Project 2.1

Code

Customer Churn

1 Read in the data

The following dataset will be used to predict `Churned` .

Hide

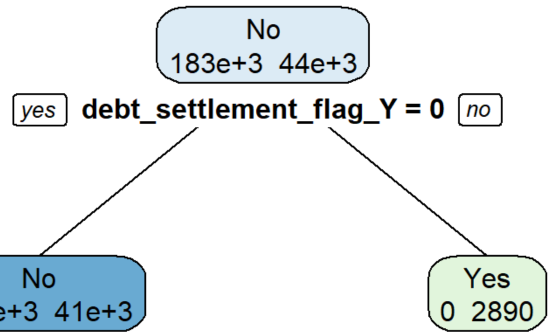
```
train=readRDS("group5AA_Black-Boopathy_train.rds")
holdout=readRDS("holdout_df.rds")
```

2 Fit Tree 1

Fit a decision tree to the training data using the default settings. Plot the tree.

Hide

```
options(scipen=999)

loan.ct1=rpart::rpart(loan_default ~ .,
                        data=train,
                        method="class")
rpart.plot::rpart.plot(loan.ct1,
                       extra=1,
                       fallen.leaves=FALSE)
```



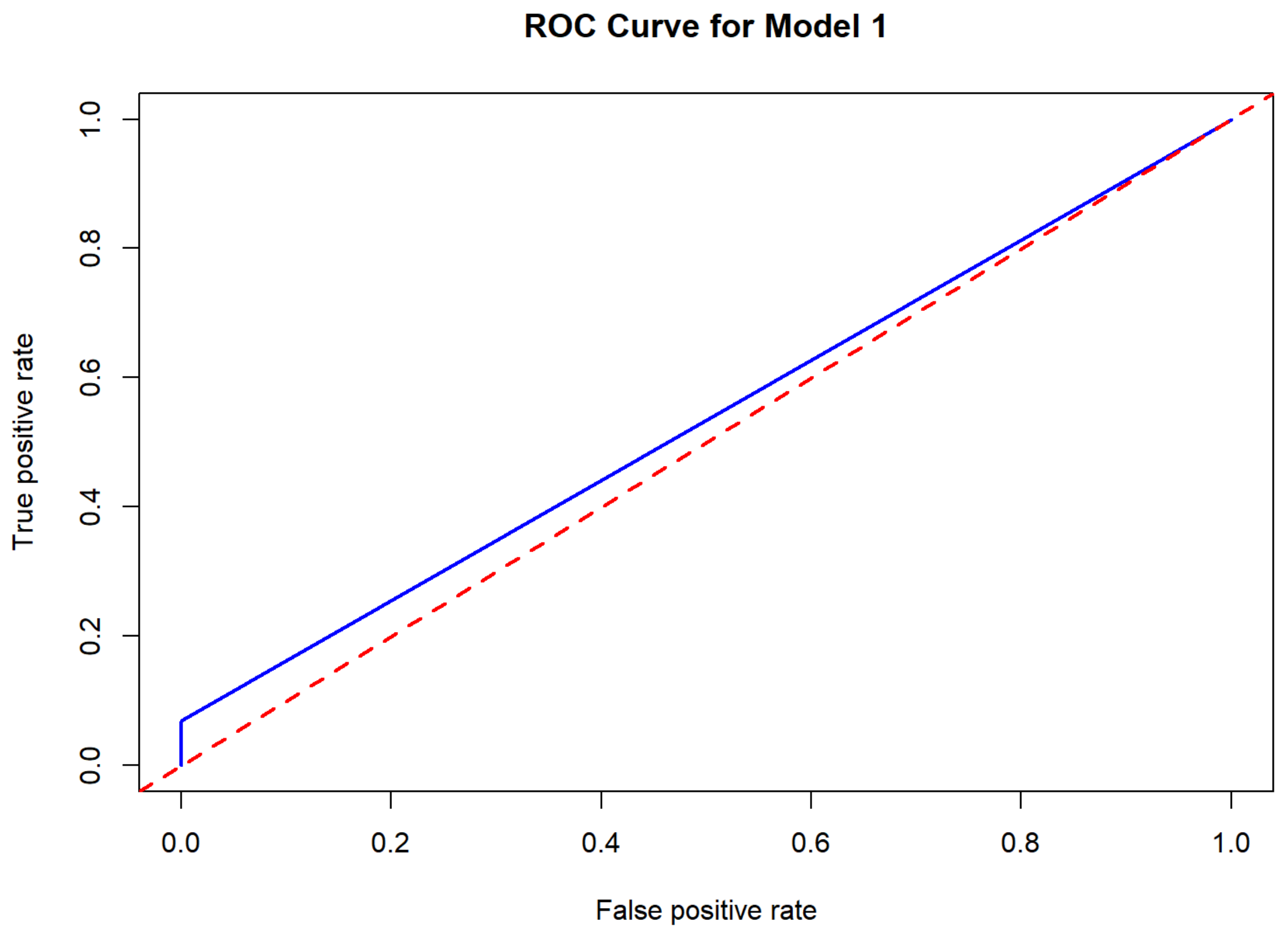
Hide

```
holdout$default.class <- predict(loan.ct1,
                                      newdata=holdout,
                                      type ="class"
                                      )
holdout$default.prob <- predict(loan.ct1,
                                      newdata=holdout,
                                      type ="prob"
                                      )[,"Yes"] #probability of "Yes"

confusionMatrix(holdout$default.class,
                holdout$loan_default,positive="Yes"


            )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  61026 13525
##        Yes     0  1004
##
##                Accuracy : 0.821
##                  95% CI : (0.8182, 0.8237)
##     No Information Rate : 0.8077
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.1071
##
##  Mcnemar's Test P-Value : < 0.00000000000000022
##
##             Sensitivity : 0.06910
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.81858
##              Prevalence : 0.19230
##          Detection Rate : 0.01329
##    Detection Prevalence : 0.01329
##       Balanced Accuracy : 0.53455
##
##        'Positive' Class : Yes
##
```

