Parin Patel

Use Decision Tree’s to Solve a Mystery in History

Introduction:

In the late 1780’s, to address the U.S Constitution still needing approval from nine of the thirteen states, The Federalist Papers, a series of 85-letters, were written in newspapers urging the ratification of the U.S Constitution. These letters contended that the US Constitution would preserve the Union and allow for the proposed federal government to act in the nations interest. The Federalist Papers have long been regarded as an extremely important source of evidence of the underlying meaning and intentions of the Constitution, as well. In fact, in 1821, in *Cohens v Virginia* Chief Justice John Marshall described the papers as the following:

> “complete commentary on our constitution; and its appealed to by all parties in the questions to which that instrument has given birth. Its intrinsic merit entitles it to this high rank, and the part two of its authors performed in framing the constitution, put it very much in their power to explain the views with which it was framed.”

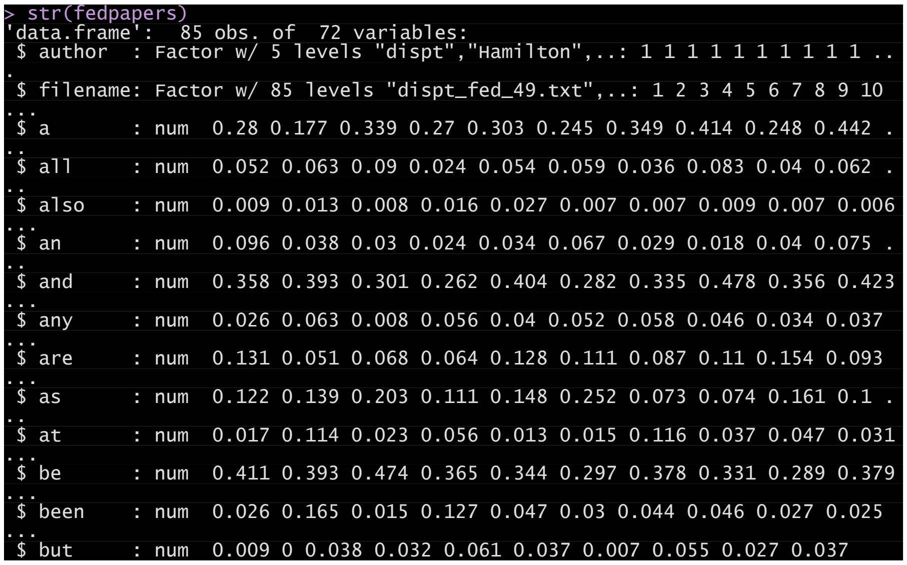
The predicament is, for a document with this caliber of importance in creating and ratifying our country’s’ Constitution, we do not know completely who wrote all the essays that make up the Federalist Papers. In fact, 11 essays are still disputed about regarding whether Madison or Hamilton wrote them.

This past week, we drew a conclusion on who wrote the disputed essays using clustering. This time, we will use decisions trees to answer this question instead.

Analysis:

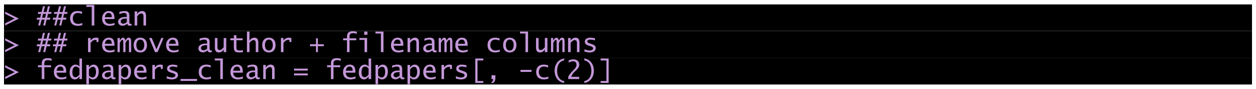
Section 1: Data preparation

Data Preparation first required loading the .csv file as “fedpapers”. Next, the file was generally assessed using str(fedpapers) (Figure 1). We then removed the first two columns (author and filename) in order to leave the data frame as just function words and feature values (Figure 2).



```
> str(fedpapers)
'data.frame':	85 obs. of  72 variables:
 $ author  : Factor w/ 5 levels "dispt","Hamilton",..: 1 1 1 1 1 1 1 1 1 1 ..
.
 $ filename: Factor w/ 85 levels "dispt_fed_49.txt",..: 1 2 3 4 5 6 7 8 9 10
...
 $ a       : num  0.28 0.177 0.339 0.27 0.303 0.245 0.349 0.414 0.248 0.442 .
..
 $ all     : num  0.052 0.063 0.09 0.024 0.054 0.059 0.036 0.083 0.04 0.062 .
..
 $ also    : num  0.009 0.013 0.008 0.016 0.027 0.007 0.007 0.009 0.007 0.006
...
 $ an      : num  0.096 0.038 0.03 0.024 0.034 0.067 0.029 0.018 0.04 0.075 .
..
 $ and     : num  0.358 0.393 0.301 0.262 0.404 0.282 0.335 0.478 0.356 0.423
...
 $ any     : num  0.026 0.063 0.008 0.056 0.04 0.052 0.058 0.046 0.034 0.037
...
 $ are     : num  0.131 0.051 0.068 0.064 0.128 0.111 0.087 0.11 0.154 0.093
...
 $ as      : num  0.122 0.139 0.203 0.111 0.148 0.252 0.073 0.074 0.161 0.1 .
..
 $ at      : num  0.017 0.114 0.023 0.056 0.013 0.015 0.116 0.037 0.047 0.031
...
 $ be      : num  0.411 0.393 0.474 0.365 0.344 0.297 0.378 0.331 0.289 0.379
...
 $ been    : num  0.026 0.165 0.015 0.127 0.047 0.03 0.044 0.046 0.027 0.025
...
 $ but     : num  0.009 0 0.038 0.032 0.061 0.037 0.007 0.055 0.027 0.037
```

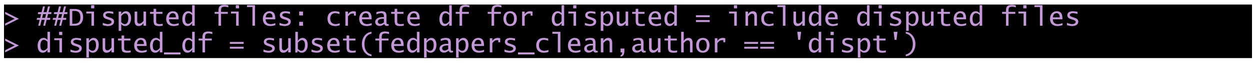
Figure 1



```
> ##clean
> ## remove author + filename columns
> fedpapers_clean = fedpapers[, -c(2)]
```

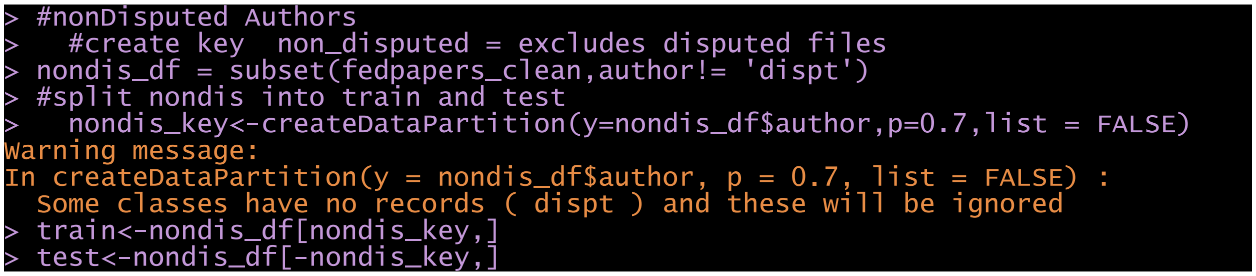
Figure 2

Next, we start preparing our data frames for the experiment. First, we established and created a disputed authors data frame that will be the prediction target for the decision tree model (Figure 3). We then created a not disputed authors data frame by exluding all rows where author is considered disputed. This not disputed data frame was then used to create our training and test datasets. These data sets were split using CreateDataPartion where 1/3 of the data was split into the test file and 2/3 was split into the train file. (Figure 4). Additionally, after parsing the data frame after the creation of the train and test files, we decided to remove all rows where the author was “HM”. We added this step to the Clean section in our code, and re-ran all below lines (figure 5).



