**Introduction**

The Federalist Papers are a series of 85 essays arguing in support of the United StatesConstitution. [Alexander Hamilton](https://www.constitutionfacts.com/us-founding-fathers/about-the-founding-fathers/#ah), [James Madison](https://www.constitutionfacts.com/us-founding-fathers/about-the-founding-fathers/#jm), and [John Jay](https://www.constitutionfacts.com/us-supreme-court/supreme-court-justices/) were the authors behind the pieces, and the three men wrote collectively under the name of Publius to stay anonymous. At the time of publication, the authorship of the articles was a closely guarded secret. It wasn't until Hamilton's death in 1804 that a list crediting him as one of the authors became public. It claimed fully two-thirds of the essays for Hamilton. Many of these would be disputed by Madison later. The Federalist Papers are considered one of the most important sources for interpreting and understanding the original intent of the Constitution.

**Alexander Hamilton was the force behind the project and was responsible for recruiting James Madison and John Jay to write with him as Publius. John Jay was the author of five of the Federalist Papers. He would later serve as Chief Justice of the United States. Jay became ill after only contributed 4 essays and was only able to write one more before the end of the project. James Madison, Hamilton's major collaborator, later President of the United States and "Father of the Constitution", wrote 29 of the Federalist Papers, although Madison himself, and many others since then, asserted that he had written more. Various statisticians tried to find the authorship of the disputed papers and came out with their conclusions. One of the most famous analysis on the same was written by Mosteller and Wallace who analyzed the frequency distributions of common words in the papers using Naïve Bayes algorithm to draw their conclusions. The analysis provided in this study will use classification techniques to create models and predict who wrote the disputed papers.**

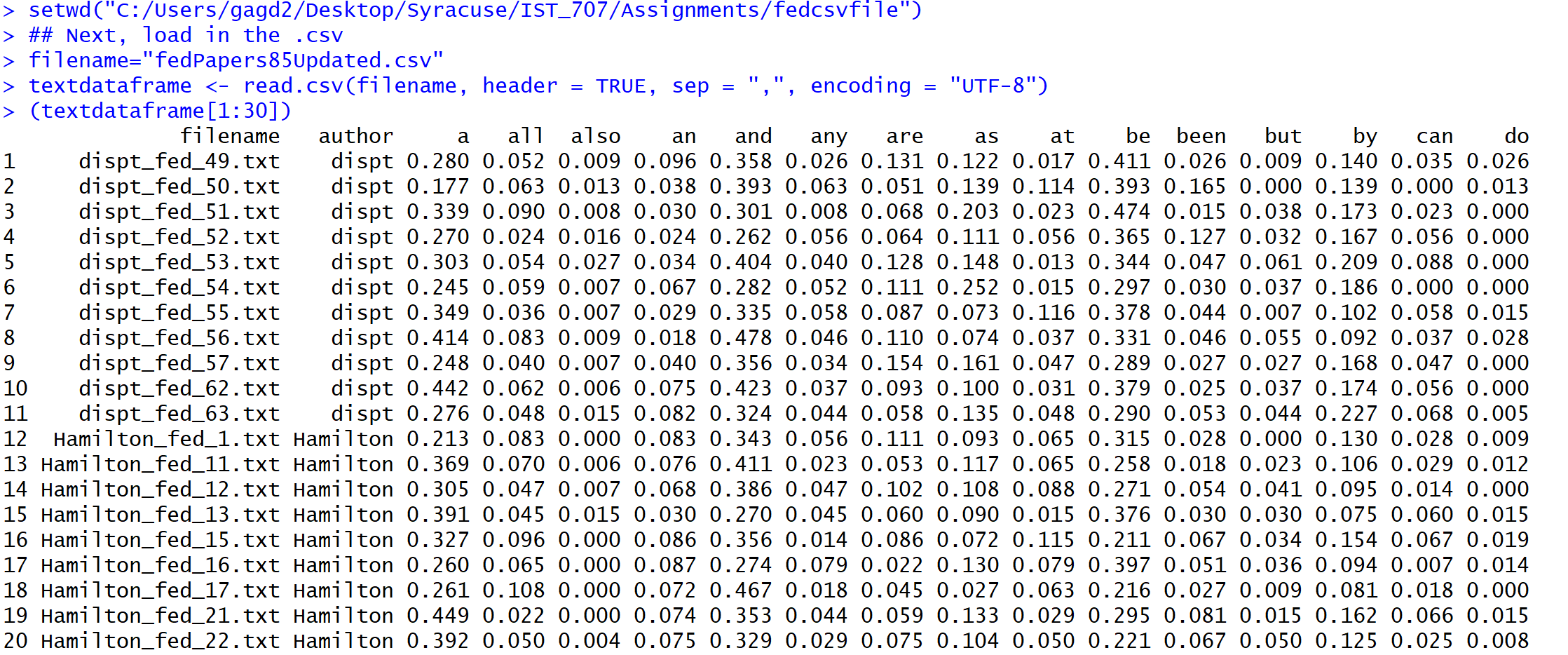
**Analysis and Models**

**About the Data**

**The original essays are available for free in the web and can be downloaded easily. The dataset is a csv file which has the filenames. There is a total of 74 rows with identified authors: 51 files written by Hamilton, 15 by Madison, 3 by Hamilton and Madison, 5 by Jay. The remaining 11 files, however, is authored by “Hamilton or Madison”. These are the famous essays with disputed authorship. To identify the authorship of the essays easily, the files were named with the author names who wrote it. For example, an essay written by Hamilton was named as ‘Hamilton\_fed\_1’. The essays that are under dispute for authorship are named as ‘dispt\_fed\_49’ with the last digits representing the file number.**

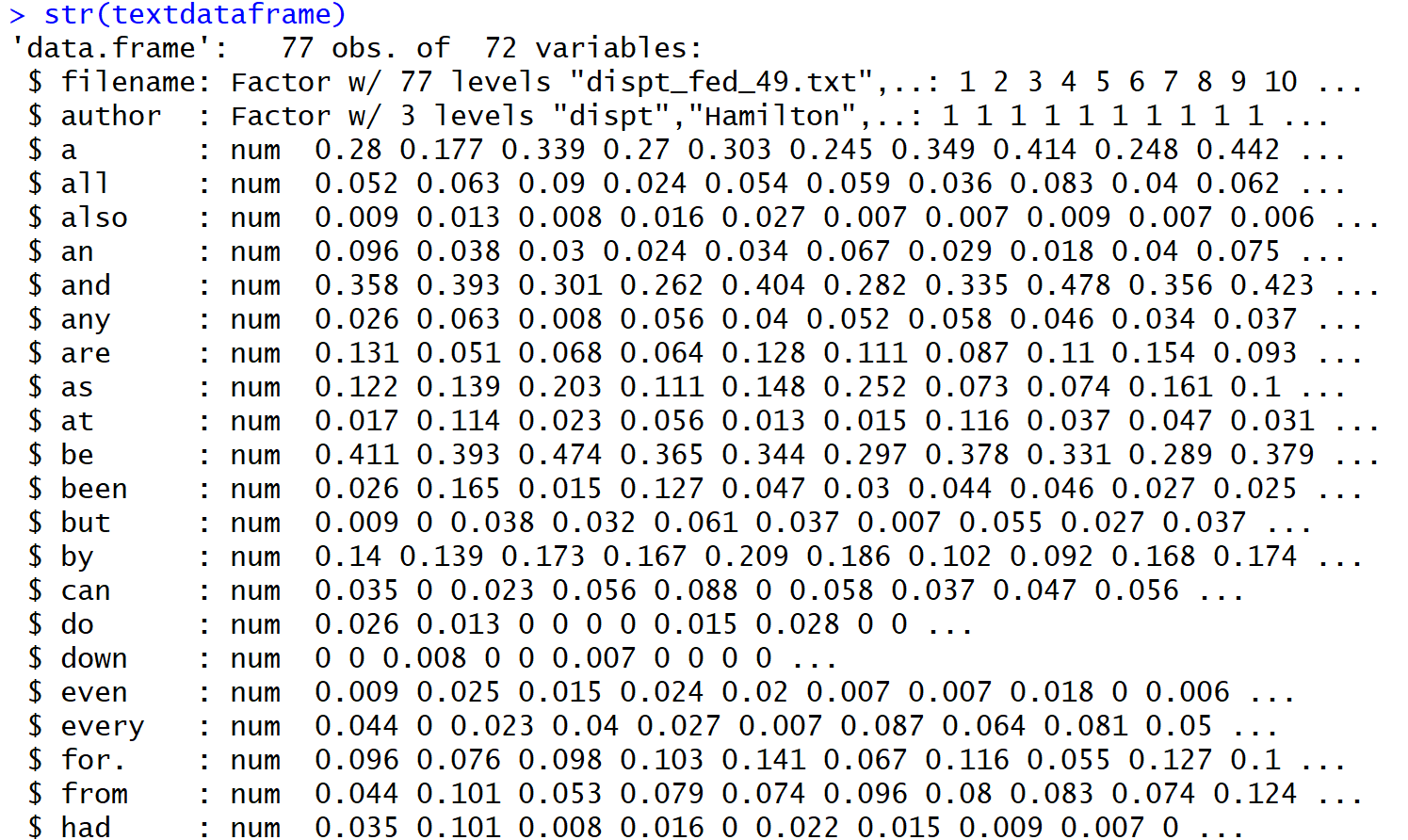
**As always, data must be cleaned and preprocessed before the analysis. Since the models are going to find out if the papers were written by Hamilton or Madison, it doesn’t make sense to keep the papers written by Jay. Hence, these papers are removed. Papers with joint ownership are also removed to avoid skewness in the analysis. Therefore, a total of 77 files in the csv file that remains. Rest of the columns in the file are the words used in the essays and the values represent the frequency of these words. The frequencies are already in a normalized state.**