Hide

```
pred <- ROCR::prediction(holdout$default.prob,holdout$loan_default)
perf <- ROCR::performance(pred, "tpr", "fpr")

# Plot Curve
plot(perf, col = "blue", lwd = 2, main = "ROC Curve for Model 1")
abline(a = 0, b = 1, col = "red", lty = 2, lwd = 2)
```



Hide

```
# Find the precision and recall for each threshold
prec <- ROCR::performance(pred, "prec")
rec <- ROCR::performance(pred, "rec")

# Extract precision and recall
precision <- prec@y.values[[1]]
recall <- rec@y.values[[1]]

# Compute F1 score
f1=2*precision*recall/(precision+recall)
f1[is.nan(f1)]=0
head(f1)
```

```
## [1] 0.0000000 0.1292732 0.3225656
```

Hide

```
# Combine F1 scores and cutoffs into a data frame
cutoffs=prec@x.values[[1]]
df_f1=data.frame(f1,cutoffs)
head(df_f1)
```

	f1 <dbl>	cutoffs <dbl>
1	0.0000000	Inf
2	0.1292732	1.0000000
3	0.3225656	0.1816958

3 rows

Hide

```
# Find the optimal F1 index, F1 score, and cutoff
opt_idx <- which.max(f1)
opt_f1 <- df_f1[opt_idx,]
opt_f1
```

	f1 <dbl>	cutoffs <dbl>
3	0.3225656	0.1816958

1 row

Hide

```
holdout$defaultopt.class <- factor(ifelse(holdout$default.prob >= 0.1816958,
"Yes", "No"), levels = c("No", "Yes"))

confusionMatrix(holdout$defaultopt.class,
                holdout$loan_default,positive="Yes"



            )
```

