Real vs. Fake News Classification Report

By

Allison Beatty (yxn127)
Joshua Ching (mrf145)
Ruyi Liang (hua075)
Ian Scarff (iie728)

The University of Texas at San Antonio
DA - 6813 - 1B6

Background

For just over the past decade, the internet and social media have connected the world in more ways than ever. Just about anyone all over the world has the ability to interact, post, and gain information from social media. In addition to common people, companies of all sizes have utilized social media platforms such as Facebook, Twitter, and Instagram to market its products and services as well as to interact with its customers. One market that relies heavily on the internet and social media is the news media. Over the past few years, the number of people getting information from televised news networks as well as printed newspapers has decreased exponentially while the number of people getting information from social media has increased. As a result, mainstream news networks and organizations such as CNN, Fox News, The New York Times, and many more have focused on gathering readers online through social media. However, given all the benefits social media brings with regards to the spread of information and news, many people and some organizations have also spread false information and narratives of hate, fear, or distrust, and claim it as legitimate news. These types of news articles and stories have commonly been classified under the term "Fake News."

Motivation

With the prevalence of Fake News on social media platforms, it has become hard for everyday people to trust what they see on these platforms. For example, Facebook removed its own "trending news feed" section after it was found that a lot of the stories that appeared there were fake. The reason for this is because this news feed was populated by news articles that had the most user interactions (likes, comments, and shares). Now, not only do news organizations have to compete with each other for readers on social media, but they also have to compete with Fake News. Our analysis aims to develop a model to classify a news article as "Real" or "Fake" based on the article's text. Specifically, our analysis hopes to answer the following:

- **Is it possible to classify the type of news article based on a set of 100 unique words?** If it is possible to classify a news article based on 100 words, with a high prediction accuracy, this would be a simplistic way for social media companies to identify potential Fake News and remove them from its platforms. This would be beneficial to social media companies because they would be able to display credibility as well as show safety from false information. This would in turn potentially draw in more users. This would also benefit news organizations because they would have less trouble competing with Fake News sources.
- **What kinds of words are important predictors for classification?** If these important words can be identified, then social media companies can communicate to users these words so that they can better identify when they are looking at false information and report it.
- **Is the time of year an important classifier?**

If time of year is an important classifier, then social media companies can focus more resources into combating Fake News when needed.

Review of Literature

For our analysis, we needed to use a mix of text mining and predictive modeling. These two methods have been combined many times in practice. In Chapter 7 of the textbook, *Text Mining in Practice with R*, various case studies provide examples of this. In one classification problem, a hospital used text mining of patient records to predict whether or not a patient would be readmitted. In a regression problem, various movie reviews were text mined to predict a movie's opening weekend revenue. In each of these case studies, an important data structure, called a Document Term Matrix (DTM), was utilized. A DTM is a matrix that describes the frequency of terms that occur in a collection of documents. For our analysis, this data structure was key.

Processing of Data

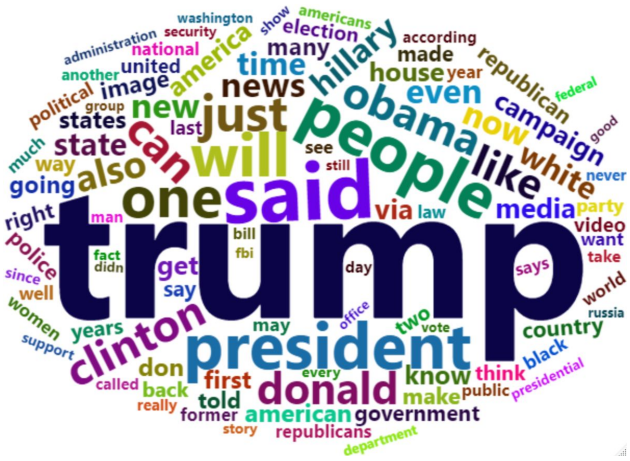
Our initial data consists of two files, True.csv and Fake.csv, from the website Kaggle.com on the webpage "Fake and real news dataset." The first file contains 21,417 observations, where each observation is a news article that has been designated "real" news, and the second file contains 23,481 observations, where each observation is a news article that has been designated "fake." Each file has attributes "title" (the title of the article), "text" (the text of the article), "subject" (the general story class of the article), and "date" (the date when the article was posted). Further documentation of the data can be found in the Appendix.

