Detection Rate          0.1929   0.5187   0.1479  0.00000  0.00000  0.00000
## Detection Prevalence    0.1929   0.5187   0.1489  0.05993  0.03558  0.03558
## Balanced Accuracy       1.0000   0.9991   0.7567       NA       NA       NA
##                      Class: 5
## Sensitivity                NA
## Specificity          0.991573
## Pos Pred Value             NA
## Neg Pred Value             NA
## Prevalence           0.000000
## Detection Rate       0.000000
## Detection Prevalence 0.008427
## Balanced Accuracy          NA
```

```
tree3final
```

```
## CART
##
## 5350 samples
##   12 predictor
##    7 classes: '-1', '0', '1', '2', '3', '4', '5'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 4815, 4816, 4814, 4813, 4815, 4815, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8594403  0.7842272
```

Using cross validation, accuracy score is0.8594408 with a kappa of 0.784228

4.

```
#imports dataset
library(readr)
Bank_Modified <- read_csv("rr/Bank_Modified.csv")
```

```
## New names:
## Rows: 690 Columns: 13
## ── Column specification
## ──────────────────────────────────────────────────────── Delimiter: "," chr
## (1): approval dbl (9): ...1, cont1, cont2, cont3, cont4, cont5, cont6,
## credit.score, ages lgl (3): bool1, bool2, bool3
## ℹ Use `spec()` to retrieve the full column specification for this data. ℹ
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## • `` -> `...1`
```

```
View(Bank_Modified)
```

```
summary(Bank_Modified)
```

```
##       ...1           cont1           cont2            cont3
##  Min.   :  1.0   Min.   :13.75   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:173.2   1st Qu.:22.60   1st Qu.: 1.000   1st Qu.: 0.165
##  Median :345.5   Median :28.46   Median : 2.750   Median : 1.000
##  Mean   :345.5   Mean   :31.57   Mean   : 4.759   Mean   : 2.223
##  3rd Qu.:517.8   3rd Qu.:38.23   3rd Qu.: 7.207   3rd Qu.: 2.625
##  Max.   :690.0   Max.   :80.25   Max.   :28.000   Max.   :28.500
##                  NA's   :12
##    bool1           bool2             cont4        bool3             cont5
##  Mode :logical   Mode :logical   Min.   : 0.0   Mode :logical   Min.   :   0
##  FALSE:329       FALSE:395       1st Qu.: 0.0   FALSE:374       1st Qu.:  75
##  TRUE :361       TRUE :295       Median : 0.0   TRUE :316       Median : 160
##                                  Mean   : 2.4                   Mean   : 184
##                                  3rd Qu.: 3.0                   3rd Qu.: 276
##                                  Max.   :67.0                   Max.   :2000
##                                                                 NA's   :13
##      cont6            approval          credit.score        ages
##  Min.   :     0.0   Length:690         Min.   :583.7   Min.   :17.00
##  1st Qu.:     0.0   Class :character   1st Qu.:666.7   1st Qu.:31.00
##  Median :     5.0   Mode  :character   Median :697.3   Median :38.00
##  Mean   :  1017.4                      Mean   :696.4   Mean   :39.59
##  3rd Qu.:   395.5                      3rd Qu.:726.4   3rd Qu.:47.00
##  Max.   :100000.0                      Max.   :806.0   Max.   :84.00
##
```

```
#removes "...1" column and sets data to a new variable
library(dbplyr)
df4 <- Bank_Modified %>% select(-c( "...1"))
summary(df4)
```