**First, necessary libraries are installed, and the working directory path is set. The csv file is read into using read.csv function and stored in a data frame.**

****

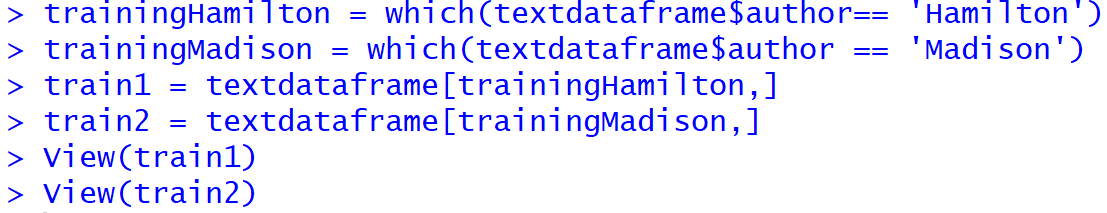
**When the data frame is viewed, the above screenshot shows the first few rows of the document and confirms that the file was read in successfully. An initial screening of the above result shows that the first two columns of the file are repetitive and one of them should be removed. Since the final prediction or the class label is the name of the author of the disputed files, *‘filename’* column can be removed.**

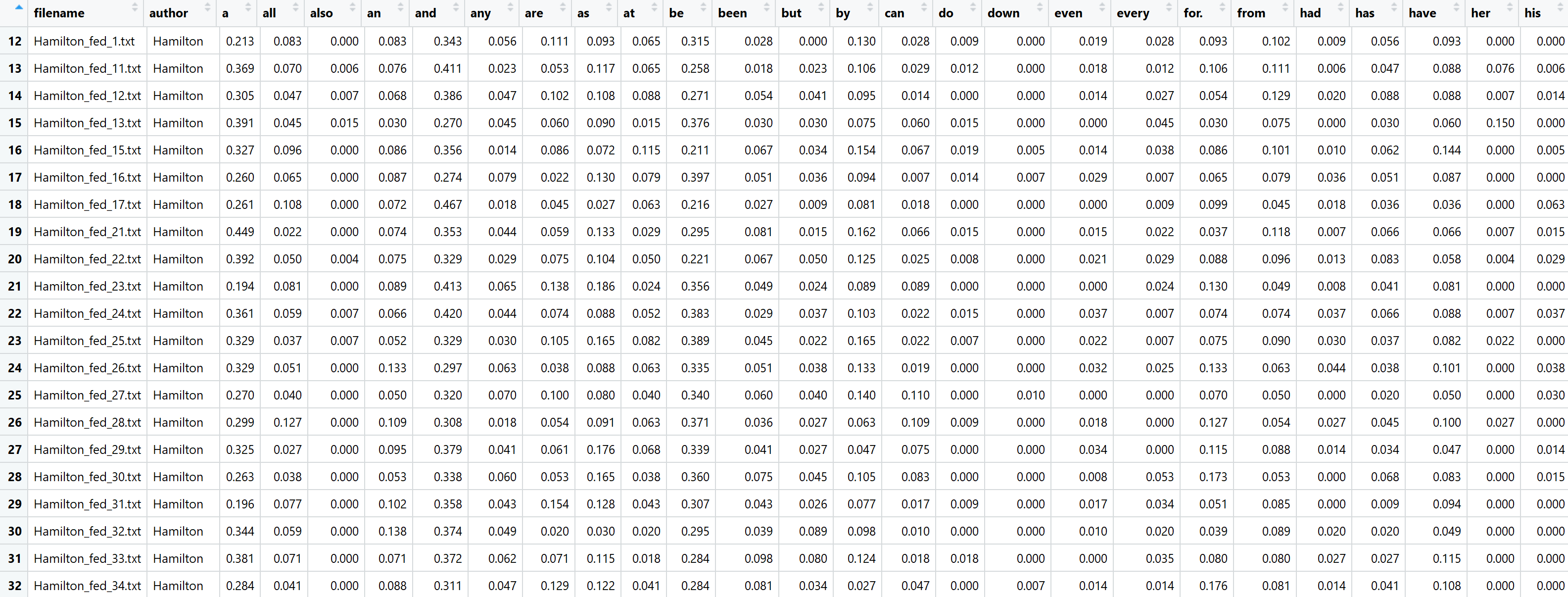
**The most important step in a classification problem is to make sure that the attributes are in the right form or data type. The class label or the column which is going to be predicted by the model should always be of type *‘factor’.* A simple *str* command shows the structure of the dataset with the data types of each attribute. The below screenshot shows that the class label *‘author’* is of type *factor* and the rest of the attributes are of the correct data types.**

****

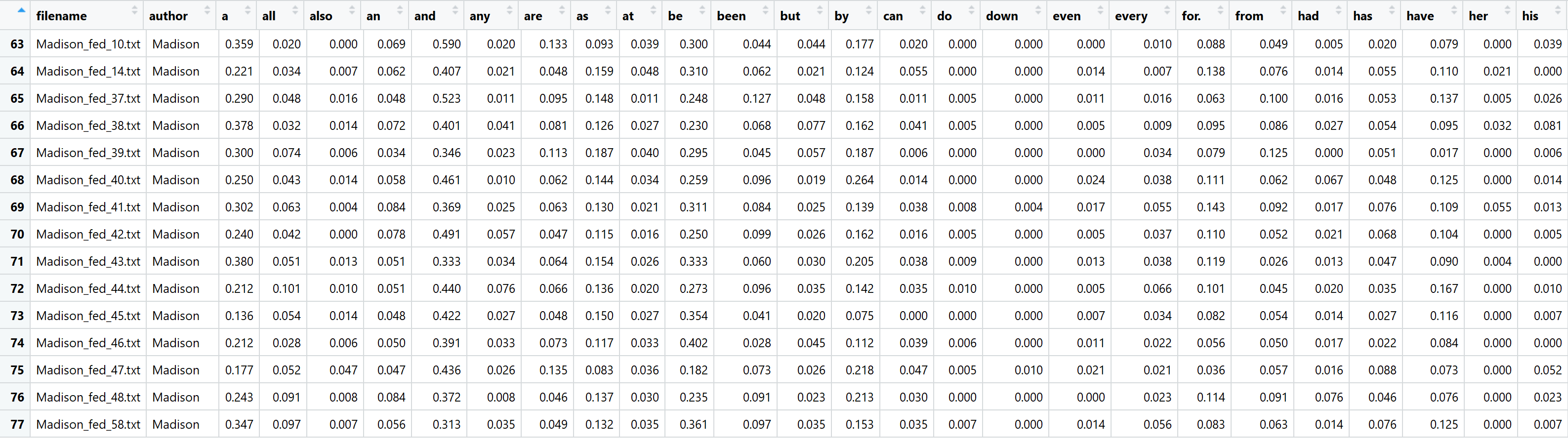
**Next, in order to apply classification techniques, data should be divided into training and test datasets. Since the final predictions are to be made on the disputed files, they are grouped into the testing set. Rest of the files are grouped into the training set. In order to build an efficient model, the training set itself is further divided into another training and testing subset. The reasoning behind this decision will be explained in the Decision tree model section.**