The two datasets were generally clean with the exception of a few observations that had either the wrong date format, had blank text, or had data in the wrong column. These formatting errors were corrected. The "subject" variable was not used in this project because some of the categories were clearly biased or not logical. For example, one classification was "left-news." Together, these datasets span a time frame from March 31, 2015 to December 31, 2017. One thing to note is that there were no observations of real news during 2015.

Prior to modeling, the following data preprocessing step were taken:

1) combined the two separate files into one
2) created a new variable, "Type," which is the category of the news type (real or fake)
3) created a month variable by extracting the month from the "date" variable
4) removed a few duplicate columns
5) text mined the "text" of the FakeNews articles in order to find the most common words by utilizing the tools in the R package, "tm"

By text mining the Fake News articles, we hope to find unique words that appear more often in Fake News than they do in Real News. This in turn would create some separation between the two classes. To determine what words classified as unique and that provided information, each article's text had all english stopwords removed as well as all numbers and punctuation removed. From text mining, we extracted the top 100 most occuring words. These words are summarized in the wordcloud below, where the size of a word indicates the frequency of that word.



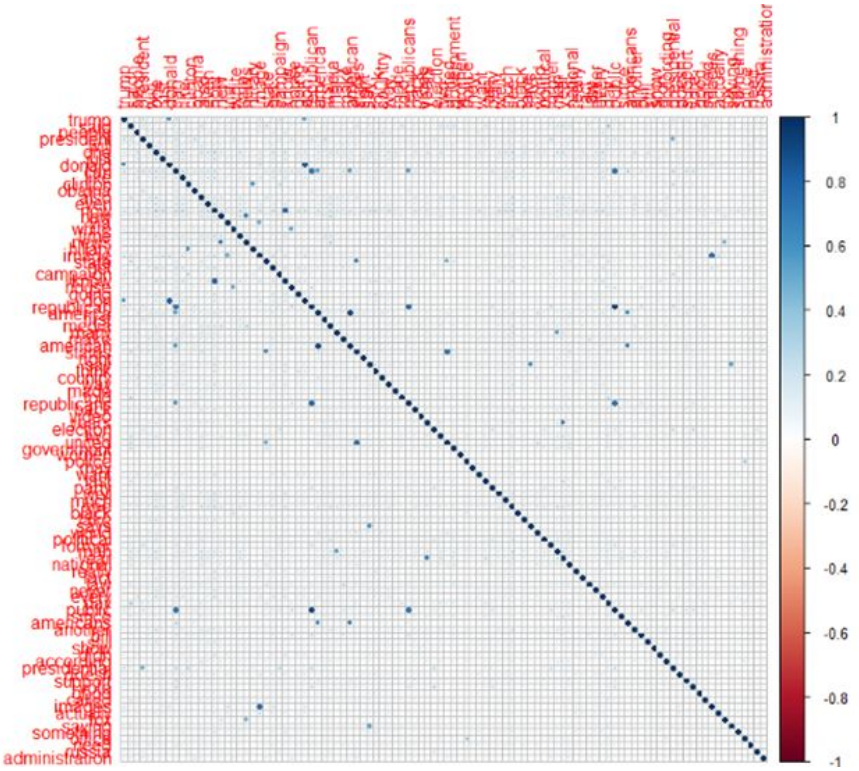
We then took these words and turned them into columns, ordered from most frequent to least frequent. The entries in these columns were the amount of occurrences of a word in a particular article, a DTM. Finally, we combined this DTM of 100 words with our new "Type" variable and our new "Month" variable to create our final dataset, which is summarized in the table below. The final dataset consisted of 38,604 observations with 102 columns, with our target variable being "Type" and our predictor variables being "Month" and the 100 unique word columns. Further documentation of the data can be found in the Appendix.