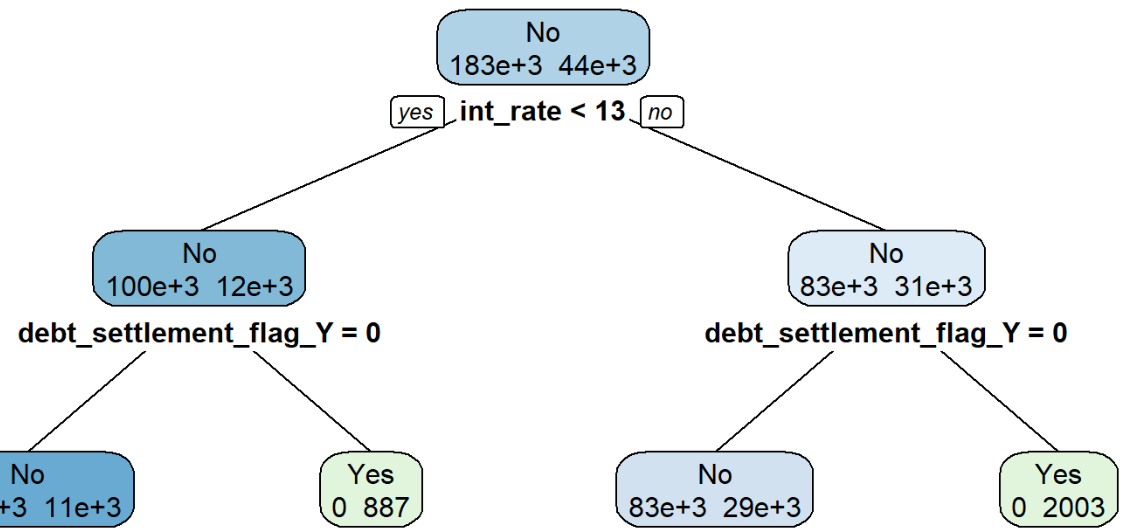
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  61026 13525
##        Yes     0  1004
##
##                Accuracy : 0.821
##                  95% CI : (0.8182, 0.8237)
##     No Information Rate : 0.8077
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.1071
##
##  Mcnemar's Test P-Value : < 0.00000000000000022
##
##             Sensitivity : 0.06910
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.81858
##              Prevalence : 0.19230
##          Detection Rate : 0.01329
##    Detection Prevalence : 0.01329
##       Balanced Accuracy : 0.53455
##
##        'Positive' Class : Yes
##
```

3 Fit Tree 2

Fit a decision tree to the training data. Make sure you are using *Entropy* as the impurity measure. Use `control=rpart.control(maxdepth=4,minbucket=10)` . Plot the tree.

Hide

```
loan.ct2=rpart::rpart(loan_default ~ .,
                        data=train,
                        method="class",
                                  parms=list(split="information"),
                    control=rpart.control(maxdepth=4,minbucket=10))