**To avoid an uneven comparison of Madison’s and Hamilton’s files, samples of equal size are taken from both sets to form a training set. For example, first, all rows that contain Hamilton’s files (51 in number) are separated and stored in a variable called ‘*trainingHamilton’* and all rows that contain Madison’s files (15 in number) are separated and stored in another variable called ‘*trainingMadison’.***

****

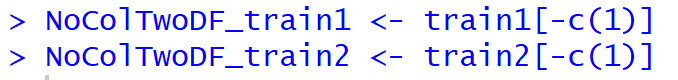
****

**A simple *View* command shows the training set (train1) with only Hamilton’s rows.**

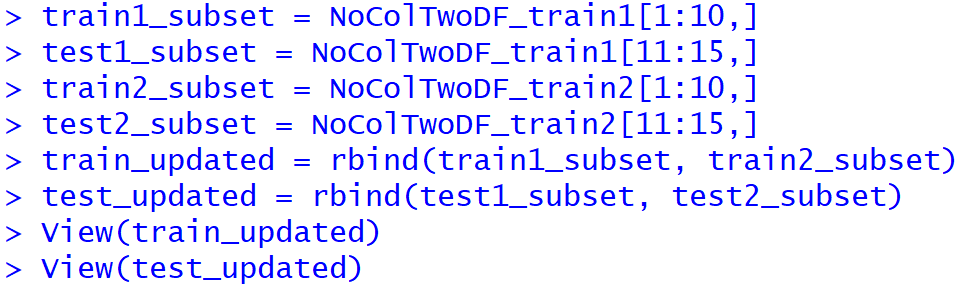
****

The above screenshot shows the training set(train2) with only Madison’s rows.

As discussed before, the column *filename* is redundant and is removed from both the datasets.



**Since the number of files in each of the above variables are not equal, the training set is made equal by taking an equal sample from each of these sets. The first 10 rows of Hamilton’s files are stored in a training variable and another 5 rows are stored in another testing variable. The same technique is applied to Madison’s files as well. A final training dataset variable is creating by combining Madison’s and Hamilton’s training sets and a test dataset variable is created by combining the two author’s test datasets. Note that this test dataset is different from the original and actual test dataset that contains the disputed paper files. This test dataset is used to test if the model’s predictions are good enough to apply on the final disputed papers test dataset.**

****

Now that the data has been cleaned and separated into the necessary formats, further analysis can be done to predict the disputed papers. A popular classification technique called decision tree algorithm will be used for the analysis of the federalist papers.

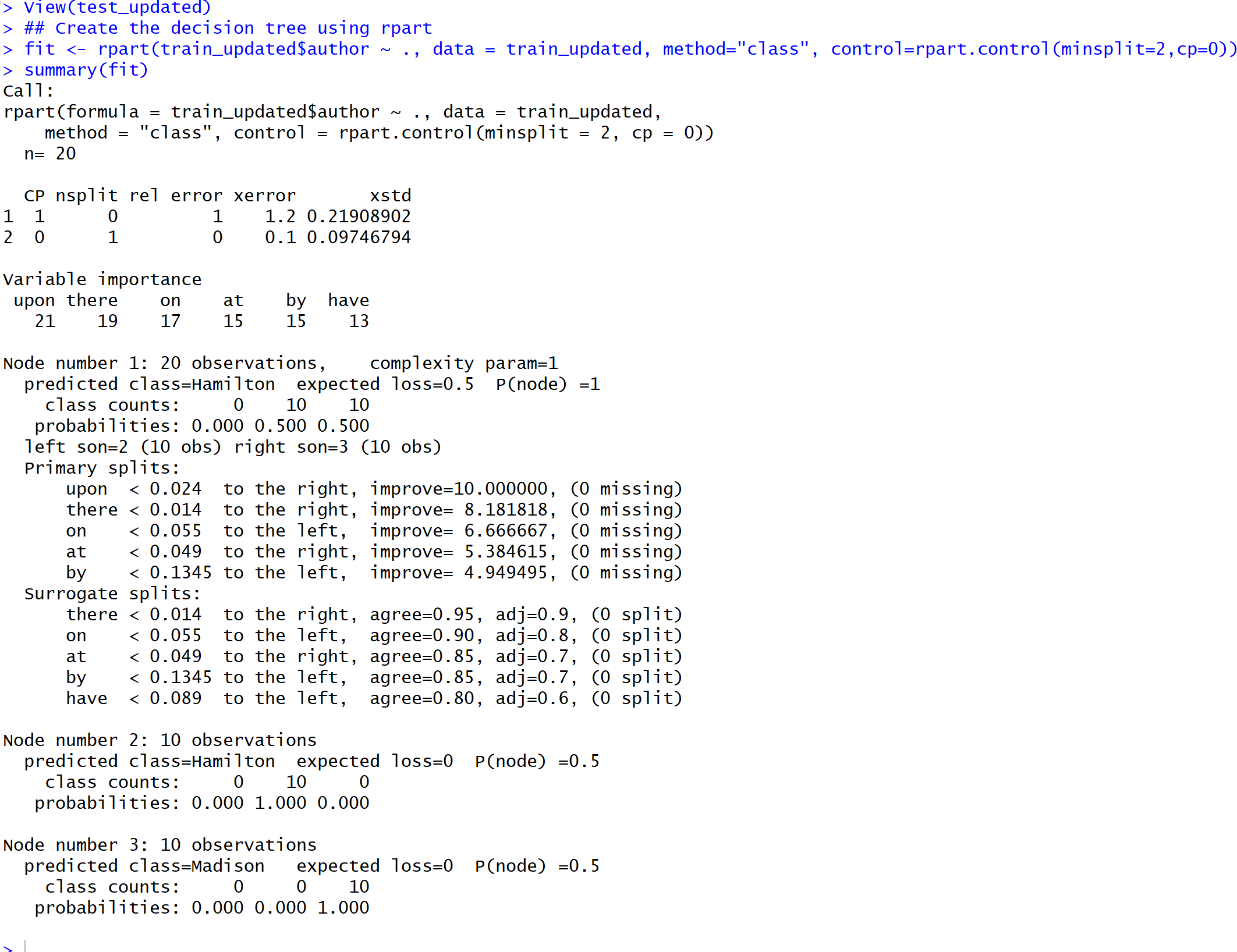
**Classification Techniques**

As per the definition provided in the book ‘Introduction to Data Mining’, classification is the task of learning a target function *f* that maps each attribute set *x* to one of the predefined class labels *y*. Classification techniques are most suited for predicting or describing datasets with binary or nominal categories. Different classification models can be built from an input data set using classification techniques. Some of the examples include decision-tree classifiers, rule-based classifiers, neural networks, support vector machines and naïve Bayes classifiers. In this paper, decision tree algorithm is used to classify the disputed papers. A general approach for solving classification problems is to divide the records of a data set into a training set and a testing set. The training set is used to build the classification model, which is subsequently applied to the test set that consists of the records with unknown class labels.