```
##      cont1           cont2            cont3          bool1
##  Min.   :13.75   Min.   : 0.000   Min.   : 0.000   Mode :logical
##  1st Qu.:22.60   1st Qu.: 1.000   1st Qu.: 0.165   FALSE:329
##  Median :28.46   Median : 2.750   Median : 1.000   TRUE :361
##  Mean   :31.57   Mean   : 4.759   Mean   : 2.223
##  3rd Qu.:38.23   3rd Qu.: 7.207   3rd Qu.: 2.625
##  Max.   :80.25   Max.   :28.000   Max.   :28.500
##  NA's   :12
##    bool2             cont4        bool3             cont5
##  Mode :logical   Min.   : 0.0   Mode :logical   Min.   :   0
##  FALSE:395       1st Qu.: 0.0   FALSE:374       1st Qu.:  75
##  TRUE :295       Median : 0.0   TRUE :316       Median : 160
##                  Mean   : 2.4                   Mean   : 184
##                  3rd Qu.: 3.0                   3rd Qu.: 276
##                  Max.   :67.0                   Max.   :2000
##                                                 NA's   :13
##      cont6            approval          credit.score        ages
##  Min.   :     0.0   Length:690         Min.   :583.7   Min.   :17.00
##  1st Qu.:     0.0   Class :character   1st Qu.:666.7   1st Qu.:31.00
##  Median :     5.0   Mode  :character   Median :697.3   Median :38.00
##  Mean   :  1017.4                      Mean   :696.4   Mean   :39.59
##  3rd Qu.:   395.5                      3rd Qu.:726.4   3rd Qu.:47.00
##  Max.   :100000.0                      Max.   :806.0   Max.   :84.00
##
```

```
#changes the "approval" column from a string to a factor
df4$approval <- as.factor(df4$approval)
```

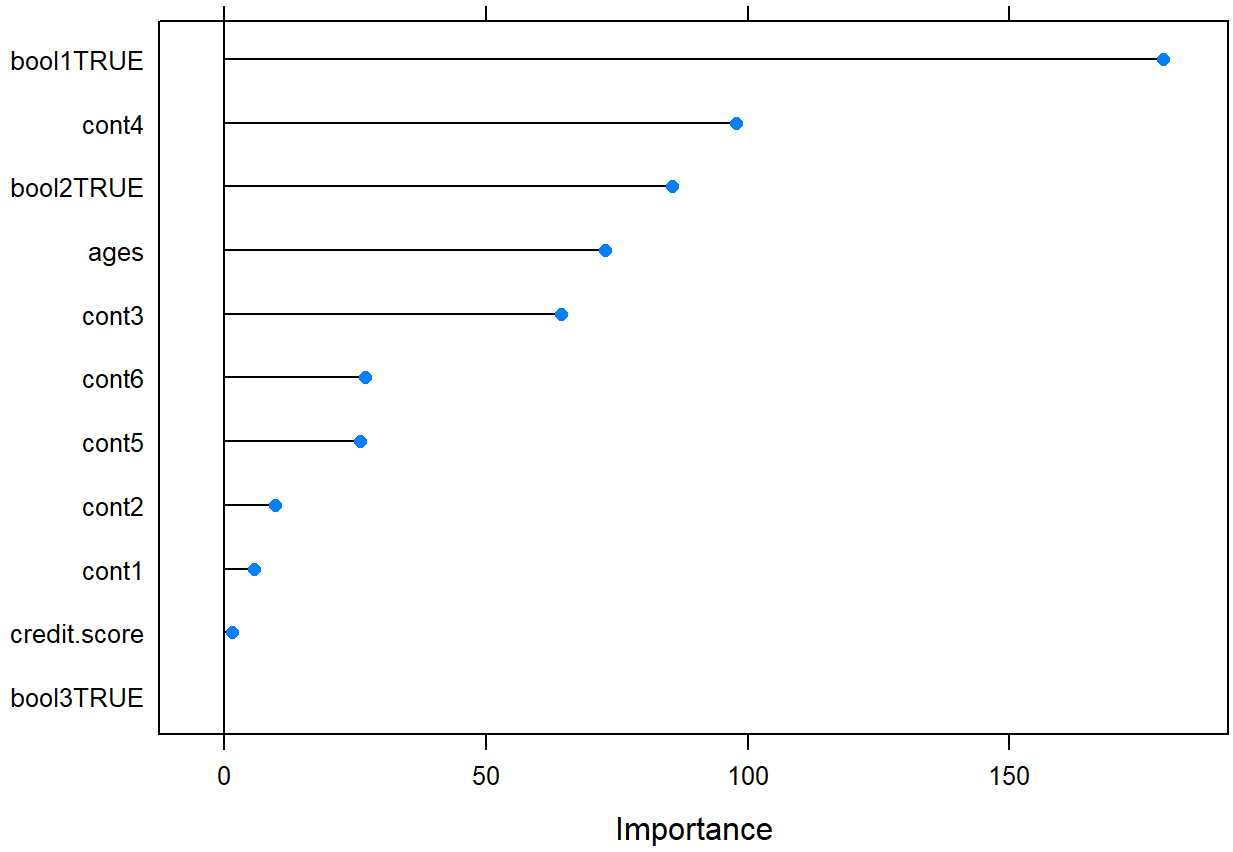
```
#builds initial tree with minsplit = 10 and maxdepth = 20

