Homework 5

Karan Ashar

3/16/2021

## Introduction

In this task we have to classify the papers which are labeled as ‘disputed’. The two output classes are ‘Hamilton’ and ‘Madison’. This is a classification problem. We are going to use the Decision Tree algorithm to solve this problem. First we will train our algorithm with the labeled data and then once the model is ready we will predict the class of the ‘disputed’ papers.

## Loading required libraries

library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.0.4

## Loading required package: rpart

library(rpart)  
library(splitTools)

## Warning: package 'splitTools' was built under R version 4.0.4

library(ranger)

## Warning: package 'ranger' was built under R version 4.0.4

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.4

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.0.6 v stringr 1.4.0  
## v tidyr 1.1.2 v forcats 0.5.1  
## v readr 1.4.0

## Warning: package 'readr' was built under R version 4.0.4

## Warning: package 'forcats' was built under R version 4.0.4

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(caret)

## Warning: package 'caret' was built under R version 4.0.4

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

set.seed(102)

## Loading data and pre-processing steps

After we load the data we will create a dataset call test\_disputed. This will contain all the papers which are marked as ‘disputed’. At the end we have to finally use our model to predict these papers. To train our model we then filter out the papers which belong to ‘Hamilton’ and ‘Madison’ and call it filtered\_data.

data<-read.csv(file ='HW4-data-fedPapers85.csv')  
  
test\_disputed=data[data$author=='dispt',]  
test\_disputed\_file\_name=test\_disputed$filename  
test\_disputed$filename<-NULL  
  
filtered\_data=data[data$author=='Hamilton' | data$author=='Madison',]  
filtered\_data\_file\_name=filtered\_data$filename  
filtered\_data$filename<-NULL

## Validation Set Approach

The first approach which we are going to use is Validation set approach. In this we split our training data into two parts - train and test. While splitting the data up we need to ensure that the target variable, in our case it is ‘author’ maintains it’s distribution. In other words, the ratio of occurrences of ‘Madison’ to ‘Hamilton’ remain similar across our train as well as our test split with respect to our original dataset.

So to summarize:   
train - It contains the data which we will train our model on.   
test - We will validate our model on this data to see it’s performace. This is the data the model will not see while training.   
test\_disputed - This dataset contains the data of the ‘disputed’ papers.

#Breaking down into train and validation set.  
inds <- partition(filtered\_data$author, p = c(train = 0.80, test = 0.20))  
  
train<-filtered\_data[inds$train,]  
test<-filtered\_data[inds$test,]  
print('Training Data distribution')

## [1] "Training Data distribution"

table(train$author)

##   
## Hamilton Madison   
## 40 12

print('Testing Data distribution')

## [1] "Testing Data distribution"

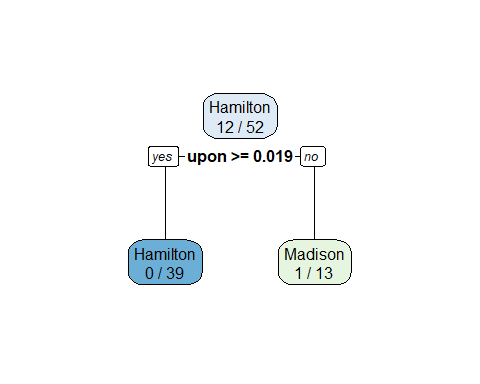
table(test$author)

##   
## Hamilton Madison   
## 11 3

**Note:**Interpreting the Trees in this report  
The label on the node indicates the class output for the observation if it ended on that node. The number’s below it indicate the miss-classified instances in that node while training. So something like 0/41 tells us that out of 41 observations 0 were miss-classified.