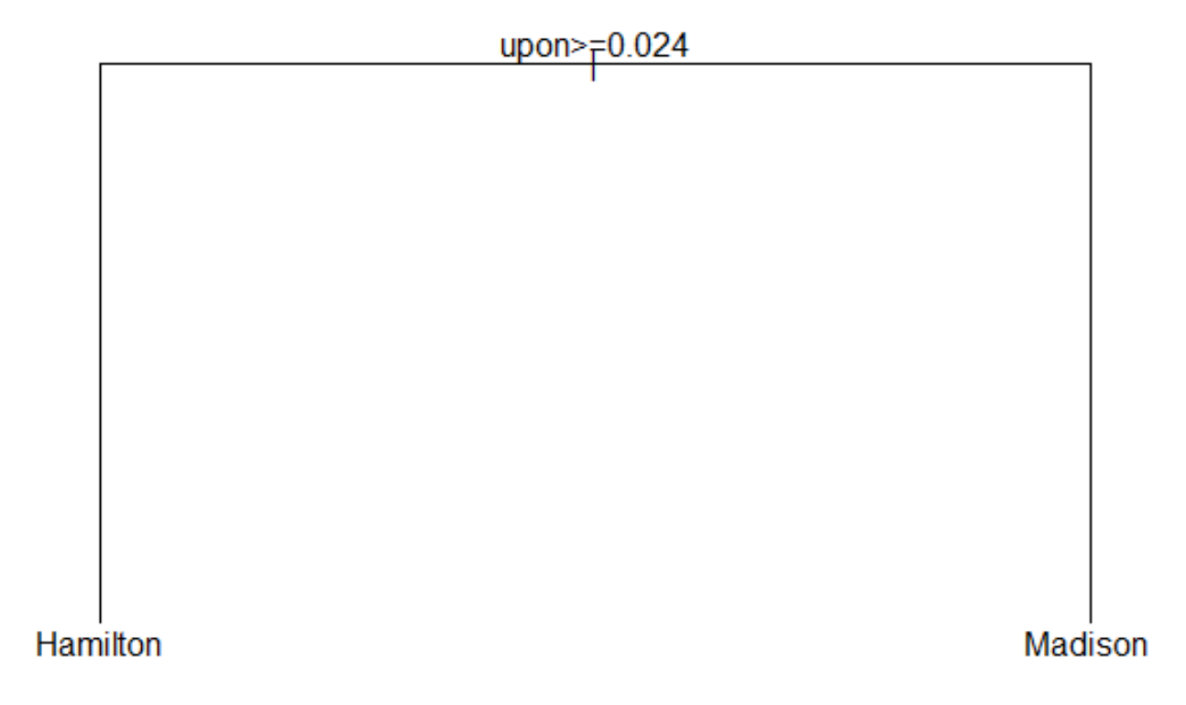
**Decision Tree Algorithm**

Decision tree algorithm is a simple and widely used classification technique that has several advantages. After the model has found the patterns in the data, it shows what decisions will be made for unseen data predictions. Decision trees are intuitive and can be read by people with little experience in data mining. Basically, the algorithm starts with all the data at the root node and scans all the variables for the best one to split on. The leaf nodes or the terminal nodes are the ones with the class labels or the column name that needs to be predicted. Classifying a test dataset is straightforward once a decision tree model has been built. Starting from the root node, the test condition is applied to the test records and the appropriate branches are followed based on the outcome. This will either lead to another internal node with another variable or to a leaf node with the class label.

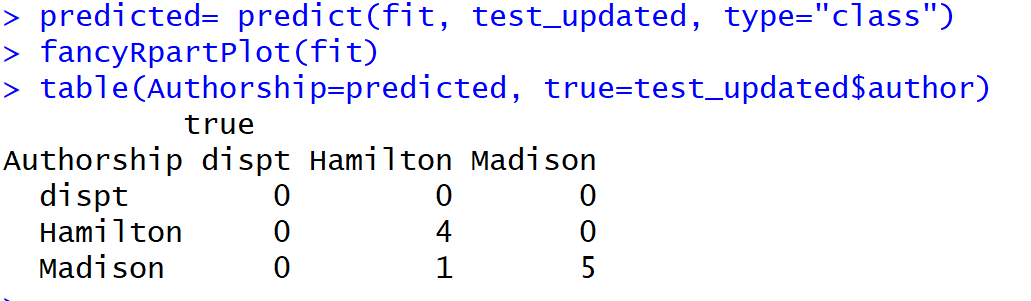
For the federalist paper analysis, the *rpart* function is used which works by splitting the dataset recursively, which means that the subsets that arise from a split are further split until a predetermined termination criterion is reached. As parameters, the training dataset is provided which specifies to use the *author* column as the *y* variable and use the rest of the columns in the file as the *x* attributes. The *method* parameter is set to *“class”,*which simply tells the algorithm that the predicted variable is discrete. Some other control parameters are also set to govern how many files\rows should sit in a bucket before looking for a split (minsplit) and the *cp* parameter is set to stop splits that aren’t deemed important enough.



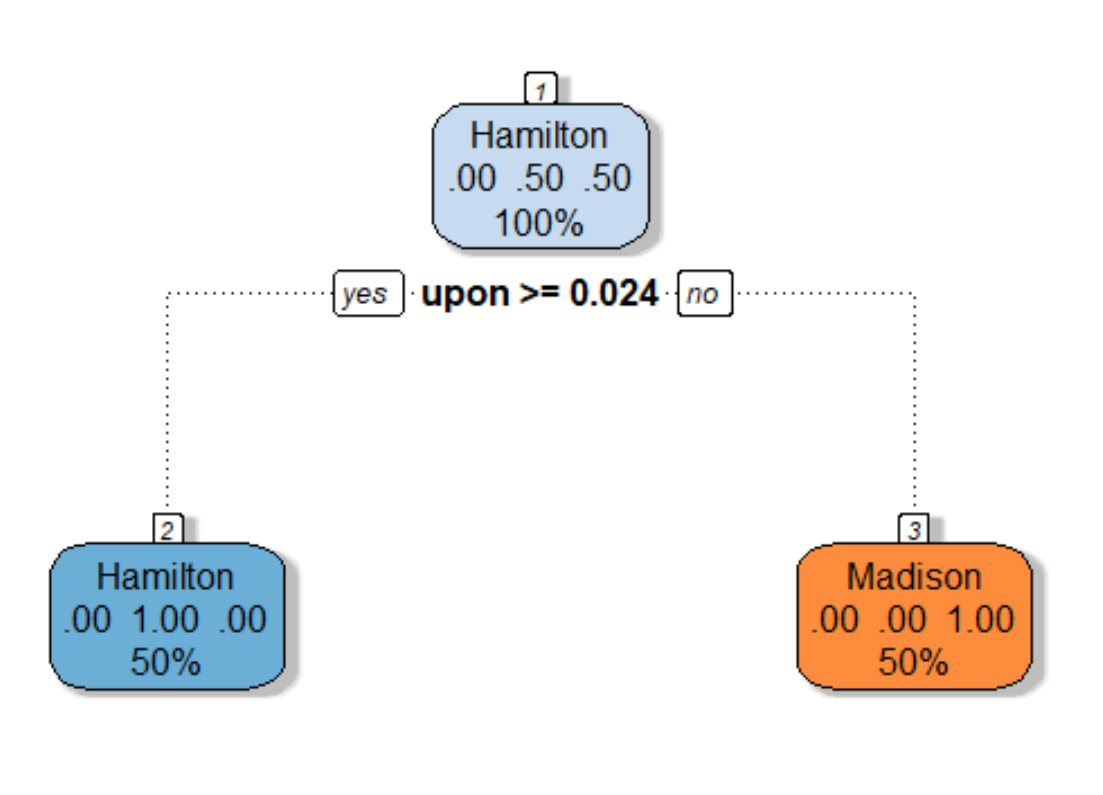
A quick summary of the model shows how the algorithm chooses the internal nodes. Using the plot and text functions, a graphical representation of the nodes can be viewed. which doesn’t give much information and hence rpart plots will be used later to visualize the decision trees.



Next, the model is tested on the test dataset (this is not the final test dataset) to see how well it fares in classifying the files accurately.