rpart.plot::rpart.plot(loan.ct2,
                       extra=1,
                       fallen.leaves=FALSE)
```



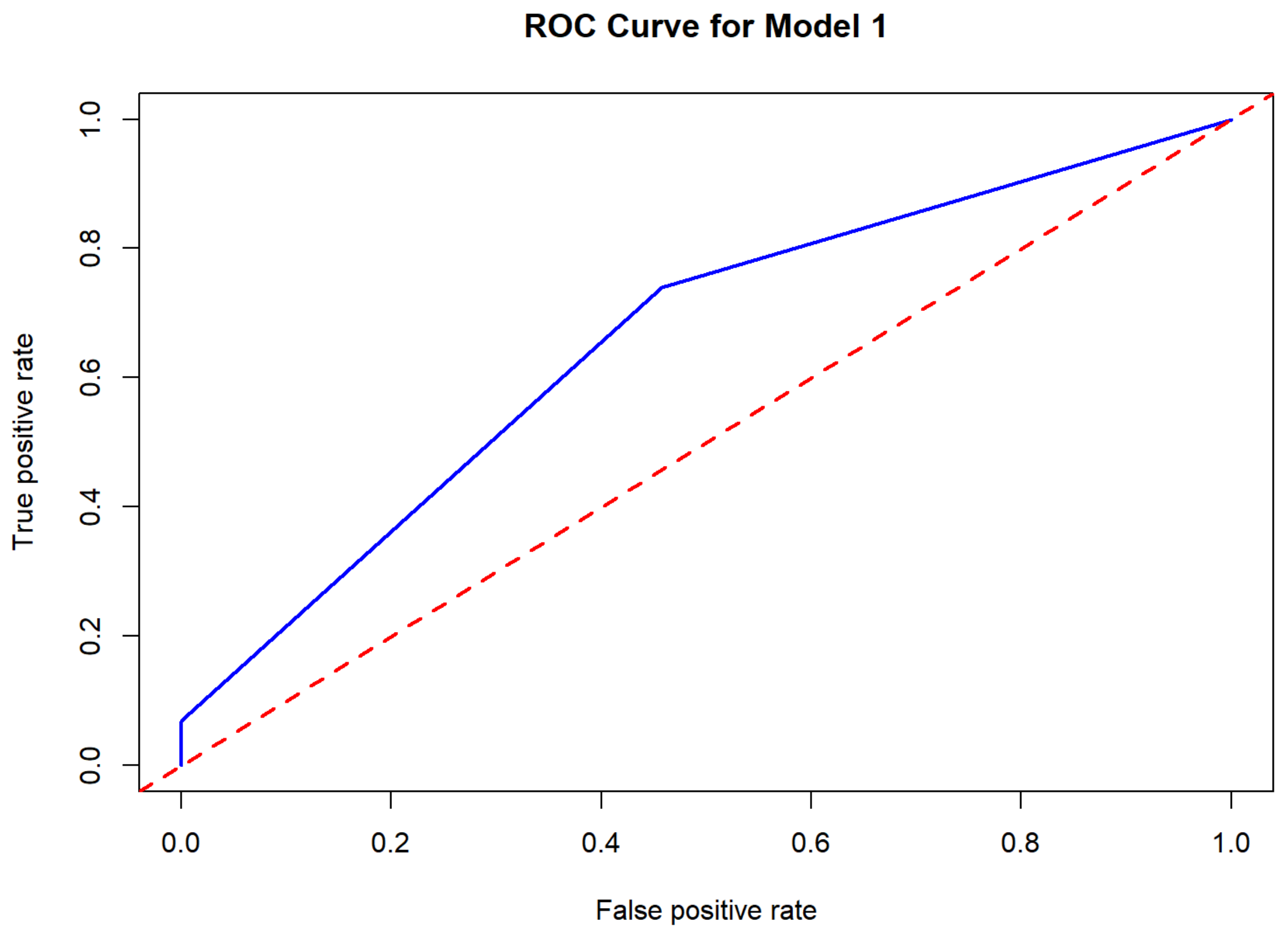
Hide

```
holdout$entropy.class <- predict(loan.ct2,
                                      newdata=holdout,
                                      type ="class"
                                      )
holdout$entropy.prob <- predict(loan.ct2,
                                      newdata=holdout,
                                      type ="prob"
                                      )[,"Yes"] #probability of "Yes"

# Create Confusion Matrix
confusionMatrix(holdout$entropy.class,
                holdout$loan_default,positive="Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  61026 13525
##        Yes     0  1004
##
##                Accuracy : 0.821
##                  95% CI : (0.8182, 0.8237)
##     No Information Rate : 0.8077
##     P-Value [Acc > NIR] : < 0.00000000000000022
##
##                   Kappa : 0.1071
##
##  Mcnemar's Test P-Value : < 0.00000000000000022
##
##             Sensitivity : 0.06910
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.81858
##              Prevalence : 0.19230
##          Detection Rate : 0.01329
##    Detection Prevalence : 0.01329
##       Balanced Accuracy : 0.53455
##
##        'Positive' Class : Yes
##
```

Hide

```
pred <- ROCR::prediction(holdout$entropy.prob,holdout$loan_default)
perf <- ROCR::performance(pred, "tpr", "fpr")

# Plot Curve
plot(perf, col = "blue", lwd = 2, main = "ROC Curve for Model 1")
abline(a = 0, b = 1, col = "red", lty = 2, lwd = 2)
```



Hide

```
# Find the precision and recall for each threshold
prec <- ROCR::performance(pred, "prec")
rec <- ROCR::performance(pred, "rec")