Base Model:   
Our model gets one observation wrong on the training data which can be observed from the Tree visualization. On the test data our model performs perfectly.  
On the disputed dataset, it classifies all papers as ‘Madison’.

fit <- rpart(author~., data = train, method = 'class')  
rpart.plot(fit,extra=3)



Prediction

predict\_unseen <-predict(fit, test[,c(-1)], type = 'class')  
  
print('Lets see our performance on the test set that we created')

## [1] "Lets see our performance on the test set that we created"

confusionMatrix(table(test$author,predict\_unseen))

## Confusion Matrix and Statistics  
##   
## predict\_unseen  
## Hamilton Madison  
## Hamilton 11 0  
## Madison 0 3  
##   
## Accuracy : 1   
## 95% CI : (0.7684, 1)  
## No Information Rate : 0.7857   
## P-Value [Acc > NIR] : 0.03417   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.7857   
## Detection Rate : 0.7857   
## Detection Prevalence : 0.7857   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Hamilton   
##

print('Prediction on disputed dataset')

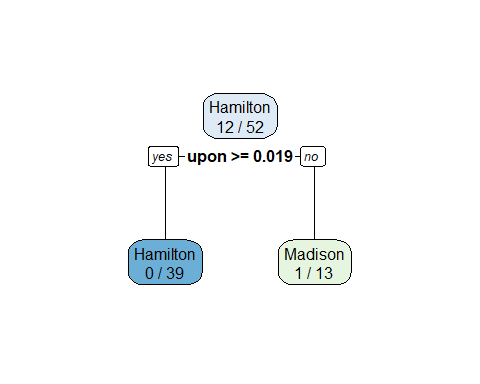
## [1] "Prediction on disputed dataset"

table(predict(fit,test\_disputed[,c(-1)],type='class'))

##   
## Hamilton Madison   
## 0 11

Changing parameters manually

fit <- rpart(author~., data = train, method = 'class',   
 minbucket =2,parms = list(split = 'information'),maxdepth= 5)  
rpart.plot(fit,extra=3)



Prediction on the validation set. It does pretty good. It’s the same performance as the base model.

predict\_unseen <-predict(fit, test[,c(-1)], type = 'class')  
confusionMatrix(table(test$author,predict\_unseen))

## Confusion Matrix and Statistics  
##   
## predict\_unseen  
## Hamilton Madison  
## Hamilton 11 0  
## Madison 0 3  
##   
## Accuracy : 1   
## 95% CI : (0.7684, 1)  
## No Information Rate : 0.7857   
## P-Value [Acc > NIR] : 0.03417   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.7857   
## Detection Rate : 0.7857   
## Detection Prevalence : 0.7857   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Hamilton   
##

Prediction on the on data with disputed papers.

table(predict(fit,test\_disputed[,c(-1)],type='class'))

##   
## Hamilton Madison   
## 0 11

**Note:**   
The model we are using is the ‘rpart’ model. Let’s look at its various hyperparameters.

modelLookup('rpart')

## model parameter label forReg forClass probModel  
## 1 rpart cp Complexity Parameter TRUE TRUE TRUE

## 

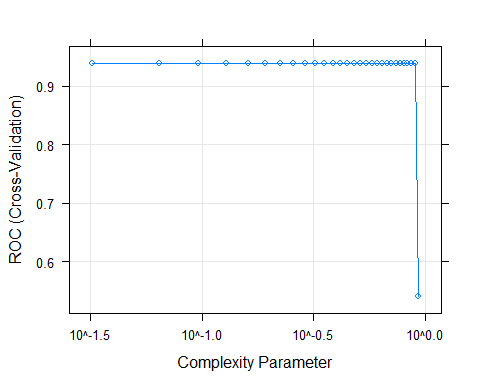
## K-fold cross validation approach

The problem with our previous approach is that the random split that we did, could give us different answers for a split that we did everytime. The model that we just ran could perform different on another split. We could be lucky in our first split. This is not a right way to conclude results. So the K-fold cross validation approach solves this problem. Here the K we have selected is ‘10’. This means the dataset will be divided into 10 parts where each of the 9 parts would be used for training and the last part would be used for testing.  
Here we also try different values for our hyperparameter ‘cp’. ‘cp’ stands for cost-complexity. It is a penalty applied with respect to the complexity of our tree. A low value of cp would indicate a very complex tree. This process would also tell us the best value for ‘cp’ as well. This is done by the tuneLength parameter in the train function. in our case it is equal to 30. So, 30 values for ‘cp’ will be tried. We use the metric ‘ROC’ to determine the best model.