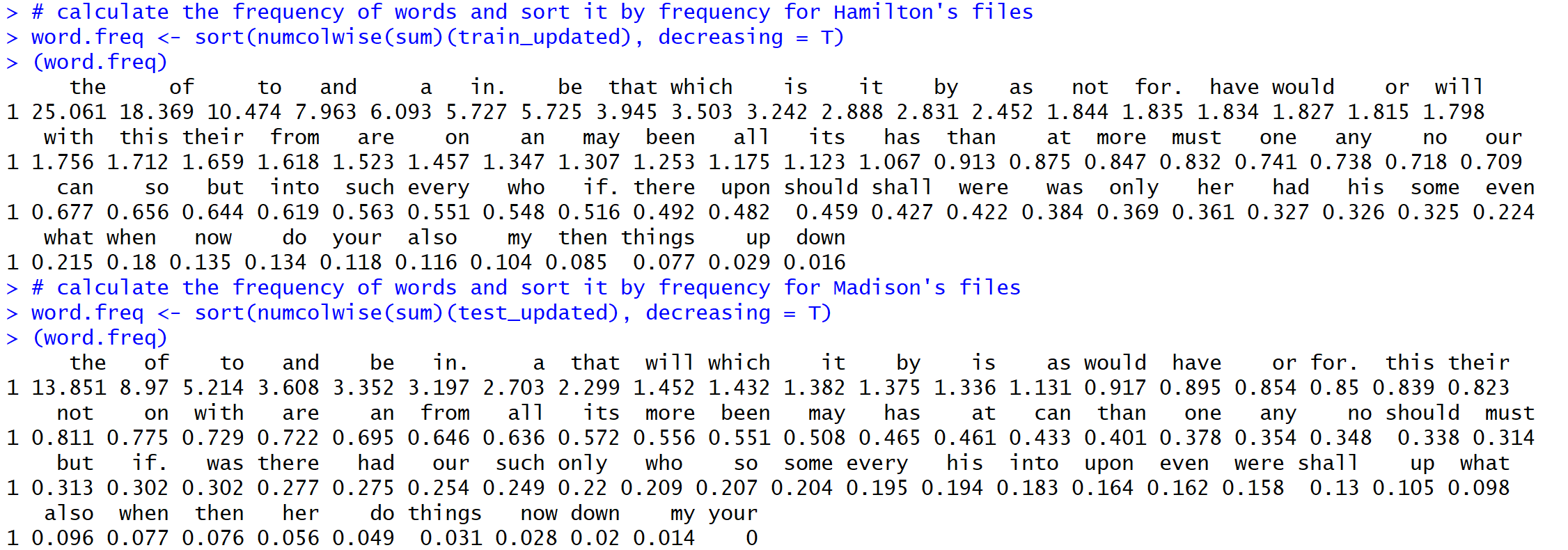
The above results show that the model was able to classify all 5 Madison files correctly but incorrectly classified one of Hamilton’s files as Madison’s file. Yet, this is not a bad model considering the results. Since there are not enough Madison files to create a bigger test set, not much exploration can be done here. A quick visualization of the model gives us the below decision tree.



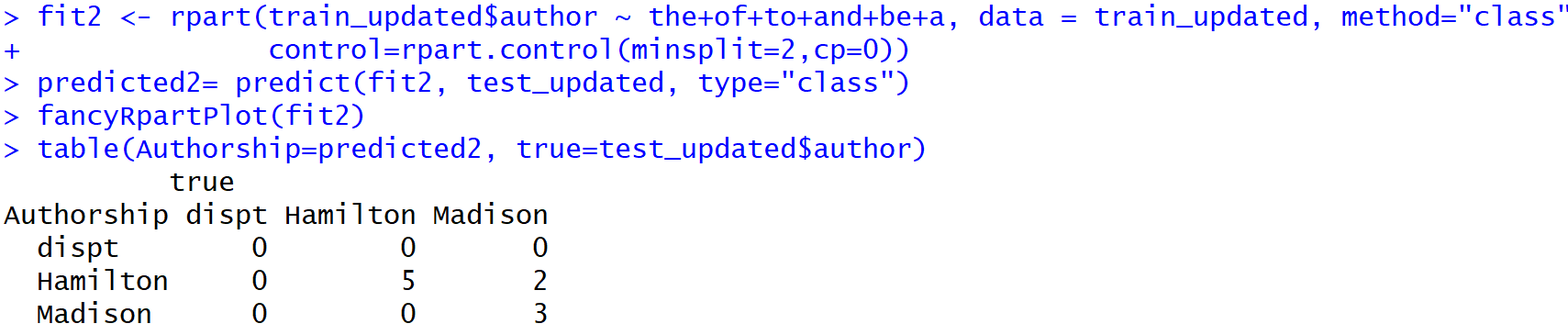
The above tree that was generated from the training set model shows that 50% of the files are Hamilton’s and the other 50% is Madison’s and the number below these proportions is the total which is 100% and indicates the proportion of the files that resides in this node or bucket. Since this one here is the root node, it is 100%. Traversing down the branches to the next node, if the word ‘*upon’* is used more than or equal to 0.024 times, the tree branches to the left node and if it is used less than 0.024 times, the tree branches to the right. If the word upon is >= 0.024, all files belong to Hamilton and the bucket votes that 50% of the files here belong to Hamilton. If the word *‘upon’* is < 0.024, all files belong to Madison and the bucket votes that 50% of the files belong to Madison.

The final nodes at the bottom of the decision tree are known as terminal nodes, or sometimes as leaf nodes. After all the Boolean choices have been made for a given file, they will end up in one of the leaf nodes, and the majority vote of all files in that bucket determine how the prediction for new files with unknown fates will be made.

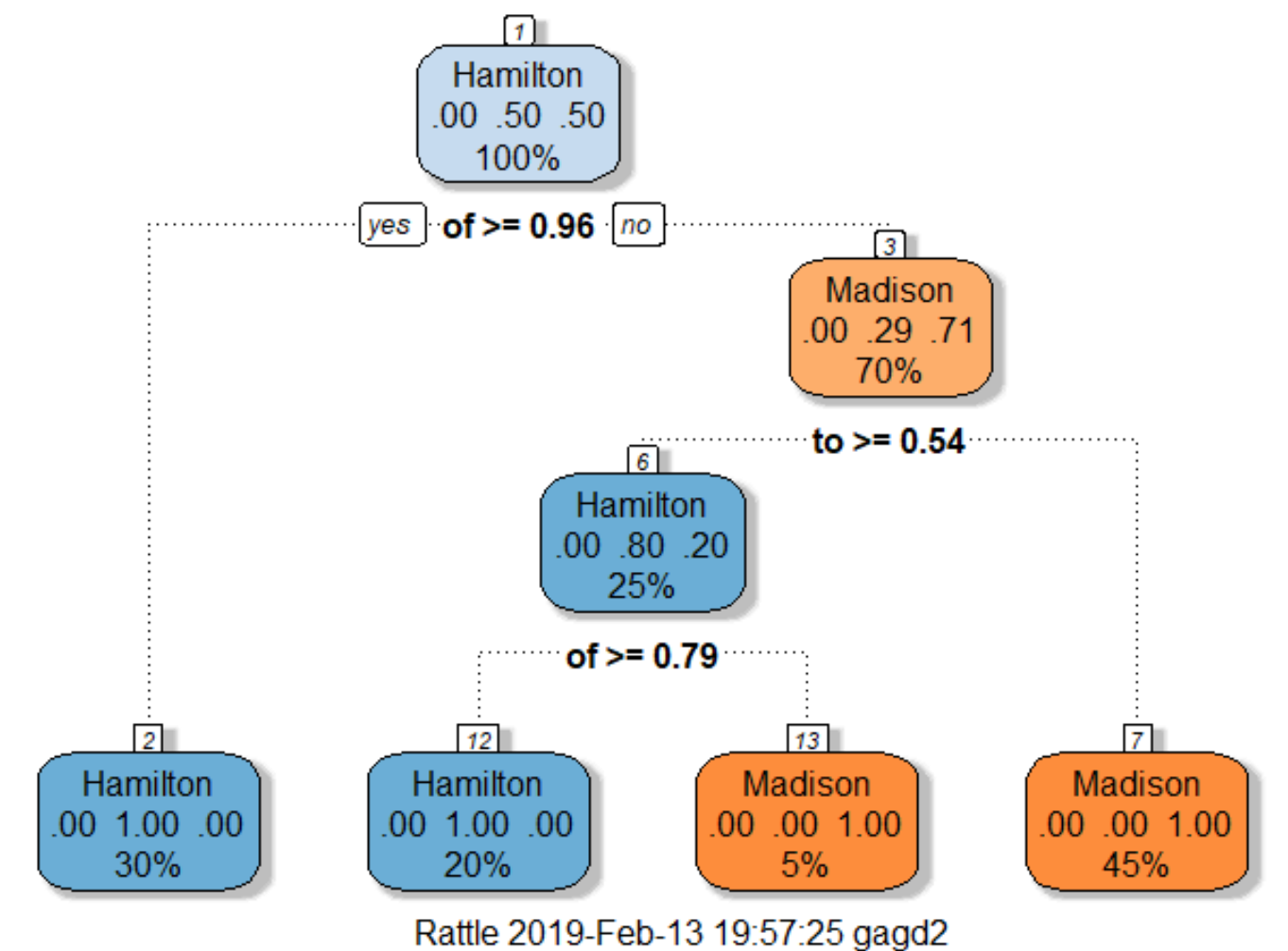
As one can see from the above tree, the algorithm used the attribute or word that it found to be the best fit for splitting. To get some more insights, user specified attributes can be fed into the model as x values for different analysis and trees. To visualize the same, first, the frequencies of the given words in all the files are calculated and the words with the highest frequency are selected as attributes for building the model.