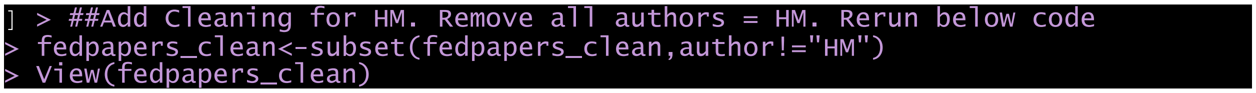
```
> ##Disputed files: create df for disputed = include disputed files
> disputed_df = subset(fedpapers_clean,author == 'dispt')
```

Figure 3



```
> #nonDisputed Authors
>   #create key  non_disputed = excludes disputed files
> nondis_df = subset(fedpapers_clean,author!= 'dispt')
> #split nondis into train and test
>   nondis_key<-createDataPartition(y=nondis_df$author,p=0.7,list = FALSE)
Warning message:
In createDataPartition(y = nondis_df$author, p = 0.7, list = FALSE) :
  Some classes have no records ( dispt ) and these will be ignored
> train<-nondis_df[nondis_key,]
> test<-nondis_df[-nondis_key,]
```

Figure 4

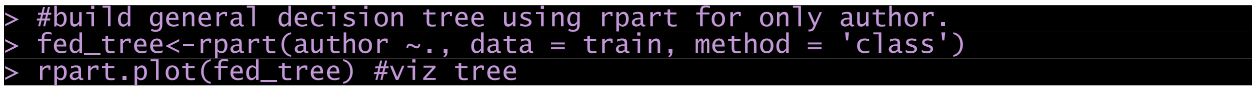


```
] > ##Add Cleaning for HM. Remove all authors = HM. Rerun below code
> fedpapers_clean<-subset(fedpapers_clean,author!="HM")
> View(fedpapers_clean)
```

Figure 5

Section 2: Build and Tune Decision Tree Models

We started by creating a general decision tree model using rpart on our train data for the variable authors. We set our method to "class" since our Y variable is a factor, and therefore the top node will always be 1 in this situation (Figure 6). We then used rpart.plot to visualize the tree (Figure 7). We then inspected our first general tree (Figure 8) using techniques like printcp and plotcp to visualize the cross-validation results (Figure 9). This helped us tune our tree by estimating how accurately the model generalized the unseen data. Note we will analyze and compare all models in the end .



```
> #build general decision tree using rpart for only author.
> fed_tree<-rpart(author ~., data = train, method = 'class')
> rpart.plot(fed_tree) #viz tree
```

Figure 6

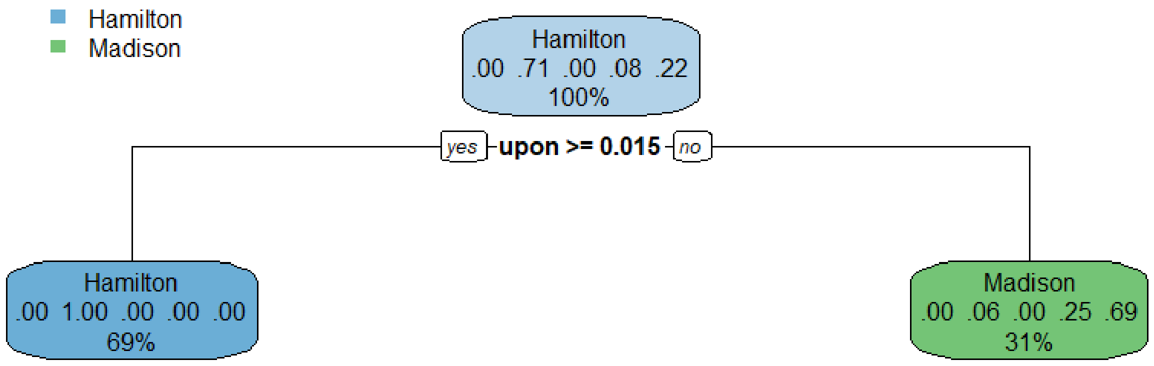
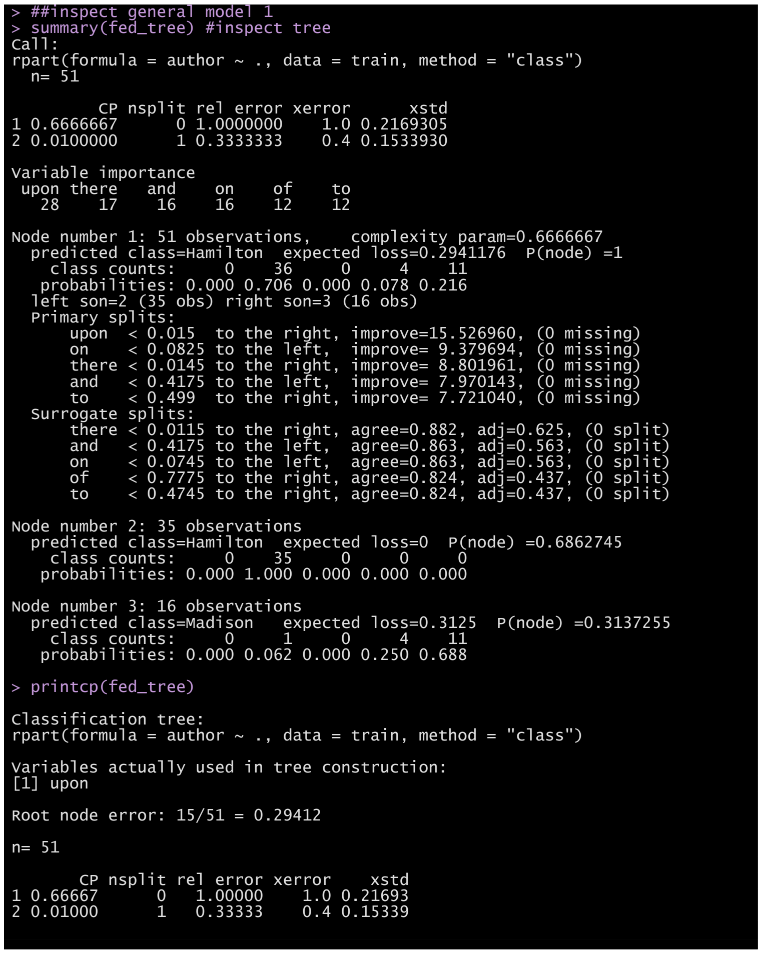


Figure 7



```
> ##inspect general model 1
> summary(fed_tree) #inspect tree
Call:
rpart(formula = author ~ ., data = train, method = "class")
  n= 51

         CP nsplit rel error xerror      xstd
1 0.6666667      0 1.0000000    1.0 0.2169305
2 0.0100000      1 0.3333333    0.4 0.1533930

Variable importance
 upon there   and    on    of    to 
   28    17    16    16    12    12 

Node number 1: 51 observations,    complexity param=0.6666667
  predicted class=Hamilton  expected loss=0.2941176  P(node) =1
    class counts:     0    36     0     4    11
   probabilities: 0.000 0.706 0.000 0.078 0.216 
  left son=2 (35 obs) right son=3 (16 obs)
  Primary splits:
      upon  < 0.015  to the right, improve=15.526960, (0 missing)
      on    < 0.0825 to the left,  improve= 9.379694, (0 missing)
      there < 0.0145 to the right, improve= 8.801961, (0 missing)
      and   < 0.4175 to the left,  improve= 7.970143, (0 missing)
      to    < 0.499  to the right, improve= 7.721040, (0 missing)
  Surrogate splits:
      there < 0.0115 to the right, agree=0.882, adj=0.625, (0 split)
      and   < 0.4175 to the left,  agree=0.863, adj=0.563, (0 split)
      on    < 0.0745 to the left,  agree=0.863, adj=0.563, (0 split)
      of    < 0.7775 to the right, agree=0.824, adj=0.437, (0 split)
      to    < 0.4745 to the right, agree=0.824, adj=0.437, (0 split)

Node number 2: 35 observations
  predicted class=Hamilton  expected loss=0  P(node) =0.6862745
    class counts:     0    35     0     0     0
   probabilities: 0.000 1.000 0.000 0.000 0.000 

Node number 3: 16 observations
  predicted class=Madison   expected loss=0.3125  P(node) =0.3137255
    class counts:     0     1     0     4    11
   probabilities: 0.000 0.062 0.000 0.250 0.688 

> printcp(fed_tree)

Classification tree:
rpart(formula = author ~ ., data = train, method = "class")

Variables actually used in tree construction:
[1] upon

Root node error: 15/51 = 0.29412

n= 51 

       CP nsplit rel error xerror    xstd
1 0.66667      0   1.00000    1.0 0.21693
2 0.01000      1   0.33333    0.4 0.15339
```