train\_control <- trainControl(method = "cv",   
 number = 10,  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE)  
model <- train(author~.,data=filtered\_data,   
 method = "rpart",   
 trControl = train\_control,  
 tuneLength=30,  
 metric = "ROC"  
 )  
print(model)

## CART   
##   
## 66 samples  
## 70 predictors  
## 2 classes: 'Hamilton', 'Madison'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 59, 59, 58, 60, 59, 60, ...   
## Resampling results across tuning parameters:  
##   
## cp ROC Sens Spec  
## 0.00000000 0.94 0.98 0.9   
## 0.03218391 0.94 0.98 0.9   
## 0.06436782 0.94 0.98 0.9   
## 0.09655172 0.94 0.98 0.9   
## 0.12873563 0.94 0.98 0.9   
## 0.16091954 0.94 0.98 0.9   
## 0.19310345 0.94 0.98 0.9   
## 0.22528736 0.94 0.98 0.9   
## 0.25747126 0.94 0.98 0.9   
## 0.28965517 0.94 0.98 0.9   
## 0.32183908 0.94 0.98 0.9   
## 0.35402299 0.94 0.98 0.9   
## 0.38620690 0.94 0.98 0.9   
## 0.41839080 0.94 0.98 0.9   
## 0.45057471 0.94 0.98 0.9   
## 0.48275862 0.94 0.98 0.9   
## 0.51494253 0.94 0.98 0.9   
## 0.54712644 0.94 0.98 0.9   
## 0.57931034 0.94 0.98 0.9   
## 0.61149425 0.94 0.98 0.9   
## 0.64367816 0.94 0.98 0.9   
## 0.67586207 0.94 0.98 0.9   
## 0.70804598 0.94 0.98 0.9   
## 0.74022989 0.94 0.98 0.9   
## 0.77241379 0.94 0.98 0.9   
## 0.80459770 0.94 0.98 0.9   
## 0.83678161 0.94 0.98 0.9   
## 0.86896552 0.94 0.98 0.9   
## 0.90114943 0.94 0.98 0.9   
## 0.93333333 0.54 0.98 0.1   
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.9011494.

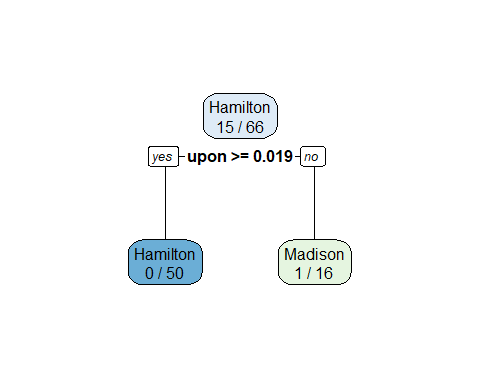
plot(model, scales = list(x = list(log = 10)))



We can observe the best ROC metric to be 0.94 for ‘cp’=0.9011494.

Lets visualize the best model via a tree structure. We observe that during training one example is miss-classified.

rpart.plot(model$finalModel,extra=3)



Prediction on the disputed papers All the disputed papers are classified as ‘Madison’.

table(predict(model,test\_disputed[,c(-1)]))

##   
## Hamilton Madison   
## 0 11

Let’s take a look at the joint authorship papers Here we can see all the joint authorship papers are also classified as ‘Madison’ This result is similar to our Clustering task as well

predict(fit,data[data$author=='HM',][,c(-1,-2)])

## Hamilton Madison  
## 63 0.07692308 0.9230769  
## 64 0.07692308 0.9230769  
## 65 0.07692308 0.9230769

## 

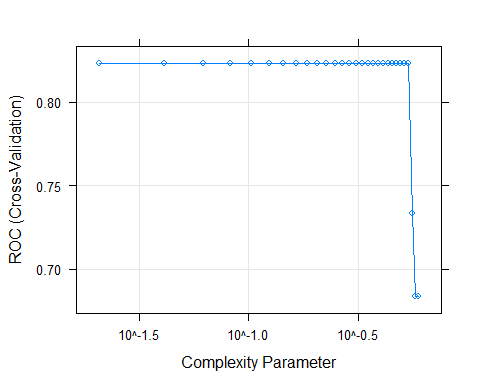
## Another experiment

I thought we should try the cross-validation process without the word ‘upon’ and see if our model performs better. This is because I see this feature being really strong and wanted to observe the results without it.