Some of the high frequency words from the above list are selected and fed into the model below.

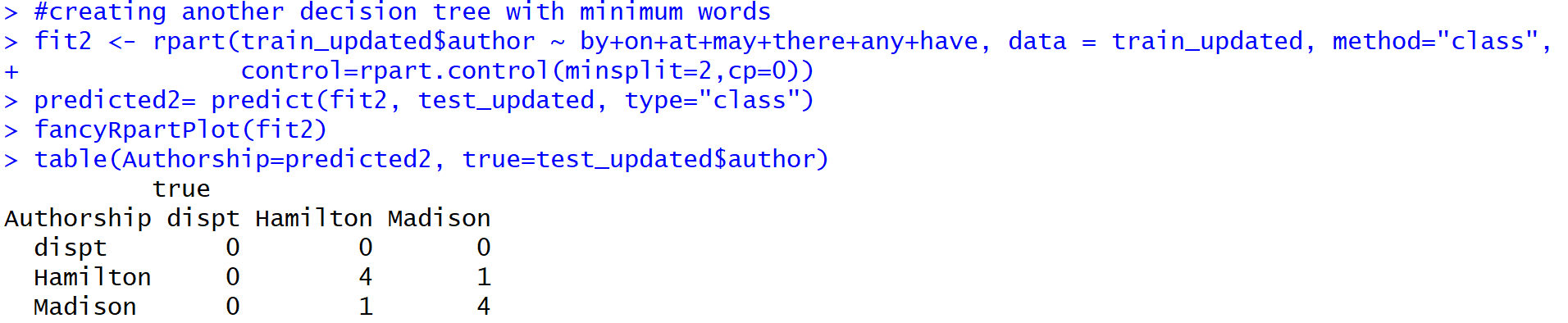


Using a confusion matrix to evaluate the accuracy of the model, the above screenshot shows that all the Hamilton’s files were classified accurately but 2 of Madison’s files were incorrectly classified as Hamilton’s files. In the previous model, only 1 file was incorrectly classified showing that the previous model is better than this one.

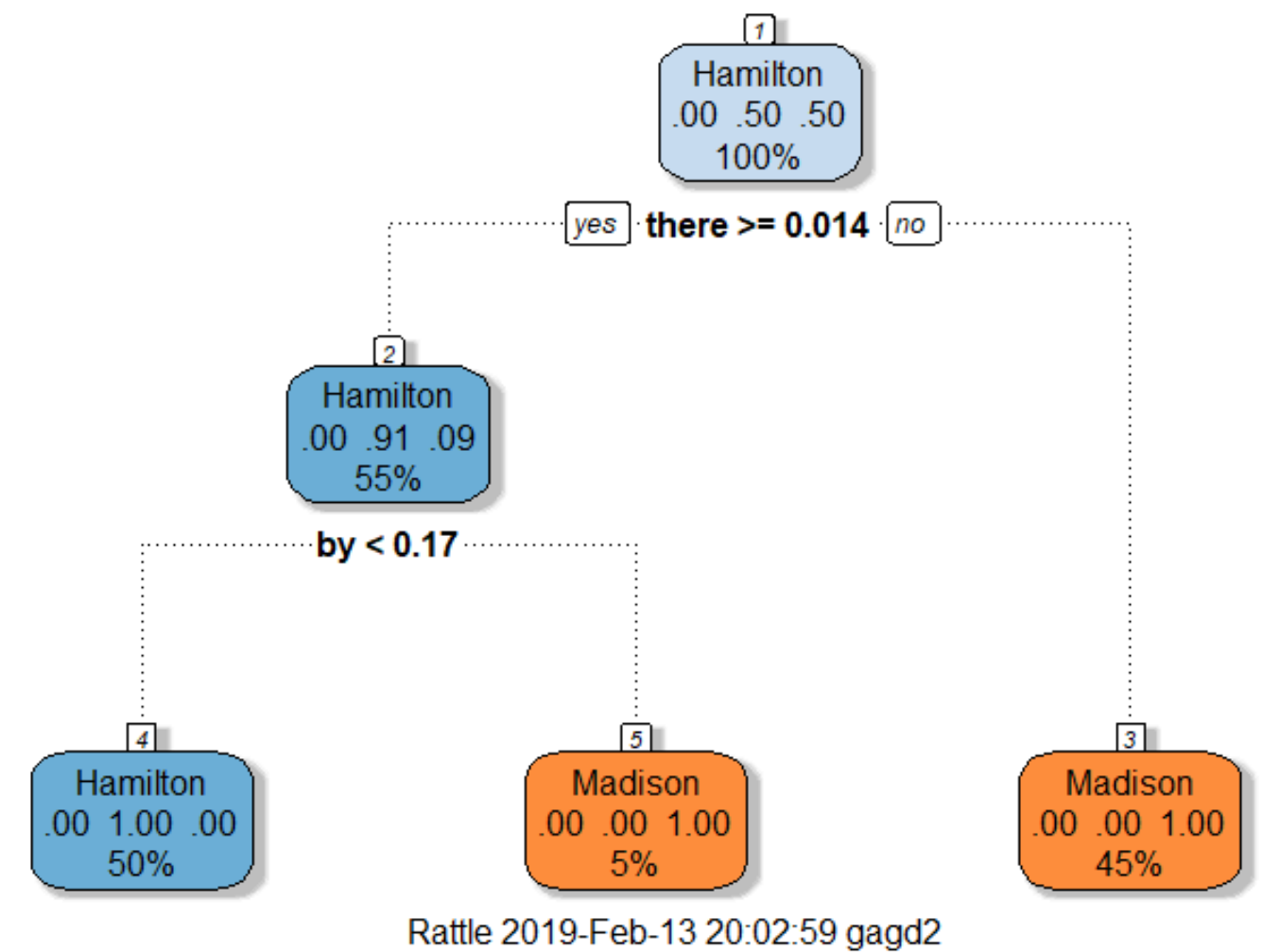


A decision tree can also be visualized which shows the different attributes that were used to build the model. Out of the words supplied, the model first picks up the word *‘of’* to split the files. It shows that 30% of the whole population belongs to Hamilton and 70% of the population of files belong to Madison. Out of the 70%, 29% of the files are classified as Hamilton and 71% of the files are classified as Madison. Since the model couldn’t conclude for the 70% of the files, it splits the tree further using the next word *‘to’* and keeps going on until all the files belong to either the Hamilton or the Madison bucket.

Likewise, a third decision tree was visualized by changing the word attributes in the model.