# Extract precision and recall
precision <- prec@y.values[[1]]
recall <- rec@y.values[[1]]

# Compute F1 score
f1=2*precision*recall/(precision+recall)
f1[is.nan(f1)]=0
head(f1)
```

```
## [1] 0.0000000 0.1292732 0.4039076 0.3225656
```

Hide

```
# Combine F1 scores and cutoffs into a data frame
cutoffs=prec@x.values[[1]]
df_f1=data.frame(f1,cutoffs)
head(df_f1)
```

	f1 <dbl>	cutoffs <dbl>
1	0.0000000	Inf
2	0.1292732	1.0000000
3	0.4039076	0.2603839
4	0.3225656	0.1016916

4 rows

Hide

```
# Find the optimal F1 index, F1 score, and cutoff
opt_idx <- which.max(f1)
opt_f1 <- df_f1[opt_idx,]
opt_f1
```

	f1 <dbl>	cutoffs <dbl>
3	0.4039076	0.2603839

1 row

Hide

```
holdout$entropyopt.class <- factor(ifelse(holdout$entropy.prob >= 0.2603839,
"Yes", "No"), levels = c("No", "Yes"))

confusionMatrix(holdout$entropyopt.class,
                holdout$loan_default,positive="Yes"


            )
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    No   Yes
##        No  61026 13525
##        Yes     0  1004
##                                                    
##                Accuracy : 0.821                    
##                  95% CI : (0.8182, 0.8237)         
##     No Information Rate : 0.8077                   
##     P-Value [Acc > NIR] : < 0.00000000000000022    
##                                                    
##                   Kappa : 0.1071                   
##                                                    
##  Mcnemar's Test P-Value : < 0.00000000000000022    
##                                                    
##             Sensitivity : 0.06910                  
##             Specificity : 1.00000                  
##          Pos Pred Value : 1.00000                  
##          Neg Pred Value : 0.81858                  
##              Prevalence : 0.19230                  
##          Detection Rate : 0.01329                  
##    Detection Prevalence : 0.01329                  
##       Balanced Accuracy : 0.53455                  
##                                                    
##        'Positive' Class : Yes                      
## 
```

4 Fit Tree 3

4.1 down sample

Find the holdout predictions for each tree. Use the `type="class"` argument to get the predicted class. Print the first few rows of the predictions.

Hide

```
table(train$loan_default)
```

```
## 
##     No    Yes 
## 182927  43507
```

Hide

```
train.us = downSample(x = train %>% select(-loan_default),
                      y = train$loan_default,
                      yname = "loan_default")
table(train.us$loan_default)
```

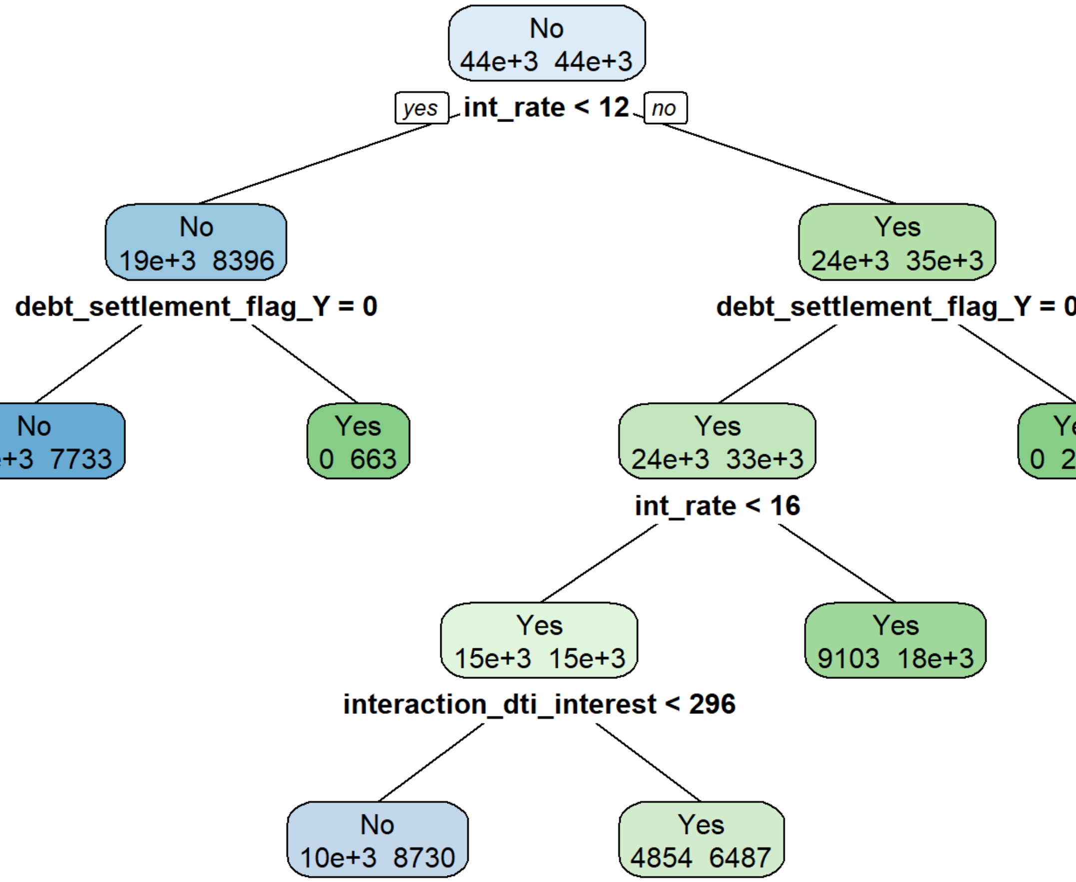
```
##
##    No   Yes
## 43507 43507
```

5 Tree 3

Use the oversampled training data to refit the model (default) used for Tree 1. Plot the tree.

Hide