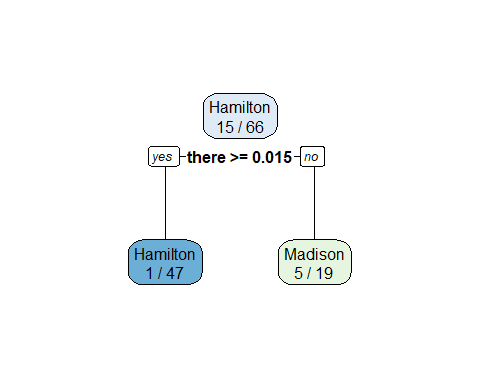
filtered\_data$upon<-NULL  
test\_disputed$upon<-NULL  
  
train\_control <- trainControl(method = "cv",   
 number = 10,  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE)  
  
  
  
model <- train(author~.,data=filtered\_data,   
 method = "rpart",   
 trControl = train\_control,  
 tuneLength=30,  
 metric = "ROC"  
)  
print(model)

## CART   
##   
## 66 samples  
## 69 predictors  
## 2 classes: 'Hamilton', 'Madison'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 58, 60, 60, 59, 59, 60, ...   
## Resampling results across tuning parameters:  
##   
## cp ROC Sens Spec  
## 0.00000000 0.8233333 0.8466667 0.8   
## 0.02068966 0.8233333 0.8466667 0.8   
## 0.04137931 0.8233333 0.8466667 0.8   
## 0.06206897 0.8233333 0.8466667 0.8   
## 0.08275862 0.8233333 0.8466667 0.8   
## 0.10344828 0.8233333 0.8466667 0.8   
## 0.12413793 0.8233333 0.8466667 0.8   
## 0.14482759 0.8233333 0.8466667 0.8   
## 0.16551724 0.8233333 0.8466667 0.8   
## 0.18620690 0.8233333 0.8466667 0.8   
## 0.20689655 0.8233333 0.8466667 0.8   
## 0.22758621 0.8233333 0.8466667 0.8   
## 0.24827586 0.8233333 0.8466667 0.8   
## 0.26896552 0.8233333 0.8466667 0.8   
## 0.28965517 0.8233333 0.8466667 0.8   
## 0.31034483 0.8233333 0.8466667 0.8   
## 0.33103448 0.8233333 0.8466667 0.8   
## 0.35172414 0.8233333 0.8466667 0.8   
## 0.37241379 0.8233333 0.8466667 0.8   
## 0.39310345 0.8233333 0.8466667 0.8   
## 0.41379310 0.8233333 0.8466667 0.8   
## 0.43448276 0.8233333 0.8466667 0.8   
## 0.45517241 0.8233333 0.8466667 0.8   
## 0.47586207 0.8233333 0.8466667 0.8   
## 0.49655172 0.8233333 0.8466667 0.8   
## 0.51724138 0.8233333 0.8466667 0.8   
## 0.53793103 0.8233333 0.8466667 0.8   
## 0.55862069 0.7333333 0.8666667 0.6   
## 0.57931034 0.6833333 0.8666667 0.5   
## 0.60000000 0.6833333 0.8666667 0.5   
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.537931.

plot(model, scales = list(x = list(log = 10)))



#table(test$author,predict(model,test))  
rpart.plot(model$finalModel,extra=3)



table(predict(model,test\_disputed[,c(-1)]))

##   
## Hamilton Madison   
## 5 6

We observe the best ROC metric reach 0.82. This is less than what we had got earlier. Also on the disputed dataset we observe changes. We can say it would be better if we preserved the feature word ‘upon’.

## Conclusion

We can observe that the classification task produces similar results as the clustering task. We take the model created with the K-fold validation approach to be our primary model along the value of ‘cp’ we found. We will also use the word ‘upon’ as one of our features. We can say it is an important feature to classify the papers written by ‘Hamilton’ and ‘Madison’.