Figure 8

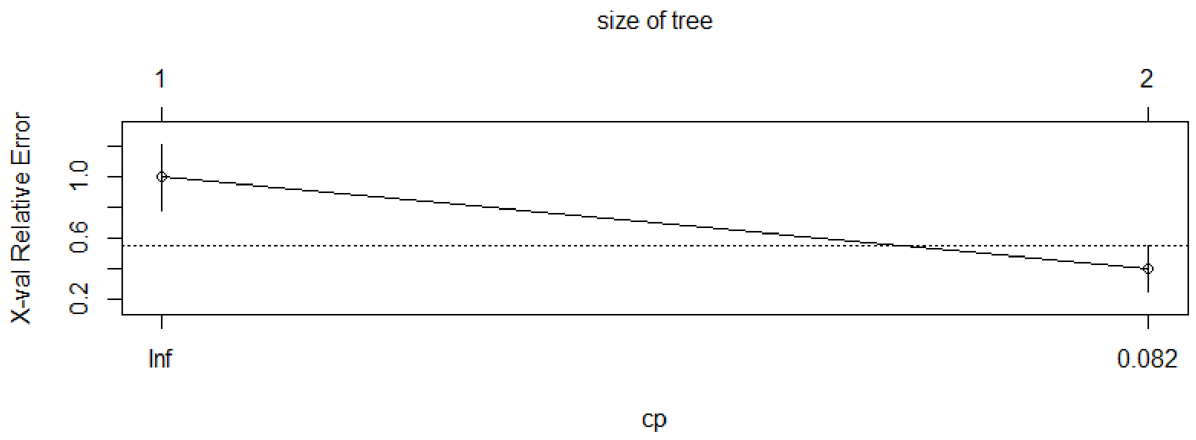
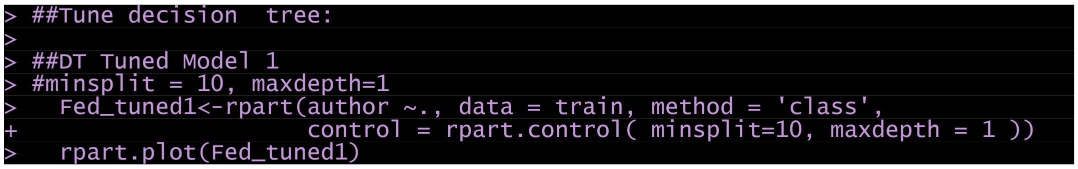


Figure 9

Based on the results of the inspection, which we will discuss in the end, we tuned our model. The first tuned model was adjusted to have a minsplit of 1 and a maxdepth of 1 (Figure 10 & 11). Our goal during inspection is to find the optimal model by using the plotcp table as our measure. Basically, we want to depict the deviation until the minimum value is reached. We will do this by adjusting the rpart control parameters to plot the cp value against the geometric mean (Figure 12 and 13).



```
> ##Tune decision  tree:
>
> ##DT Tuned Model 1
> #minsplit = 10, maxdepth=1
>   Fed_tuned1<-rpart(author ~., data = train, method = 'class',
+                     control = rpart.control( minsplit=10, maxdepth = 1 ))
>   rpart.plot(Fed_tuned1)
```

Figure 10

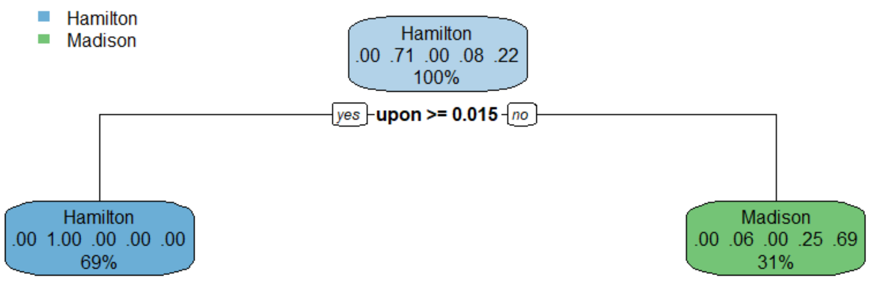


Figure 11

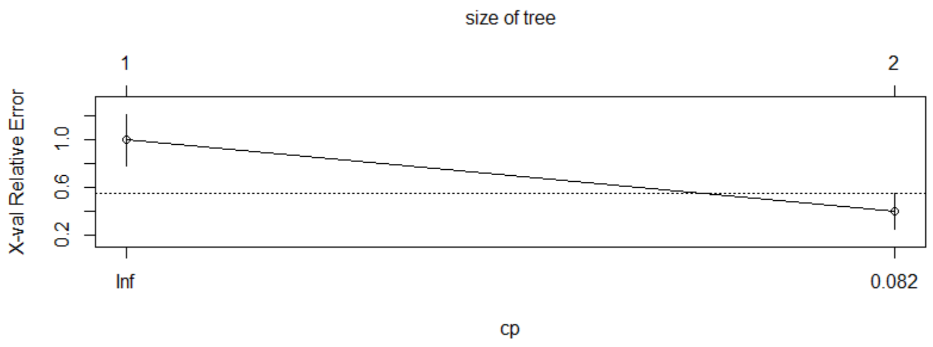
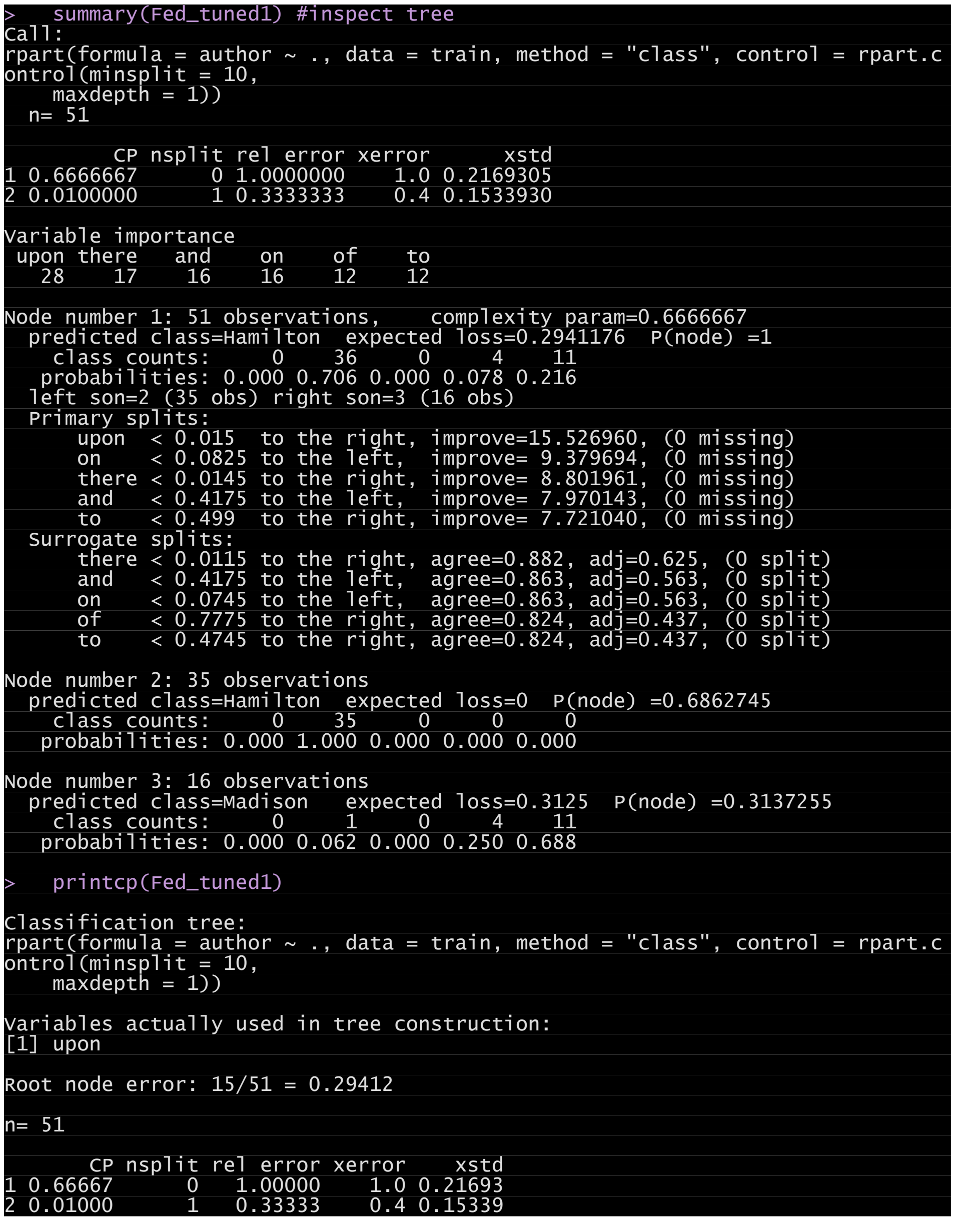