```
loan.ct3=rpart::rpart(loan_default ~ .,
                        data=train.us,
                        method="class")
rpart.plot::rpart.plot(loan.ct3,
                       extra=1,
                       fallen.leaves=FALSE)
```



Hide

```
holdout$ostree.class <- predict(loan.ct3, newdata = holdout, type = "class")

holdout$ostree.prob <- predict(loan.ct3,
                              newdata=holdout,
                              type ="prob"
                              )[,"Yes"] #probability of "Yes"

confusionMatrix(holdout$ostree.class,
                holdout$loan_default,
                positive="Yes"
                )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  41521  5535
##        Yes 19505  8994
##
##                Accuracy : 0.6686
##                  95% CI : (0.6652, 0.6719)
##     No Information Rate : 0.8077
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2191
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.6190
##             Specificity : 0.6804
##          Pos Pred Value : 0.3156
##          Neg Pred Value : 0.8824
##              Prevalence : 0.1923
##          Detection Rate : 0.1190
##    Detection Prevalence : 0.3772
##       Balanced Accuracy : 0.6497
##
##        'Positive' Class : Yes
##
```

6 Tree 4

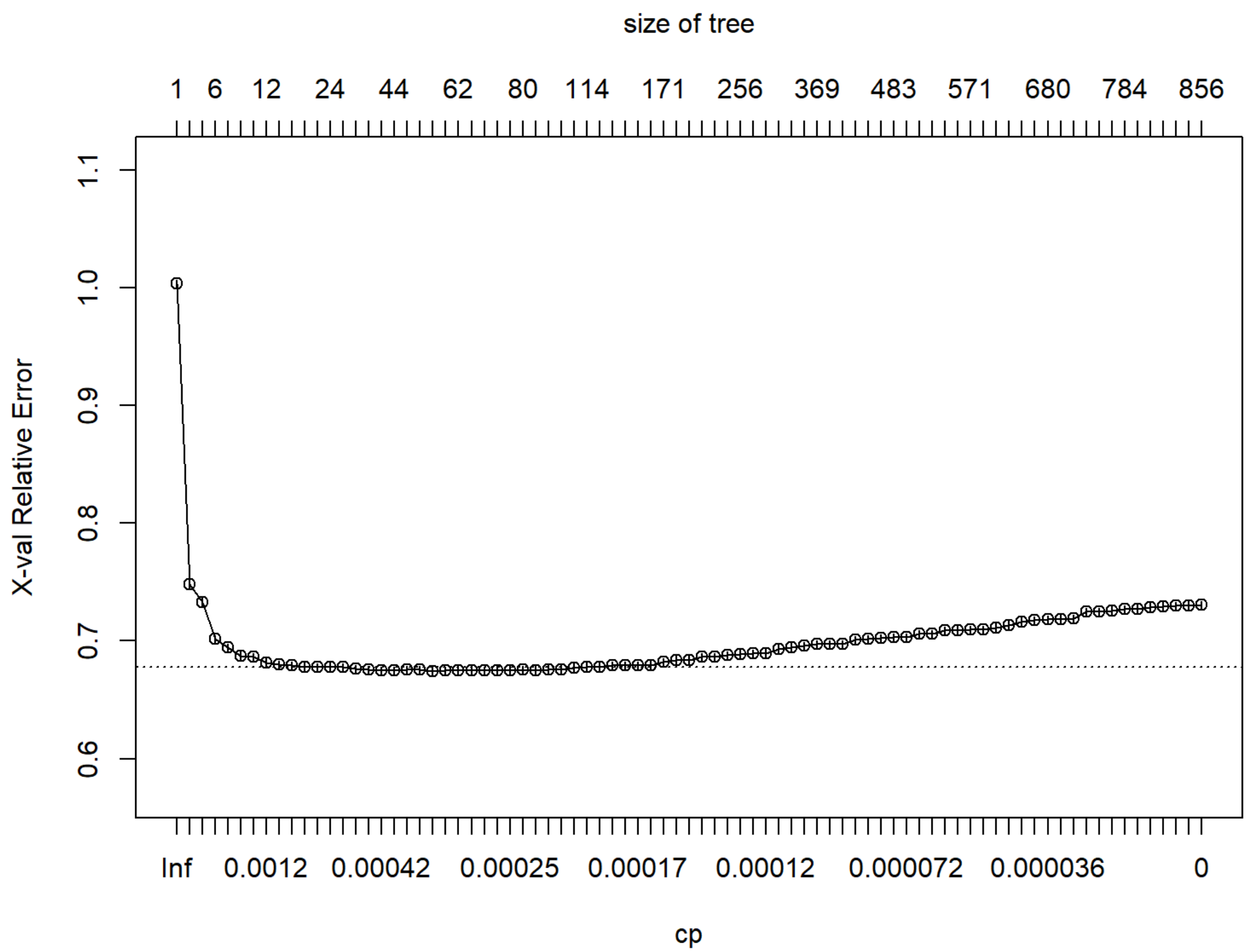
Use the under sampled training data, and 5-fold cross validation to fit the a tree.

6.1 Step 1

Use the rpart() function, adding the argument that controls for cross validation. Within the `rpart.control()` fucntion, set `cp=0` , xval=5 , `and` minbucket= `30` Print the `cp` results for each model size and a plot of the cross validation Be sure to use set.seed(123).

Hide

```
set.seed(123)
loan.ct4=rpart( loan_default~ .,
                data=train.us,
                method="class",
                control=rpart.control(cp=0,  minbucket=30, xval=5)
               )
plotcp(loan.ct4)
```



6.2 Step 2

Find the `cp` with the minimum cross-validation error. Print this value

Hide