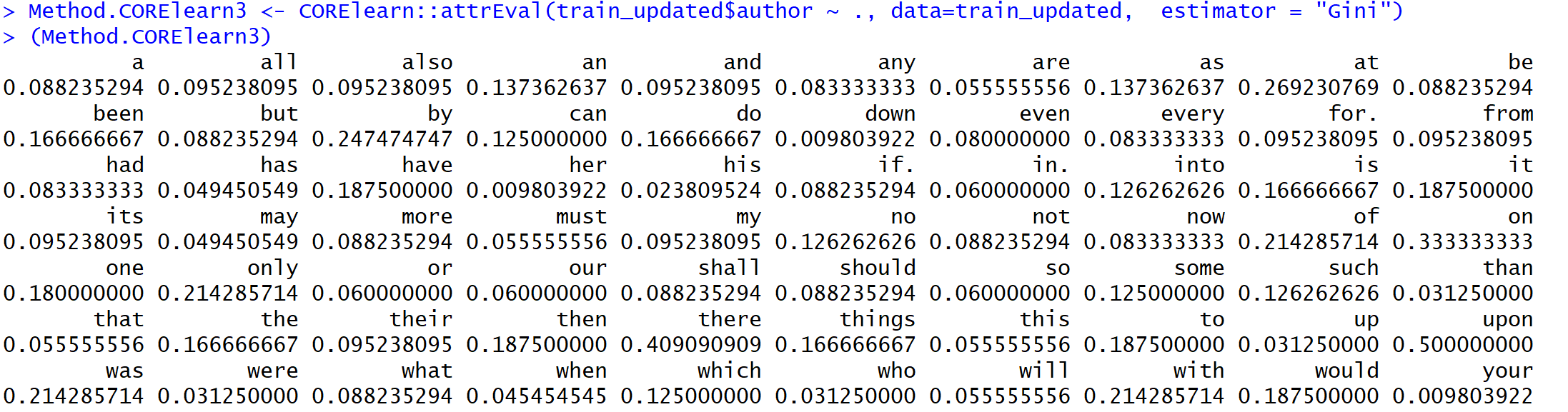
The above output is better than the second model but not as good as the first one. Here, one Madison file and one Hamilton file were incorrectly classified and the rest 8 were correctly classified.



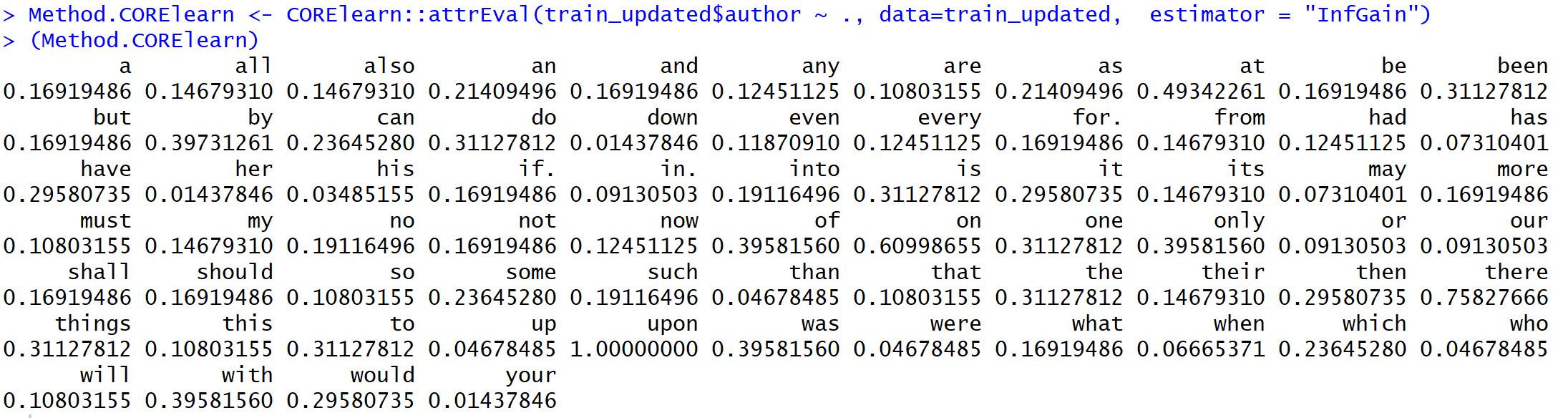
For the above tree, the model first picks the word *‘there’* and splits the files that have the word ‘there’ < 0.014 and classifies 45% of the entire file population as 100% Madison authored files. For files where the word *‘there’* is >= 0.014, the model classifies 55% of the entire file population as 91% Hamilton authored and 9% Madison authored files. Since it was not able to make a complete classification, the node is further split using the word *‘by’* which results in the final leaf nodes that classify 50% of the 55% files from the previous node as Hamilton’s and the rest 5% as Madison’s.

Splitting rules can be constructed in many ways, all of which are based on the notion of *impurity- a* measure of the degree of heterogeneity of the leaf nodes. Put another way, a leaf node that contains a single class is homogeneous and has impurity=0. There are three popular impurity quantification methods: Entropy (aka information gain), Gini Index and Gain Ratio.

The first model and tree that was built showed that the node was split using the word *‘upon’.* Why did the model select this attribute to split and classify the label? When the Gini index method was used to calculate the indexes, the word *‘upon’* had the maximum value of 0.5 which made the model choose this word. The below screenshot shows the Gini index values of all the attributes.

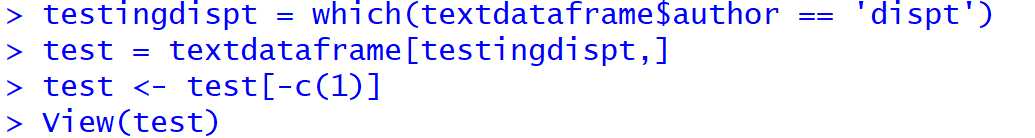


Another way to determine the goodness of a split is to calculate the information gain. Comparing the degree of impurity of the parent node before splitting with the degree of impurity of the child nodes after splitting, the performance of the model can be evaluated. The larger the difference the better the model. The below screenshot shows the information gain values of all the attributes in our dataset. The word ‘upon’ once again has the highest information gain value of 1.