Type	month	trump	said	people	president	will	one	....
fake	3	0	0	1	3	0	4	....
fake	**3**	**0**	**0**	**2**	**0**	**1**	**1**	**....**
fake	3	0	3	0	1	1	1	....
fake	**3**	**0**	**1**	**1**	**3**	**3**	**5**	**....**
fake	3	0	3	0	0	1	2	....
fake	**3**	**0**	**0**	**0**	**1**	**0**	**1**	**....**
....	....	....	....	....	....	....	....	....
real	**5**	**4**	**1**	**0**	**3**	**0**	**3**	**....**
real	5	15	15	1	8	4	5	....
real	**5**	**20**	**13**	**1**	**13**	**2**	**6**	**....**
real	5	1	3	0	2	0	0	....
real	**5**	**4**	**3**	**1**	**2**	**0**	**2**	**....**
real	5	14	15	1	4	0	1	....
....	**....**	**....**	**....**	**....**	**....**	**....**	**....**	**....**

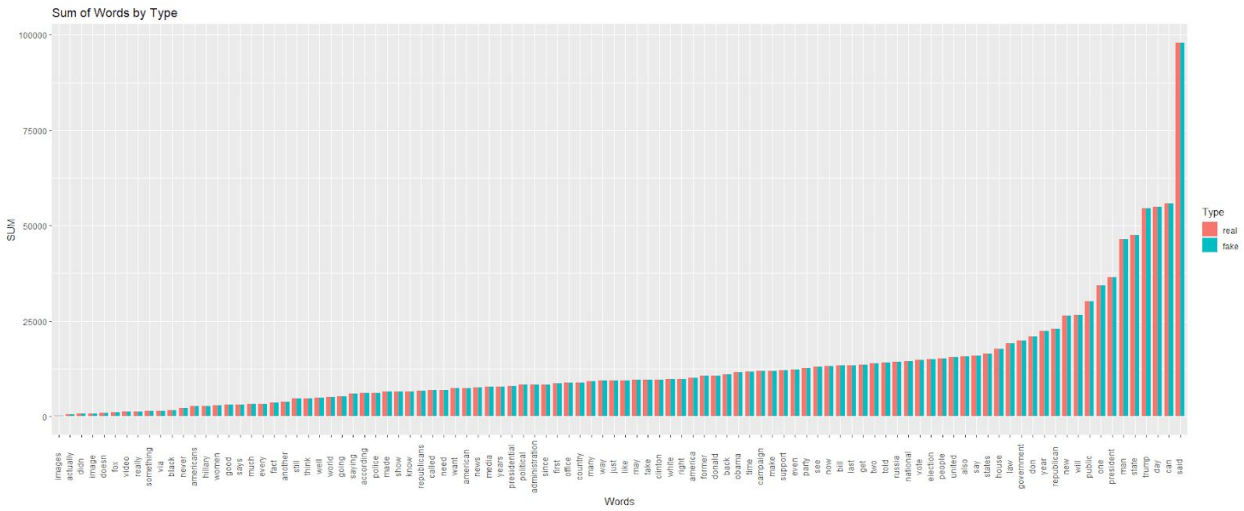
Exploratory Analysis

We first looked at the correlations between word frequencies. These correlations are summarized in the graph below. Based on this graph, we can see that there are some strong correlations between some words, but overall there are not many. One possible reason why we see such high correlations could be that these words appear next to each other in a sentence naturally, such as full names and titles (i.e. Hilary Clinton, Donald Trump, President). Graphing correlations from Fake News, Real News, and individual months yielded the same pattern of a few strong correlations, but very little in total.

Word Frequency Correlation Pot (Full Data)



Next, we examined the total frequencies of each word based on news type. This is summarized in the graph below. Bars in red signify Real News and bars in green signify Fake News. Based on this graph, the total frequencies for each word are similar for each news type. We can also see that some words are used more frequently than other words for both news types.



Analysis Methodology

For our analysis, four classification modes (Logistic Regression, Decision Tree, Random Forest, and Neural Network) were constructed and compared against each other to determine which is optimal for predicting Real vs. Fake News. To build each model, the *train* function from the R package "caret" was used in addition to a control function using the *trainControl* function. All models were built using the same random seed (210). To split the data into training and testing datasets, stratified random sampling by means of the *createDataPartition* function was utilized to allocate 80% to training (30,884 observations) and 20% to testing (7,720 observations). In each model, the target variable was 'Type," and the predictors were the 100 word variables and the "Month" variable. Models were compared based on their test accuracy, test error rate, ROC curves, and the area under these curves. For predictions, if the predicted probability of the response "fake" was greater than or equal to 0.75, then the model predicted "fake," "real" otherwise.