train_control = trainControl(method = "cv", number = 10)

tree4 <- train(approval ~., data = df4, control = rpart.control(minsplit =  10, maxdepth = 2
0), trControl = train_control, method = "rpart1SE",na.action=na.exclude)
tree4
```

```
## CART
##
## 690 samples
##  11 predictor
##   2 classes: '-', '+'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 599, 599, 599, 599, 600, 601, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8662965  0.7319701
```

```
#Run variable importance analysis

var_imp <- varImp(tree4, scale = FALSE)
var_imp
```

```
## rpart1SE variable importance
##
##              Overall
## bool1TRUE    179.282
## cont4         97.700
## bool2TRUE     85.622
## ages          72.800
## cont3         64.343
## cont6         26.828
## cont5         25.878
## cont2          9.620
## cont1          5.646
## credit.score   1.504
## bool3TRUE      0.000
```

```
#plots variable importance
plot(var_imp)
```



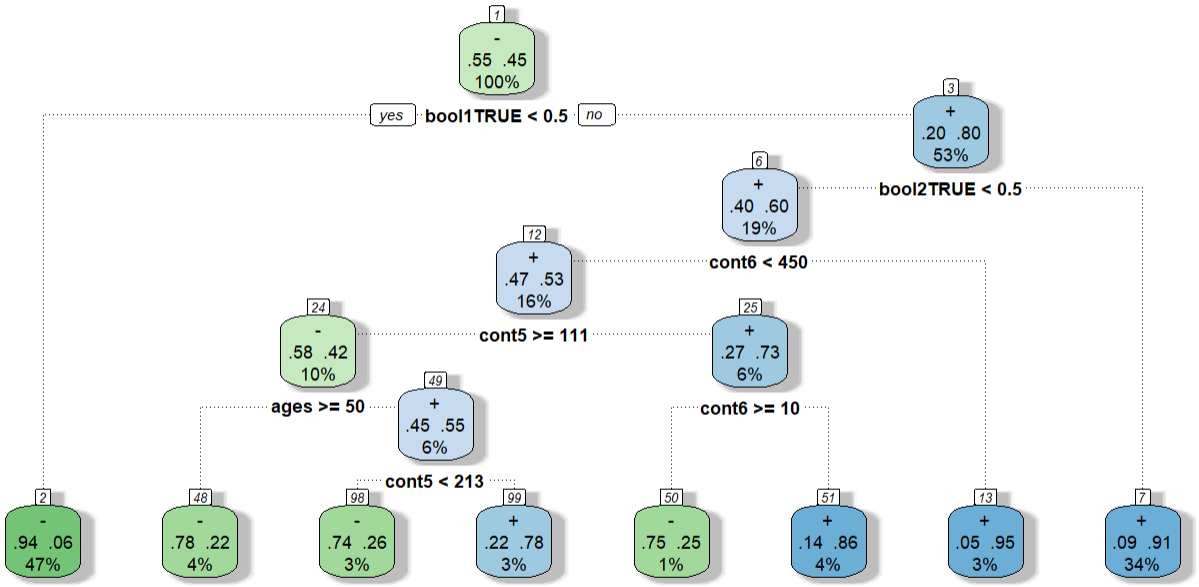