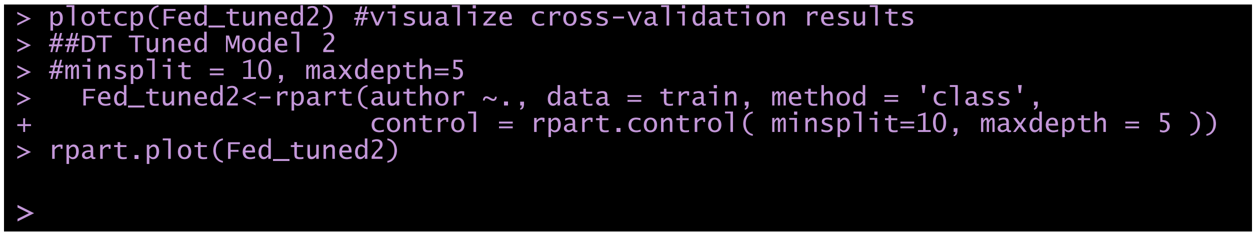
Figure 12



```
>   summary(Fed_tuned1) #inspect tree
Call:
rpart(formula = author ~ ., data = train, method = "class", control = rpart.c
ontrol(minsplit = 10,
    maxdepth = 1))
  n= 51

         CP nsplit rel error xerror      xstd
1 0.6666667      0 1.0000000    1.0 0.2169305
2 0.0100000      1 0.3333333    0.4 0.1533930

Variable importance
 upon there   and    on    of    to
   28    17    16    16    12    12

Node number 1: 51 observations,    complexity param=0.6666667
  predicted class=Hamilton  expected loss=0.2941176  P(node) =1
    class counts:     0    36     0     4    11
   probabilities: 0.000 0.706 0.000 0.078 0.216
  left son=2 (35 obs) right son=3 (16 obs)
  Primary splits:
      upon  < 0.015  to the right, improve=15.526960, (0 missing)
      on    < 0.0825 to the left,  improve= 9.379694, (0 missing)
      there < 0.0145 to the right, improve= 8.801961, (0 missing)
      and   < 0.4175 to the left,  improve= 7.970143, (0 missing)
      to    < 0.499  to the right, improve= 7.721040, (0 missing)
  Surrogate splits:
      there < 0.0115 to the right, agree=0.882, adj=0.625, (0 split)
      and   < 0.4175 to the left,  agree=0.863, adj=0.563, (0 split)
      on    < 0.0745 to the left,  agree=0.863, adj=0.563, (0 split)
      of    < 0.7775 to the right, agree=0.824, adj=0.437, (0 split)
      to    < 0.4745 to the right, agree=0.824, adj=0.437, (0 split)

Node number 2: 35 observations
  predicted class=Hamilton  expected loss=0  P(node) =0.6862745
    class counts:     0    35     0     0     0
   probabilities: 0.000 1.000 0.000 0.000 0.000

Node number 3: 16 observations
  predicted class=Madison   expected loss=0.3125  P(node) =0.3137255
    class counts:     0     1     0     4    11
   probabilities: 0.000 0.062 0.000 0.250 0.688

>   printcp(Fed_tuned1)

Classification tree:
rpart(formula = author ~ ., data = train, method = "class", control = rpart.c
ontrol(minsplit = 10,
    maxdepth = 1))

Variables actually used in tree construction:
[1] upon

Root node error: 15/51 = 0.29412

n= 51

       CP nsplit rel error xerror    xstd
1 0.66667      0   1.00000    1.0 0.21693
2 0.01000      1   0.33333    0.4 0.15339
```

Figure 13

We see from the cp plot that the Tuned Model 1 did not change from the original decision tree model. Therefore, we have adjusted the parameters in the second tuned model for maxdepth. We have kept the min-split the same as the previous value, 10 . However, our maxdepth has now been increased to 5 (Figure 14 & 15). Our inspection of our second tune model shows our relative error decreasing. We seem to be on the right track.



```
> plotcp(Fed_tuned2) #visualize cross-validation results
> ##DT Tuned Model 2
> #minsplit = 10, maxdepth=5
>   Fed_tuned2<-rpart(author ~., data = train, method = 'class',
+                    control = rpart.control( minsplit=10, maxdepth = 5 ))
> rpart.plot(Fed_tuned2)

>
```

Figure 14

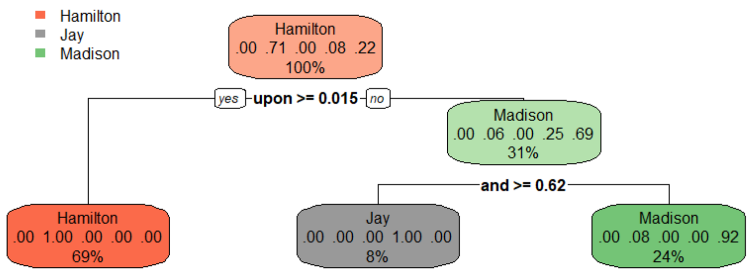


Figure 15

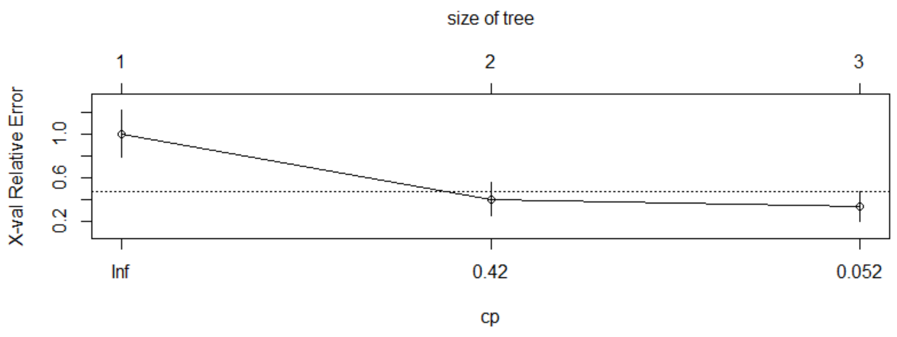
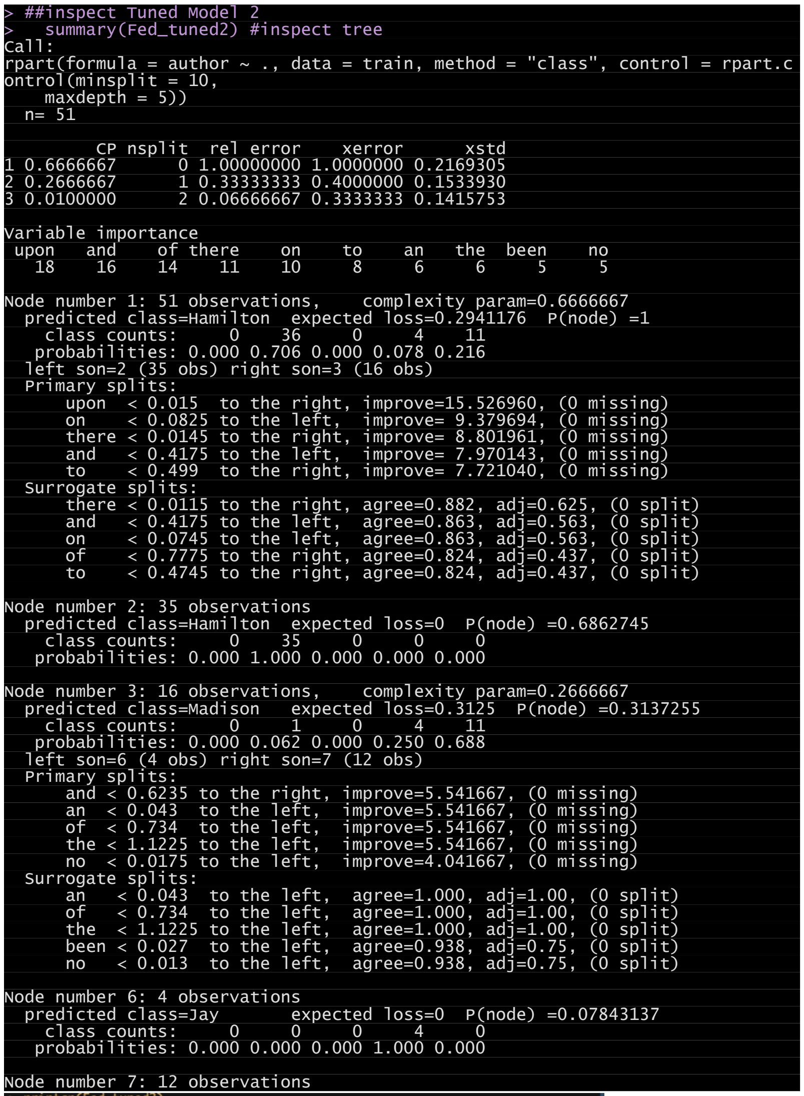
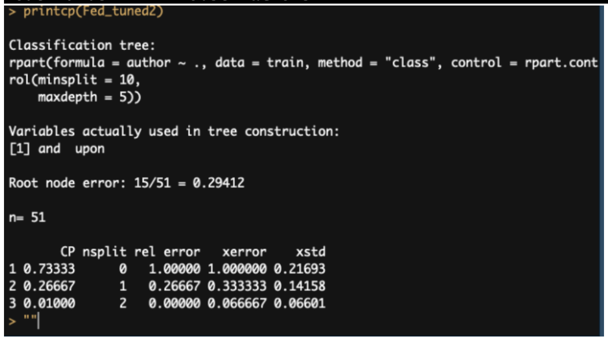