Logistic Regression:

The first model used in this analysis was a logistic regression model. This is a parametric model, meaning that it has some underlying assumptions about the data that needed to be met. These assumptions are as follows:

1. For binary logistic regression, the dependent variable must be binary.
2. Individual observations must be independent.
3. There must be little to no multicollinearity between predictors.
4. There is a linear relationship between the independent predictors and the log odds.

However, in reality, there is no guarantee that the data follows these assumptions perfectly. This may cause logistic regression to be ineffective. Therefore, we then decided to run some nonparametric models including: Decision Tree, Random Forest, and Neural Network.

Decision Tree:

The second model used in this analysis was a decision tree model. The goal of a decision tree is to partition the data into smaller, more similar groups while maximizing accuracy. It does this by splitting the data into nodes. These nodes can then create new nodes. In the "caret" package, a regular decision tree has one tuning parameter called cp. This is a complexity parameter used to control and find the optimal size of a tree.

Random Forest:

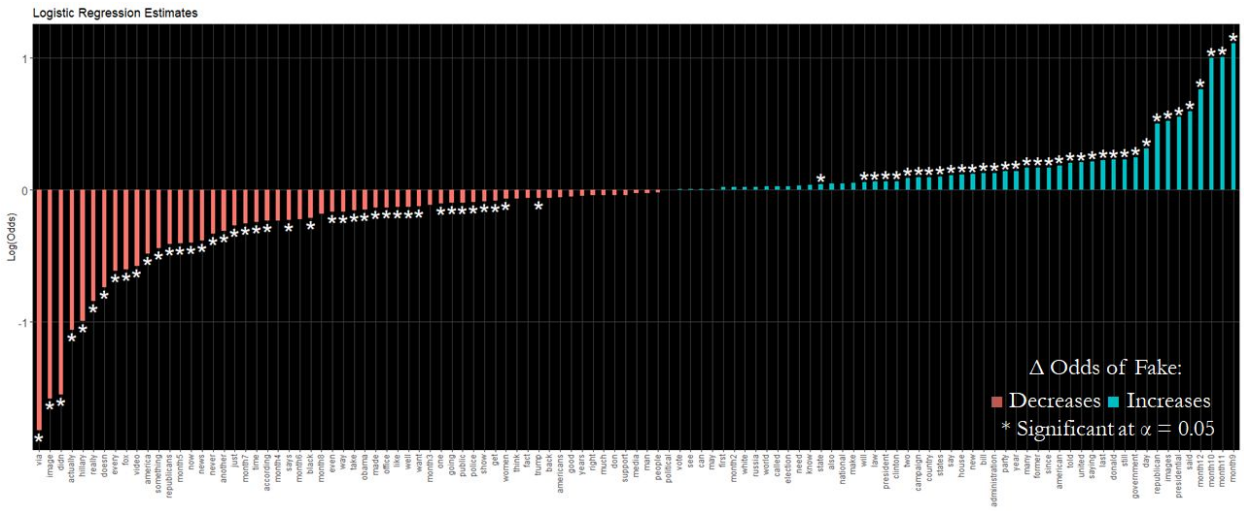
The third model used in this analysis was a random forest model. Building on a decision tree, a random forest (RF) model creates a large number of these trees (i.e. a forest). For a new observation, each tree casts a vote for what it should be classified as. The proportion of votes across the forest creates the predicted probability vector. In addition, RF models are able to de-correlate the trees in the forest. In the "caret" package, the RF model has one tuning parameter, mtry. This is the number of randomly selected predictors to choose from at each split. For this study, the number of trees created in the forest was set to 150 with a node size of 50.