```
best_cp=loan.ct4$cptable[which.min(loan.ct4$cptable[,"xerror"]), "CP"] #pick t
he tree with the lowest cross validated error
print(best_cp)
```

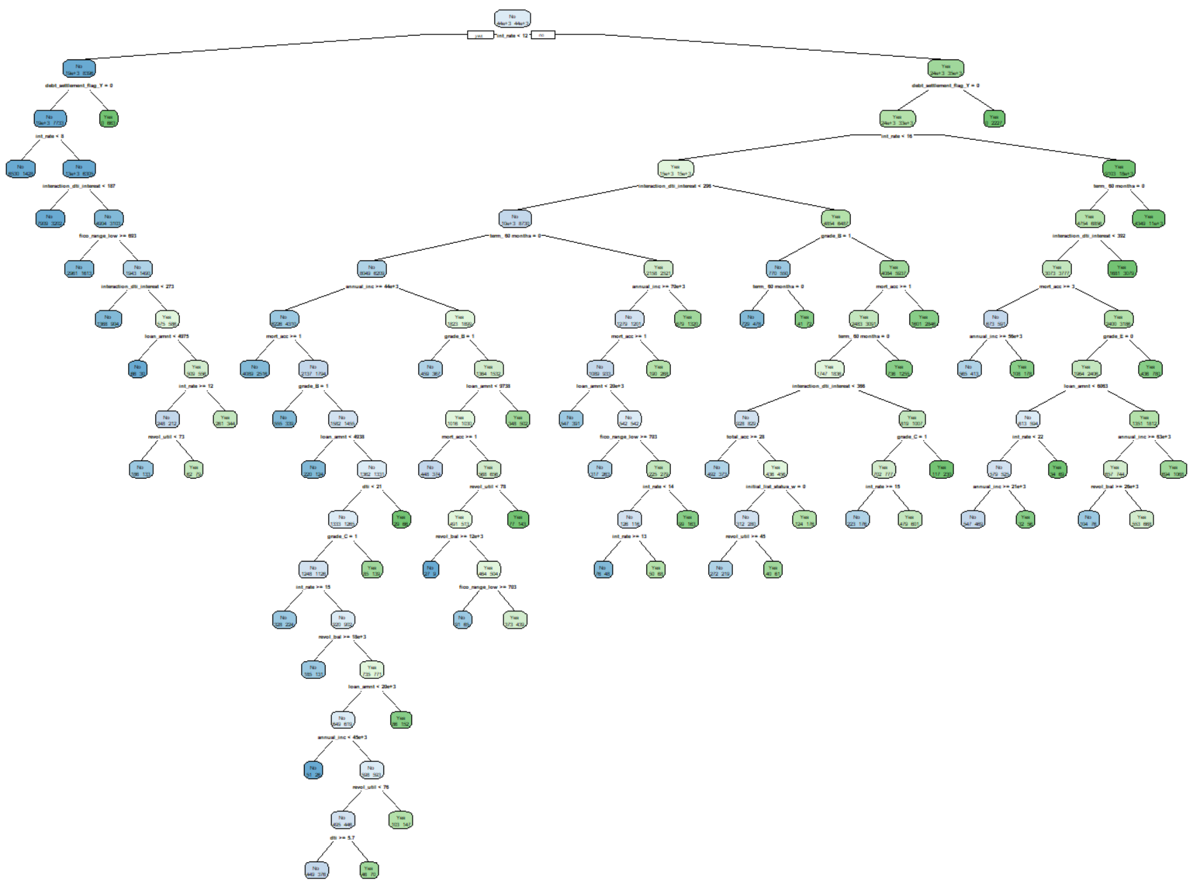
```
## [1] 0.0003064641
```

6.3 Step 3

Use the best `cp` value to prune the tree. Plot the pruned tree.

Hide

```
pruned_tree=prune(loan.ct4, cp=best_cp)
rpart.plot(pruned_tree, extra=1, fallen.leaves=FALSE)
```



6.4 Step 4

Use the pruned tree to make predictions on the holdout data. Print the first few rows of the predictions.

Hide

```
holdout=readRDS("holdout_df.rds")

holdout$pruned.class <- predict(pruned_tree,
                                     newdata=holdout,
                                     type ="class"
                                     )

holdout$pruned.prob <- predict(pruned_tree,
                                      newdata=holdout,
                                      type ="prob"
                                      )[,"Yes"] #probability of "Yes"
```

6.5 Step 5

Create a confusion matrix for the pruned tree using the holdout data. Print the confusion matrix.

Hide

```
confusionMatrix(holdout$pruned.class,
                holdout$loan_default,
                positive="Yes"
                )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  41212  5185
##        Yes 19814  9344
##
##                Accuracy : 0.6691
##                  95% CI : (0.6658, 0.6725)
##     No Information Rate : 0.8077
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2302
##
##  Mcnemar's Test P-Value : <0.0000000000000002
##
##             Sensitivity : 0.6431
##             Specificity : 0.6753
##          Pos Pred Value : 0.3205
##          Neg Pred Value : 0.8882
##              Prevalence : 0.1923
##          Detection Rate : 0.1237
##    Detection Prevalence : 0.3859
##       Balanced Accuracy : 0.6592
##
##        'Positive' Class : Yes
##
```

Hide

```
#saveRDS(holdout, "holdout_df_Singletree.rds")
```