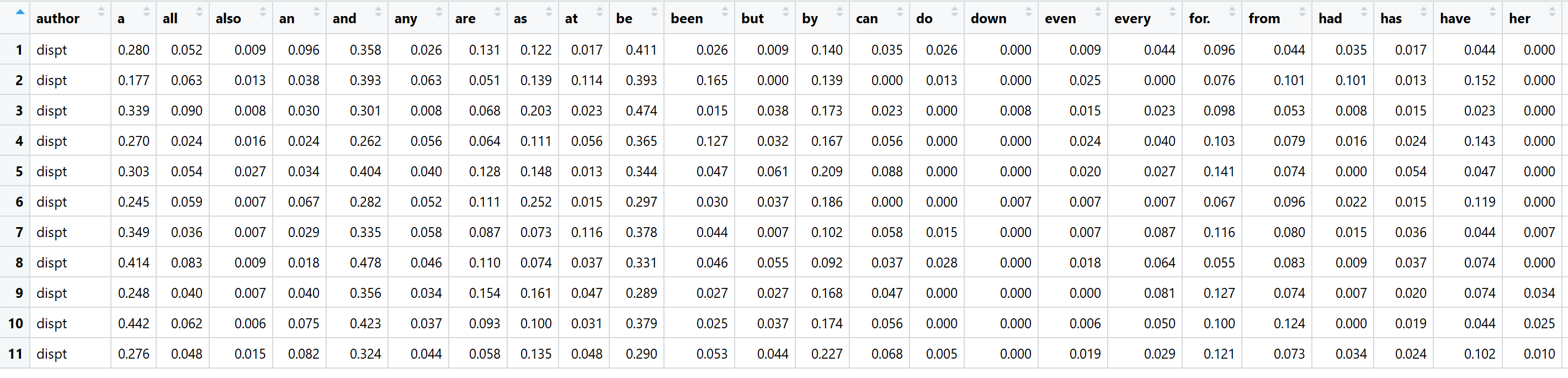


**Results**

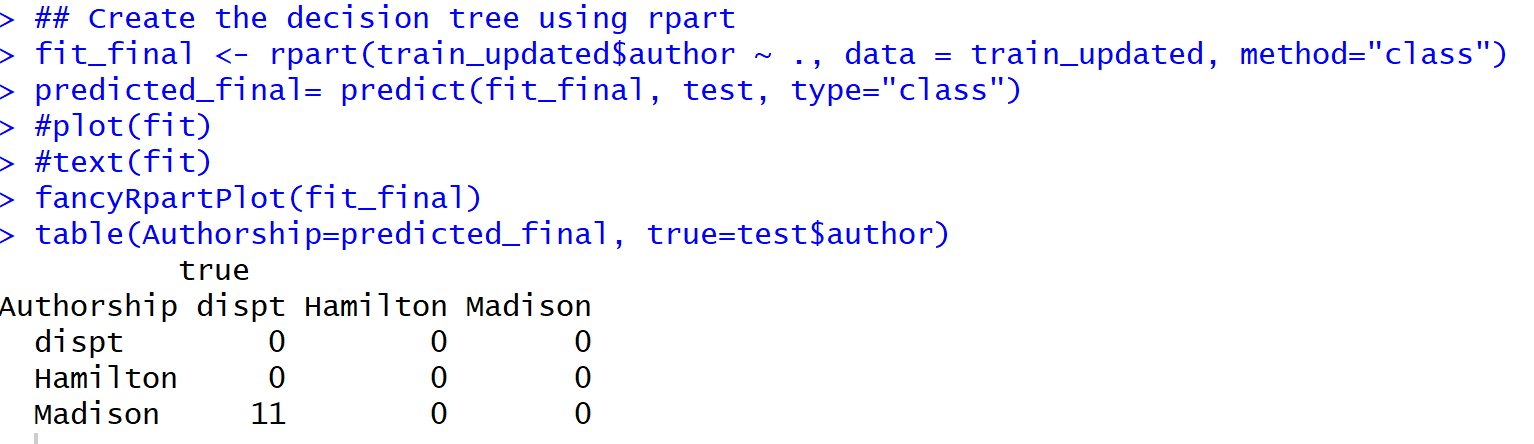
Once different models and tried out and evaluated, the best model and decision tree is selected to predict the actual test data which has the disputed files.



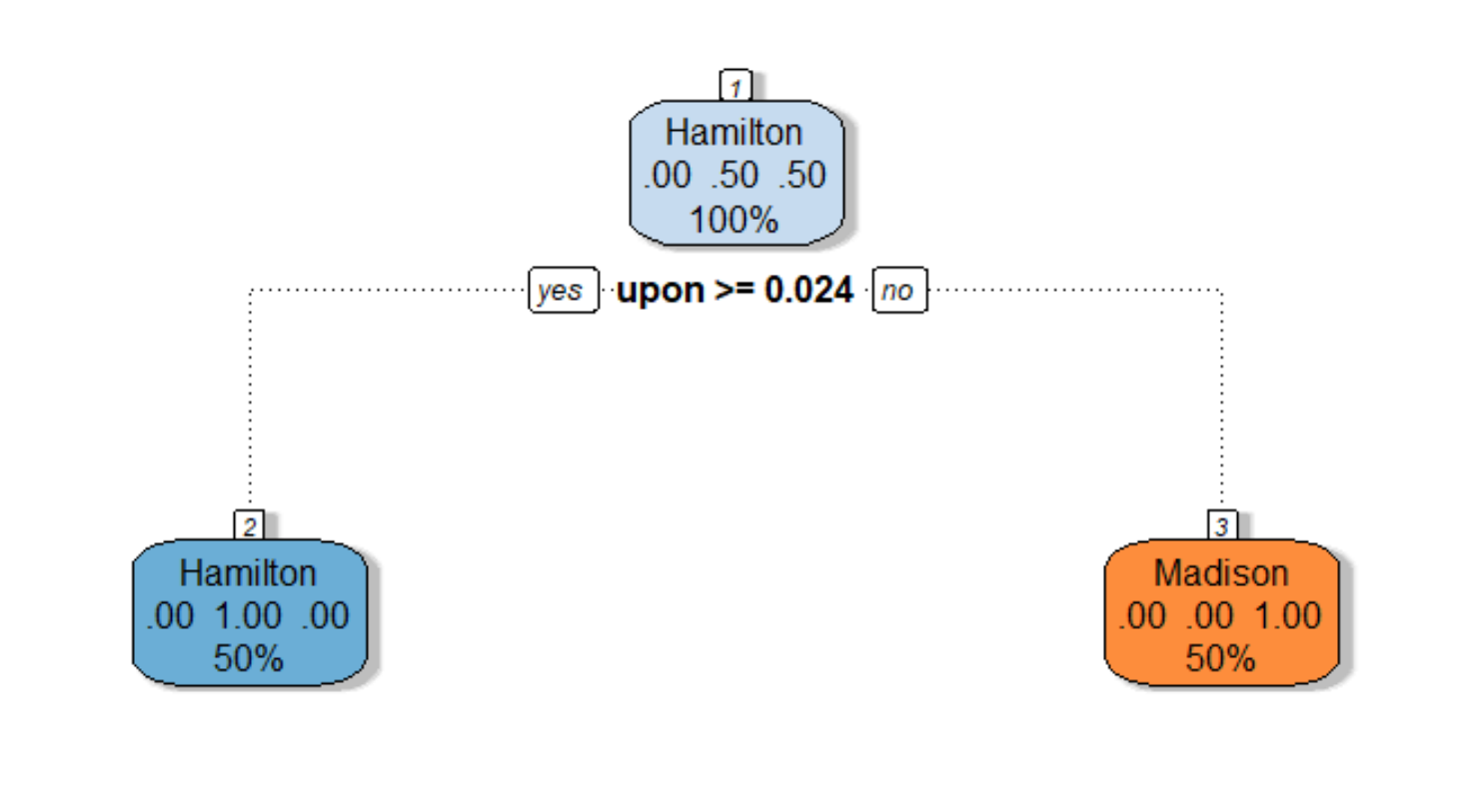
A quick view of the test data shows that it only has the disputed files.



The model is applied to this original test dataset and a confusion matrix is calculated to see the predictions. The below output shows that all the 11 disputed files are classified as Madison’s.



The decision tree is also visualized once again to see which attribute it selected to predict the disputed files.



Comparing the results of the classification technique with the clustering algorithm used in the previous paper, the decision tree algorithm predicted better. It was able to classify all files into a category as opposed to the clustering algorithm which predicted only some of the files. The clustering algorithm was not able to successfully cluster 5 out of the 11 documents. Whereas, the decision tree algorithm was able to classify all 11 files into one label.

The nice thing about decision trees is that they are easy to explain to the end users. This is primarily because they reflect the way common people think about how decisions are made in real life – via a set of binary choices based on appropriate criteria. That said, in many situations, decision trees turn out to be unstable. Small changes in the dataset can lead to wildly different trees. It turns out that this limitation can be addressed by building a variety of trees using different starting points and then averaging over them which is how random forests technique operate upon. Employing different techniques for different scenarios will be the best way to go about this.

**The idea of using cluster and classification analysis for text mining developed from the pioneering work done by Mosteller and Wallace in the 1960’s. They analyzed the frequency distributions of high frequency words in the Federalist papers to draw their conclusions. An important observation to note from their work is the fact that they not only used statistics but involved domain experts in their studies to get different insights.** Also, the results from the decision tree model are close to the results found by these gentlemen which further speaks to the accuracy of these results.

**Conclusion**

**After thorough cleaning, processing and analysis of the data effectively, it can be concluded that the author of the famously disputed federalist papers is James Madison and not Alexander Hamilton. Had these techniques been discovered during the era of the dispute, matters could have been solved quickly and amicably. However, these techniques can be applied now and can easily find out and resolve plagiarism or copyright issues. Using some of the powerful techniques of data mining, valuable information can be gained in various applications. Without these techniques, manual analysis would have taken an insurmountable amount of time and energy to do the same. It is also impossible to capture this level of insight due to the sheer volume and complexity of the data. Not only do these techniques help to identify trends and patterns that cannot be seen with a naked eye, it also helps to extract useful information from both structured and unstructured data.**