Neural Network:

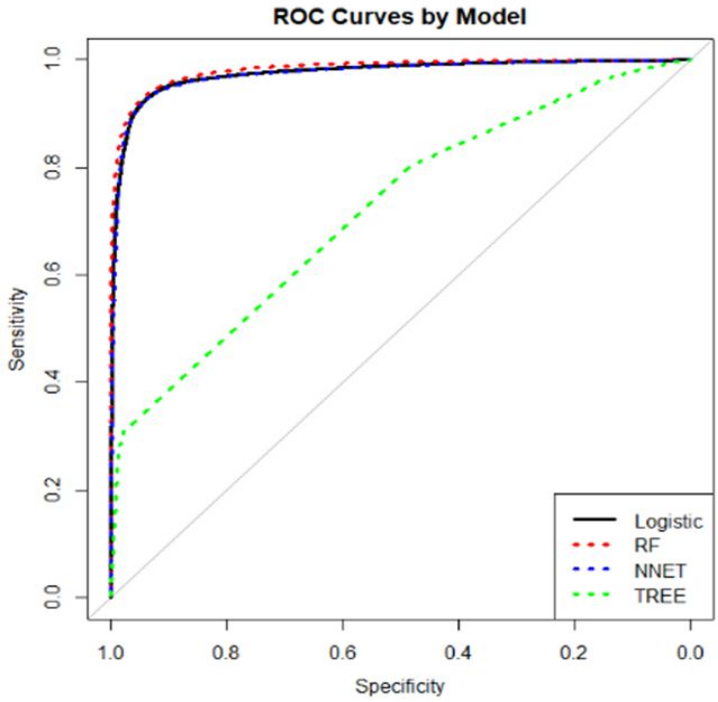
The fourth and final model used in this analysis was a neural network. A neural network (NNET) works by constructing an architecture of layers. The first layer is composed of the individual predictors. From this first layer, multiple hidden layers are created from transformations of linear combinations of the predictors. These hidden layers are then used to make a prediction. In the "caret" package, the neural network model has two tuning parameters, size and decay. The optimal tuning parameters for all models are summarized in the Appendix.

Results

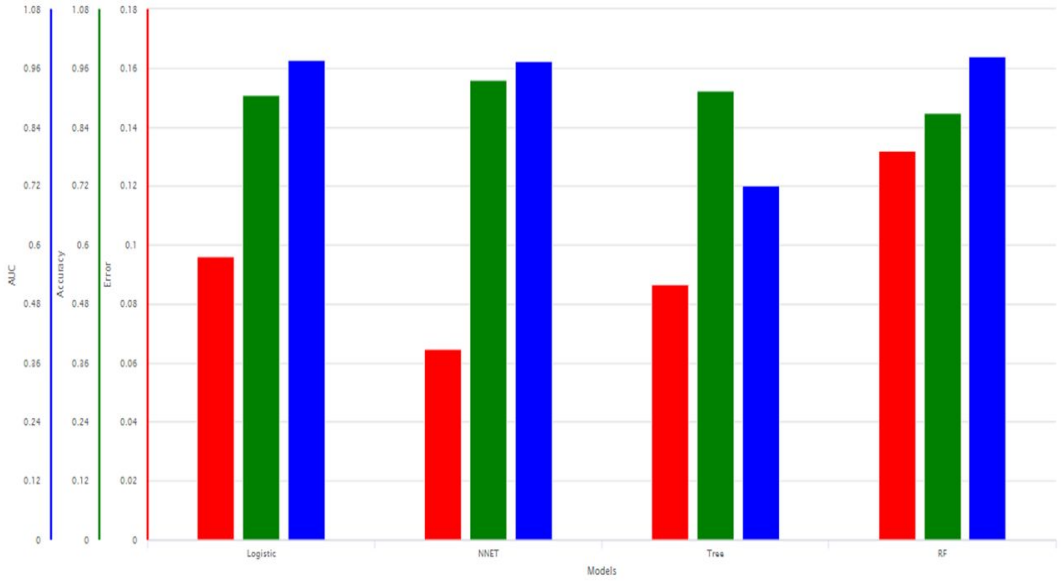
First, we will discuss the estimated beta values of the logistic regression model. The beta values in logistic regression are log(odds). The graph below summarizes the output of the logistic model. Estimates in blue signify an increase in the odds of an observation being fake news, while the red signifies a decrease in the odds of an observation being fake news. The white stars indicate that a variable is statistically significant at a 95% Confidence Level. From this graph, we can see many words and a few months have extreme beta values at each end. For example, month 9 has an estimated beta value of 1.108956. This means that on average, with all other variables held constant, news articles posted in month 9 have higher odds of being fake than in the reference level, in this case month 1. On the opposite end of the graph, the word "via" has an estimated beta value of -1.825141. This means that on average, with all other variables held constant, for a one unit increase in the frequency of the word "via," the odds of a new article decrease by a multiple of 0.1612.