Figure 16



```
> ##inspect Tuned Model 2
>   summary(Fed_tuned2) #inspect tree
Call:
rpart(formula = author ~ ., data = train, method = "class", control = rpart.c
ontrol(minsplit = 10,
    maxdepth = 5))
  n= 51

         CP nsplit  rel error    xerror      xstd
1 0.6666667      0 1.00000000 1.0000000 0.2169305
2 0.2666667      1 0.33333333 0.4000000 0.1533930
3 0.0100000      2 0.06666667 0.3333333 0.1415753

Variable importance
 upon   and    of there    on    to    an   the  been    no 
   18    16    14    11    10     8     6     6     5     5 

Node number 1: 51 observations,    complexity param=0.6666667
  predicted class=Hamilton  expected loss=0.2941176  P(node) =1
    class counts:     0    36     0     4    11
   probabilities: 0.000 0.706 0.000 0.078 0.216 
  left son=2 (35 obs) right son=3 (16 obs)
  Primary splits:
      upon  < 0.015  to the right, improve=15.526960, (0 missing)
      on    < 0.0825 to the left,  improve= 9.379694, (0 missing)
      there < 0.0145 to the right, improve= 8.801961, (0 missing)
      and   < 0.4175 to the left,  improve= 7.970143, (0 missing)
      to    < 0.499  to the right, improve= 7.721040, (0 missing)
  Surrogate splits:
      there < 0.0115 to the right, agree=0.882, adj=0.625, (0 split)
      and   < 0.4175 to the left,  agree=0.863, adj=0.563, (0 split)
      on    < 0.0745 to the left,  agree=0.863, adj=0.563, (0 split)
      of    < 0.7775 to the right, agree=0.824, adj=0.437, (0 split)
      to    < 0.4745 to the right, agree=0.824, adj=0.437, (0 split)

Node number 2: 35 observations
  predicted class=Hamilton  expected loss=0  P(node) =0.6862745
    class counts:     0    35     0     0     0
   probabilities: 0.000 1.000 0.000 0.000 0.000 

Node number 3: 16 observations,    complexity param=0.2666667
  predicted class=Madison   expected loss=0.3125  P(node) =0.3137255
    class counts:     0     1     0     4    11
   probabilities: 0.000 0.062 0.000 0.250 0.688 
  left son=6 (4 obs) right son=7 (12 obs)
  Primary splits:
      and < 0.6235 to the right, improve=5.541667, (0 missing)
      an  < 0.043  to the left,  improve=5.541667, (0 missing)
      of  < 0.734  to the left,  improve=5.541667, (0 missing)
      the < 1.1225 to the left,  improve=5.541667, (0 missing)
      no  < 0.0175 to the left,  improve=4.041667, (0 missing)
  Surrogate splits:
      an   < 0.043  to the left,  agree=1.000, adj=1.00, (0 split)
      of   < 0.734  to the left,  agree=1.000, adj=1.00, (0 split)
      the  < 1.1225 to the left,  agree=1.000, adj=1.00, (0 split)
      been < 0.027  to the left,  agree=0.938, adj=0.75, (0 split)
      no   < 0.013  to the left,  agree=0.938, adj=0.75, (0 split)

Node number 6: 4 observations
  predicted class=Jay       expected loss=0  P(node) =0.07843137
    class counts:     0     0     0     4     0
   probabilities: 0.000 0.000 0.000 1.000 0.000 

Node number 7: 12 observations
```

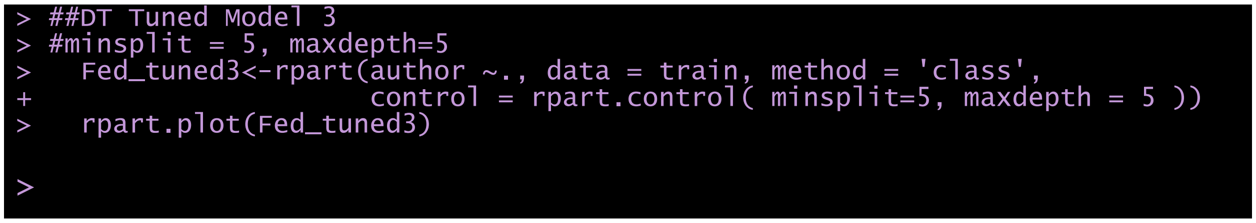


```
> printcp(Fed_tuned2)

Classification tree:
rpart(formula = author ~ ., data = train, method = "class", control = rpart.cont
rol(minsplit = 10,
    maxdepth = 5))

Variables actually used in tree construction:
[1] and  upon

Root node error: 15/51 = 0.29412

n= 51

       CP nsplit rel error   xerror    xstd
1 0.73333      0   1.00000 1.000000 0.21693
2 0.26667      1   0.26667 0.333333 0.14158
3 0.01000      2   0.00000 0.066667 0.06601
> ""
```

Figure 17

Considering that our second tuned model was a better fit, with higher accuracy then our previous model. We decided to adjust the third model further but looking at minsplit. This is the minimum number of observations that must exist in a node in order for a split to be attempted.

We decided to lower this to consider and adjust for potential overfitting. In the third tuned model, the minsplit was lowered to 5, while the maxdepth was maintained at 5.



```
> ##DT Tuned Model 3
> #minsplit = 5, maxdepth=5
>   Fed_tuned3<-rpart(author ~., data = train, method = 'class',
+                     control = rpart.control( minsplit=5, maxdepth = 5 ))
>   rpart.plot(Fed_tuned3)

>
```