```
#removes all variables that are not part of the top 6 based on variable importance analysis

df4new <- df4
df4new <- df4new %>% select(-c( "cont5", "cont2", "cont1", "credit.score", "bool3"))
```

```
#buuilds new model based on only top 6 variables

train_control = trainControl(method = "cv", number = 10)

tree4new <- train(approval ~., data = df4new, control = rpart.control(minsplit =  10, maxdept
h = 20), trControl = train_control, method = "rpart1SE",na.action=na.exclude)
tree4new
```

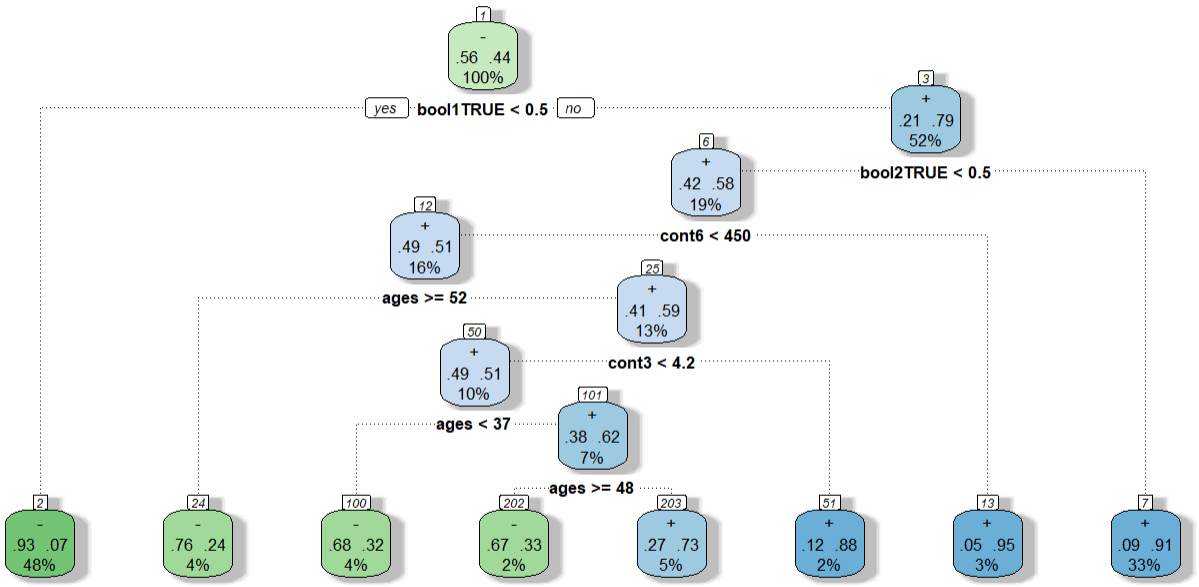
```
## CART
##
## 690 samples
##   6 predictor
##   2 classes: '-', '+'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 621, 621, 621, 621, 621, 621, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8696011  0.7349566
```

Accuracy of the original model had a score of 0.8755088 with a Kappa of 0.7507051. The accuracy slightly decreased when evaluating with only the top 6 variables to 0.8593588 with a kappa of 0.7163436

```
library(rattle)
#visualizes the original model

fancyRpartPlot(tree4$finalModel, caption = "")
```



```
#visualizes the new model

fancyRpartPlot(tree4new$finalModel, caption = "")
```



In this case, it doesn't seem that reducing the number of variables had an effect on the size of the tree.