Figure 18

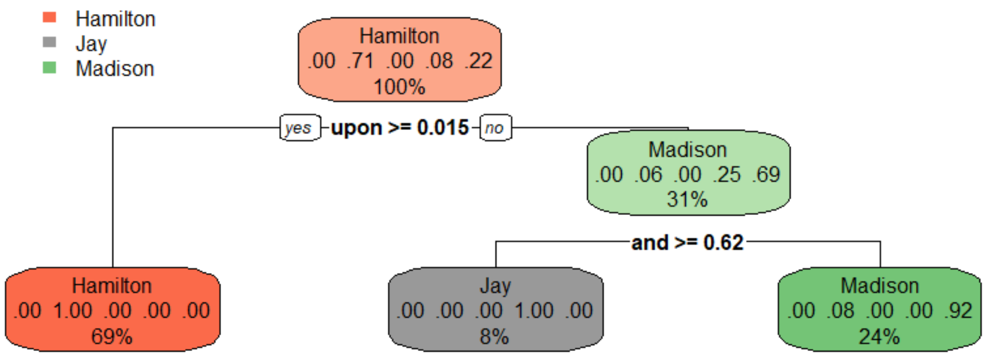


Figure 19

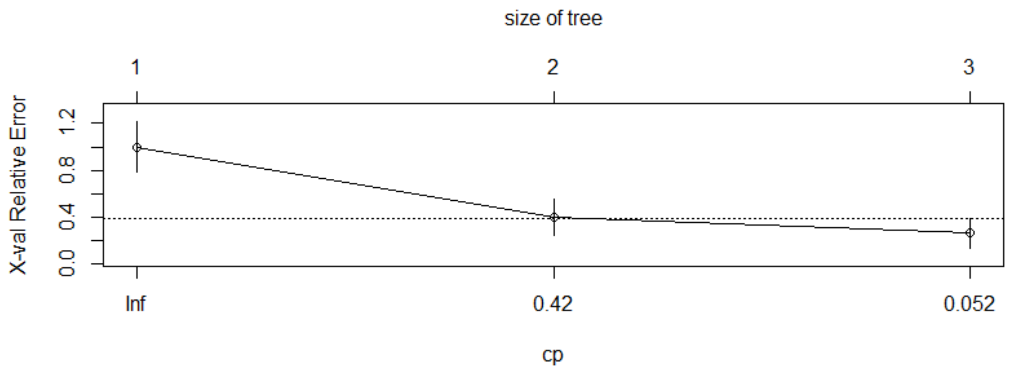
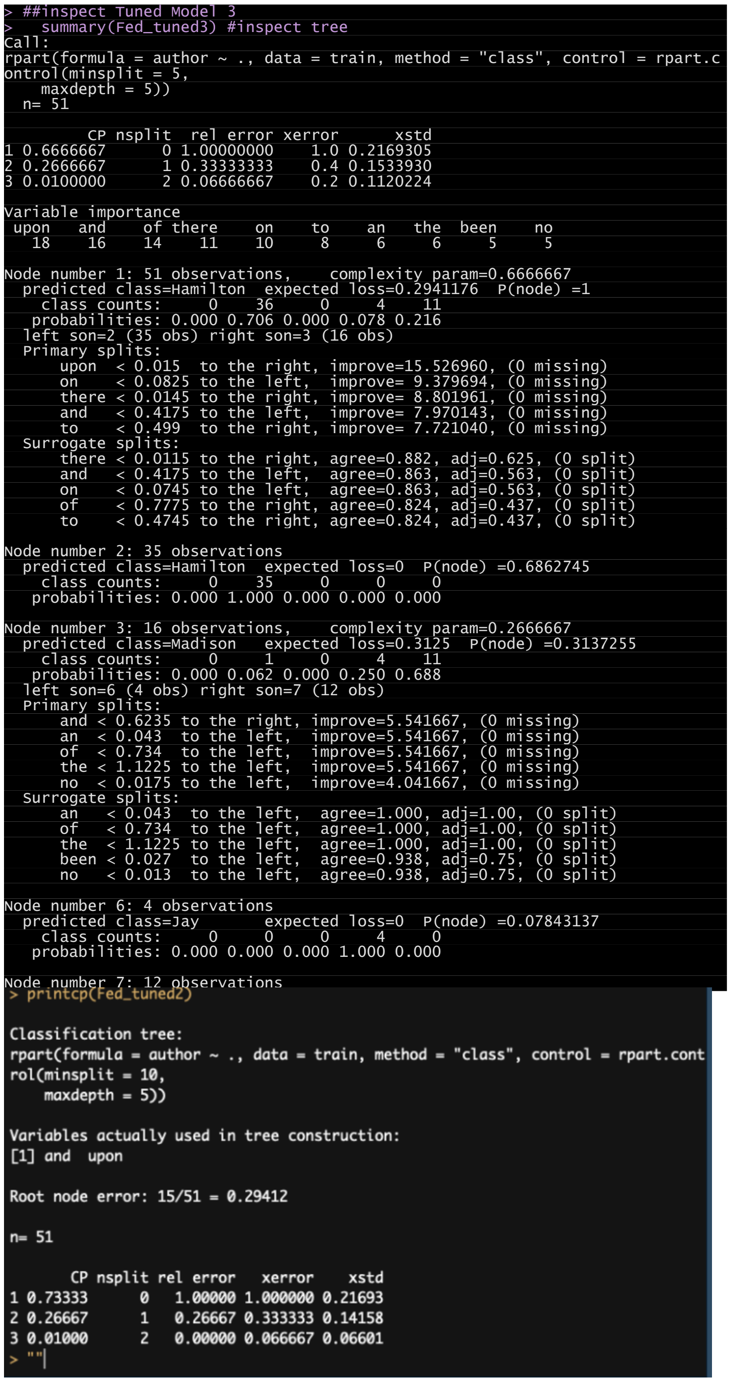


Figure 20

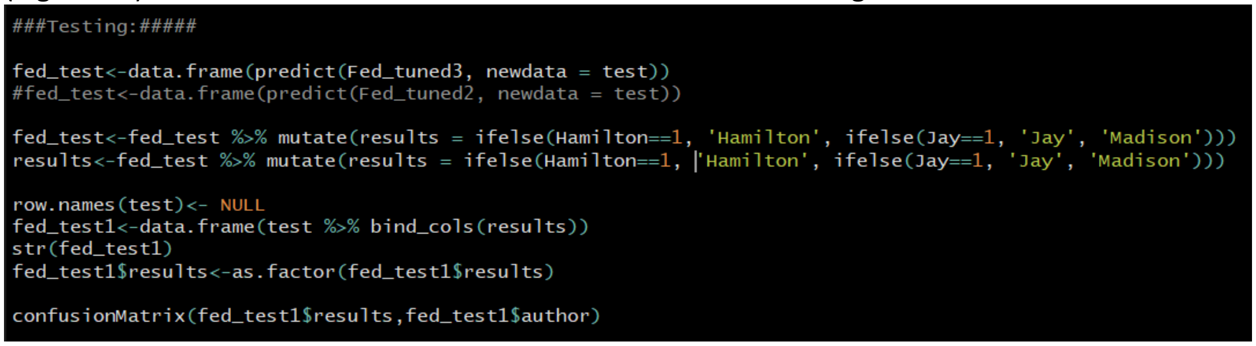


```
> ##inspect Tuned Model 3
>   summary(Fed_tuned3) #inspect tree
Call:
rpart(formula = author ~ ., data = train, method = "class", control = rpart.c
ontrol(minsplit = 5,
    maxdepth = 5))
  n= 51

         CP nsplit  rel error xerror      xstd
1 0.6666667      0 1.00000000    1.0 0.2169305
2 0.2666667      1 0.33333333    0.4 0.1533930
3 0.0100000      2 0.06666667    0.2 0.1120224

Variable importance
 upon   and    of there    on    to    an   the  been    no 
   18    16    14    11    10     8     6     6     5     5 

Node number 1: 51 observations,    complexity param=0.6666667
  predicted class=Hamilton  expected loss=0.2941176  P(node) =1
    class counts:     0    36     0     4    11
   probabilities: 0.000 0.706 0.000 0.078 0.216 
  left son=2 (35 obs) right son=3 (16 obs)
  Primary splits:
      upon  < 0.015  to the right, improve=15.526960, (0 missing)
      on    < 0.0825 to the left,  improve= 9.379694, (0 missing)
      there < 0.0145 to the right, improve= 8.801961, (0 missing)
      and   < 0.4175 to the left,  improve= 7.970143, (0 missing)
      to    < 0.499  to the right, improve= 7.721040, (0 missing)
  Surrogate splits:
      there < 0.0115 to the right, agree=0.882, adj=0.625, (0 split)
      and   < 0.4175 to the left,  agree=0.863, adj=0.563, (0 split)
      on    < 0.0745 to the left,  agree=0.863, adj=0.563, (0 split)
      of    < 0.7775 to the right, agree=0.824, adj=0.437, (0 split)
      to    < 0.4745 to the right, agree=0.824, adj=0.437, (0 split)

Node number 2: 35 observations
  predicted class=Hamilton  expected loss=0  P(node) =0.6862745
    class counts:     0    35     0     0     0
   probabilities: 0.000 1.000 0.000 0.000 0.000 

Node number 3: 16 observations,    complexity param=0.2666667
  predicted class=Madison   expected loss=0.3125  P(node) =0.3137255
    class counts:     0     1     0     4    11
   probabilities: 0.000 0.062 0.000 0.250 0.688 
  left son=6 (4 obs) right son=7 (12 obs)
  Primary splits:
      and < 0.6235 to the right, improve=5.541667, (0 missing)
      an  < 0.043  to the left,  improve=5.541667, (0 missing)
      of  < 0.734  to the left,  improve=5.541667, (0 missing)
      the < 1.1225 to the left,  improve=5.541667, (0 missing)
      no  < 0.0175 to the left,  improve=4.041667, (0 missing)
  Surrogate splits:
      an   < 0.043  to the left,  agree=1.000, adj=1.00, (0 split)
      of   < 0.734  to the left,  agree=1.000, adj=1.00, (0 split)
      the  < 1.1225 to the left,  agree=1.000, adj=1.00, (0 split)
      been < 0.027  to the left,  agree=0.938, adj=0.75, (0 split)
      no   < 0.013  to the left,  agree=0.938, adj=0.75, (0 split)

Node number 6: 4 observations
  predicted class=Jay       expected loss=0  P(node) =0.07843137
    class counts:     0     0     0     4     0
   probabilities: 0.000 0.000 0.000 1.000 0.000 

Node number 7: 12 observations
```

```
> printcp(Fed_tuned2)

Classification tree:
rpart(formula = author ~ ., data = train, method = "class", control = rpart.cont
rol(minsplit = 10,
    maxdepth = 5))

Variables actually used in tree construction:
[1] and  upon

Root node error: 15/51 = 0.29412

n= 51 

       CP nsplit rel error   xerror    xstd
1 0.73333      0   1.00000 1.000000 0.21693
2 0.26667      1   0.26667 0.333333 0.14158
3 0.01000      2   0.00000 0.066667 0.06601
> ""
```

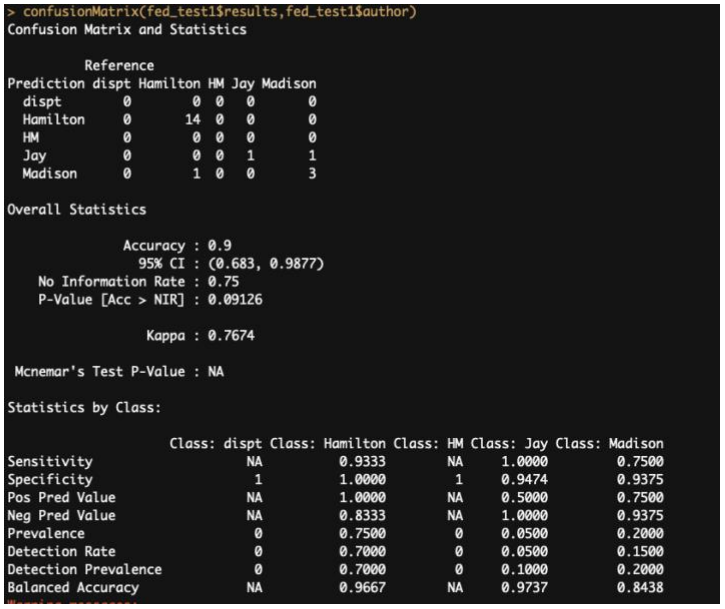
Figure 21

After establishing the best fit training model (discussed below in results). We went ahead and created our testing model. Our test model is used to help us predict (next section) who the disputed author is. The data set was mutated to account for if an author wrote the paper, then they would be assigned so. Some issues were run into during the creation of the test model. All issues seem to have been worked out. First, we had a problem with the row names not being correctly identified. To solve this we set the row.names to null. Initially, we were making all changes within the “test” data set. However, we had issues creating a confusion matrix. We solved the issue by creating an additional test set (fed_test1) and were able to bind all the columns with nullified row names into it. Additionally, we set the changed the results to factor (Figure 22). The results of our confusion matrix can be found in Figure 23.



```
###Testing:#####

fed_test<-data.frame(predict(Fed_tuned3, newdata = test))
#fed_test<-data.frame(predict(Fed_tuned2, newdata = test))

fed_test<-fed_test %>% mutate(results = ifelse(Hamilton==1, 'Hamilton', ifelse(Jay==1, 'Jay', 'Madison')))
results<-fed_test %>% mutate(results = ifelse(Hamilton==1, 'Hamilton', ifelse(Jay==1, 'Jay', 'Madison')))

row.names(test)<- NULL
fed_test1<-data.frame(test %>% bind_cols(results))
str(fed_test1)
fed_test1$results<-as.factor(fed_test1$results)

confusionMatrix(fed_test1$results,fed_test1$author)
```

Figure 22



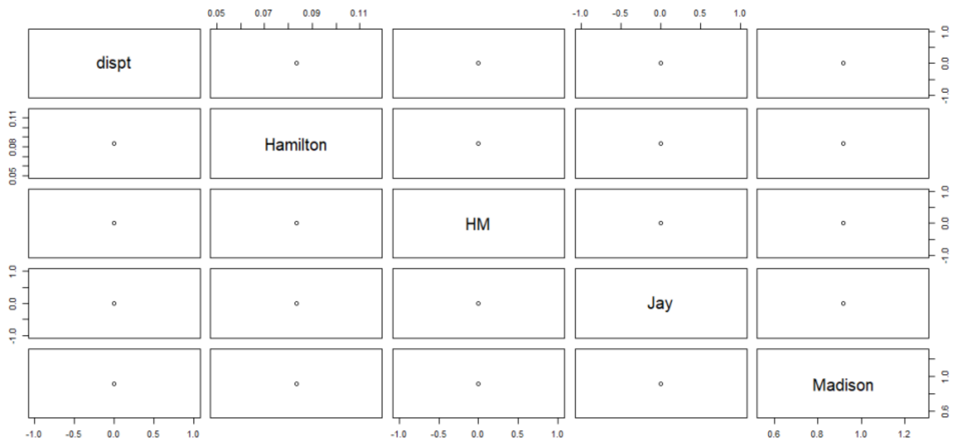
```
> confusionMatrix(fed_test1$results,fed_test1$author)
Confusion Matrix and Statistics

          Reference
Prediction dispt Hamilton HM Jay Madison
  dispt        0        0  0   0       0
  Hamilton     0       14  0   0       0
  HM           0        0  0   0       0
  Jay          0        0  0   1       1
  Madison      0        1  0   0       3

Overall Statistics

               Accuracy : 0.9
                 95% CI : (0.683, 0.9877)
    No Information Rate : 0.75
    P-Value [Acc > NIR] : 0.09126

                  Kappa : 0.7674

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: dispt Class: Hamilton Class: HM Class: Jay Class: Madison
Sensitivity                    NA          0.9333        NA     1.0000         0.7500
Specificity                     1          1.0000         1     0.9474         0.9375
Pos Pred Value                 NA          1.0000        NA     0.5000         0.7500
Neg Pred Value                 NA          0.8333        NA     1.0000         0.9375
Prevalence                      0          0.7500         0     0.0500         0.2000
Detection Rate                  0          0.7000         0     0.0500         0.1500
Detection Prevalence            0          0.7000         0     0.1000         0.2000
Balanced Accuracy              NA          0.9667        NA     0.9737         0.8438
```

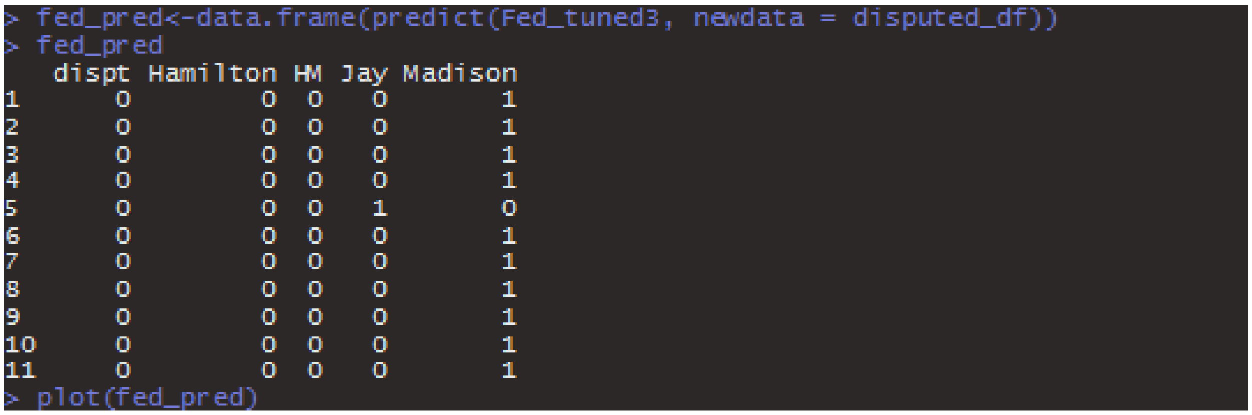
Figure 23

Section 3: Prediction

Now that have built the classification models, we are able to build a prediction model. Based on the results (discussed below), we applied our best fit model to the disputed papers. On our case this was the third model. We first created our prediction data frame to include the tuned model 3 and, adjusted newdata to be the disputed dataset. The results are shown in figure 24. We plotted this (figure 25) for graphical effects as well.

Parin Patel





```
> fed_pred<-data.frame(predict(Fed_tuned3, newdata = disputed_df))
> fed_pred
   dispt Hamilton HM Jay Madison
1      0        0  0   0       1
2      0        0  0   0       1
3      0        0  0   0       1
4      0        0  0   0       1
5      0        0  0   1       0
6      0        0  0   0       1
7      0        0  0   0       1
8      0        0  0   0       1
9      0        0  0   0       1
10     0        0  0   0       1
11     0        0  0   0       1
> plot(fed_pred)
```

Results:

To predict the likely author of the disputed Federalist papers, we needed to first train a model. The results of our first general training model (Figure 7) show root note error, or the percentage of correctly sorted records at the first (root) splitting node, to be 29.4%. Additionally, relative error (1-R^2) is the error for the predictions of the data that were used to estimate the model. For the original, untuned model, the relative error is 33% with two splits, or nodes. It is key to note that in the original model, no papers have been assigned to Jay. The model will be tuned in the next stages to account for papers written by Jay.

Note: We went back and removed papers written by “HM” because we did not want our model to be adjusted for both authors, Madison or Hamilton. When we originally ran the full code and trees to account for “HM”, the level of accuracy was significantly lower than when we removed it. The results for the original model, stated above, are with “HM” removed.

As stated previously, our goal is to find the optimal training model. We choose to do this by using a plot of cp and relative error percentages to understand the accuracy of the model. By adjusting the rpart control parameters, we can plot the cp value (the complexity parameter) against the geometric mean. This way, we can adjust the parameters until the minimum value is reached. The results of the first tuned model showed that specifying the minsplit to 10 maxdepth to 1 did not affect the model much from the original. The relative error was again 33% with two splits, and the root note error was 29.4% again, as well. This similar outcome of the first model is likely because the original model set these values as the default since we did not specify in the code.

In the second tune model, we initially see in the tree that Jay has been included in the decision making process. This was missing from the first model, but is more representative of the data We see from the plot and summary of cp that our relative error is 29%. This has decreased from the first model. Therefore, our second model is more accurate. It is interesting that the split for papers being written by Jay, instead of Madison, are 62% more likely to include the word "and".

In our third tuned model, the tree does not split any further, so our optimal size of tree of 3. This has an error of our relative error is the same as model 2 at 29% Even though the parameters in model 3 are different than 2, we still have come to the same results. We believe that we have tuned our model to its best level through model three.

For the testing data set, the results show (Figure 23) that our model has a 90% prediction accuracy. In addition, the confusion matrix shows that the model correctly almost correctly assigned the papers, the exception being one of Madison's papers went to Hamilton. As shown in the reference, the positive predicted value is highest with Hamilton's assigned papers. However, Jay has the highest level of sensitivity in the model.

Part 3:
The results of the prediction model show that our of the 11 disputed papers, 10 were written by Madison and 1 was written by Jay. Since we removed HM from the set, 0 papers were associated with it. Hamilton had none of the disputed papers assigned. Therefore, based on this model, we can say Madison wrote majority of the disputed papers.

Conclusion:

While the Federalist Papers remain, an important document related to the history of founding and ratification of the US Constitution, understanding who the author of the 11 disputed papers is still considered somewhat of a mystery. While our results were able to, for the most part, show a stronger likelihood that the author was Madison, the results are still not conclusive. One of the main questions raised through analysis is that what should be do with papers assigned to Hamilton and Madison "HM". In the future, further analysis should be done on this data set and then added to the models final conclusion. We decided to remove it from this model because we were worried it would confused the models final results because training it to understand that a paper could be written by more than 1 author would be out of the scope of this assignment.

In conclusion, while we did find that the disputed papers are likely the product of Madison, it is also possible that he had help. Further modeling and analysis is required.

Appendix A: R Code

```
install.packages("rpart")
install.packages("caret")
install.packages("rpart.plot")
install.packages("RColorBrewer")
install.packages("dplyr")
library("caret")
library("rpart")
library("lattice")
library("ggplot2")
library("rpart.plot")
library("RColorBrewer")
library("dplyr")

##load data
data_path=file.choose()
fedpapers=read.csv(data_path)
str(fedpapers)



################### PART 1 ###################
################### preprocess ###################
##Reference used: https://rstudio-pubs-
static.s3.amazonaws.com/246827_b9a4314244084f00ad5139144a1997d8.html


##clean
## remove author + filename columns
fedpapers_clean = fedpapers[, -c(2)]
##Add Cleaning for HM. Remove all authors = HM. Rerun below code
fedpapers_clean<-subset(fedpapers_clean,author!="HM")
View(fedpapers_clean)

##Prepare  dataframes: training and test data
```

Parin Patel

```
  #nonDisputed Authors
  #create non_disputed df= excludes disputed files
nondis_df = subset(fedpapers_clean,author!= 'dispt')

  #split nondis into train and test
  nondis_key<-createDataPartition(y=nondis_df$author,p=0.7,list = FALSE)
  train<-nondis_df[nondis_key,]
  test<-nondis_df[-nondis_key,]


##Disputed files: create df for disputed = include disputed files
disputed_df = subset(fedpapers_clean,author == 'dispt')

################### PART 2 ###################
################### Decision tree ###################

##reference :  https://rpubs.com/chengjiun/52658,
##reference: https://cran.r-project.org/web/packages/rpart/rpart.pdf


#build general decision tree using rpart for only author.
fed_tree<-rpart(author ~., data = train, method = 'class')
rpart.plot(fed_tree) #viz tree


##inspect general model 1
summary(fed_tree) #inspect tree
printcp(fed_tree)
plotcp(fed_tree) #visualize cross-validation results
##Note: Cross-valuidation  estimates how accurately the
    #model generlizes the unseen data. (how well it performs/perdicts)

    #plotcp = cp values plotted agaisnt geometric mean
      #to depict the deviation until the minimum value is
      #reached. .
    ##Look into setting minsplot = 10


##Tune decision  tree:

##DT Tuned Model 1
#minsplit = 10, maxdepth=1
  Fed_tuned1<-rpart(author ~., data = train, method = 'class',
                  control = rpart.control( minsplit=10, maxdepth = 1 ))
```

Parin Patel

```
  rpart.plot(Fed_tuned1)

  ##inspect Tuned Model 1
  summary(Fed_tuned1) #inspect tree
  printcp(Fed_tuned1)
  plotcp(Fed_tuned1) #visualize cross-validation results


##DT Tuned Model 2
#minsplit = 10, maxdepth=5
  Fed_tuned2<-rpart(author ~., data = train, method = 'class',
                control = rpart.control( minsplit=10, maxdepth = 5 ))
  rpart.plot(Fed_tuned2)

  ##inspect Tuned Model 2
  summary(Fed_tuned2) #inspect tree
  printcp(Fed_tuned2)
  plotcp(Fed_tuned2) #visualize cross-validation results




  ##DT Tuned Model 3
#minsplit = 5, maxdepth=5
  Fed_tuned3<-rpart(author ~., data = train, method = 'class',
                control = rpart.control( minsplit=4, maxdepth = 2 ))
  rpart.plot(Fed_tuned3)
  ##inspect Tuned Model 3
  summary(Fed_tuned3) #inspect tree
  printcp(Fed_tuned3)
  plotcp(Fed_tuned3) #visualize cross-validation results


  ##DT Tuned Model 4 - ONLY A CHE
  ##Fed_tuned4<-rpart(author ~., data = train, method = 'class',
               # control =CK TO MAKE SURE 3 IS BEST
  #minsplit = 3, maxdepth=4 rpart.control( minsplit=3, maxdepth = 4 ))
  ##rpart.plot(Fed_tuned4)
  ##inspect Tuned Model 3
  #summary(Fed_tuned4) #inspect tree
  #printcp(Fed_tuned4)
  #plotcp(Fed_tuned4) #visualize cross-validation results
```

Parin Patel

```
###Testing:#####

fed_test<-data.frame(predict(Fed_tuned3, newdata = test))
#fed_test<-data.frame(predict(Fed_tuned2, newdata = test))

fed_test<-fed_test %>% mutate(results = ifelse(Hamilton==1, 'Hamilton', ifelse(Jay==1, 'Jay',
'Madison')))
results<-fed_test %>% mutate(results = ifelse(Hamilton==1, 'Hamilton', ifelse(Jay==1, 'Jay',
'Madison')))

row.names(test)<- NULL
fed_test1<-data.frame(test %>% bind_cols(results))
str(fed_test1)
fed_test1$results<-as.factor(fed_test1$results)


confusionMatrix(fed_test1$results,fed_test1$author)

  ###################PART 3###################
  ################### Prediction ###################

fed_pred<-data.frame(predict(Fed_tuned3, newdata = disputed_df))
fed_pred
plot(fed_pred)

#method 2
fed_pred2<-predict.rpart(Fed_tuned3,newdata = disputed_df, na.action = na.omit, type =
"prob")
fed_pred2
plot(fed_pred2)
```