Next we will compare the models. First, the graph of the ROC Curves below show that the logistic regression, random forest, and neural network models have similar curves. These three models well outperformed the decision tree model.

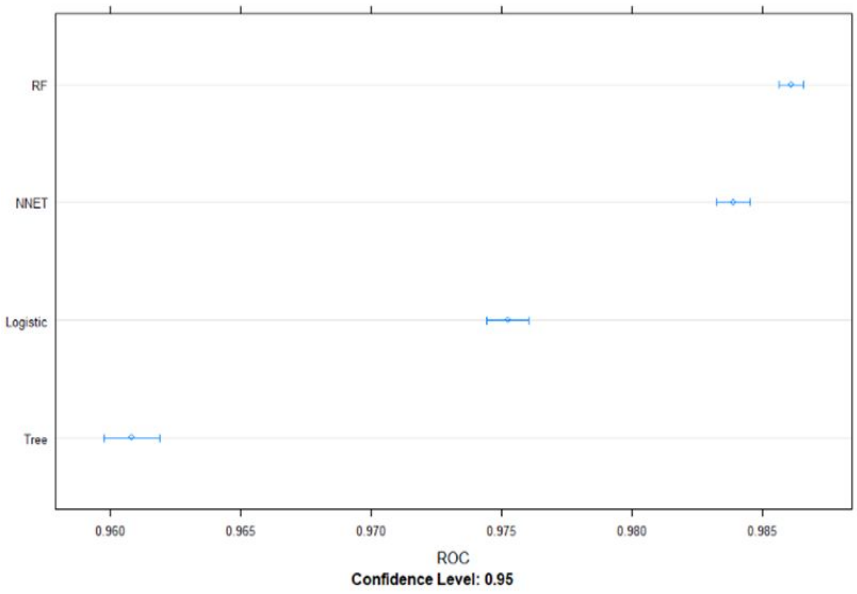


Second, the graph and table below summarizes the fit statistics of the four models. Bars in blue represent a model's AUC (area under the curve), bars in green represent a model's test accuracy, and bars in red represent a model's test error rate.



Models	Test Accuracy	Test Error	AUC
Logistic	0.9040155	0.09598446	0.9752146
NNET	0.9352332	0.06476684	0.9743436
Tree	0.9132124	0.08678756	0.7209131
RF	0.8680052	0.13199482	0.9833086

From the graph and table above, we can see that the neural network had the highest test accuracy, closely followed by the logistic and decision tree models, while the random forest model had the worst accuracy. The random forest did however have the highest AUC, closely followed by the logistic and neural network models, with the decision tree having the worst. Based on this, the logistic, neural network, and random forest look to be similar. However, based on the resamples of the ROC statistic summarized below, the random forest model appears to be the best. One Sample t-tests for the differences between each of these statistics have p-values far less than 0.05. Note, these resamples are based on the training dataset and may differ from the test results.



However, when looking at the confusion matrices of the four models, we notice something important. The random forest has the highest rate of predicting Real News when in reality it is Fake News (a Type II Error). For our research purposes, this is not good. The neural network model has the lowest rate of this. These confusion matrices are displayed below. All other model outputs (Accuracy graphs, Variable Importance, individual ROC Curves, etc.) can be found in the Appendix.

Logistic Regression			
		True Value	
		fake	*real*
Predicted Value	*fake*	2817	76
	real	665	4162

Decision Tree			
		True Value	
		fake	*real*
Predicted Value	*fake*	3023	211
	real	459	4027

Neural Network			
		True Value	
		fake	*real*
Predicted Value	*fake*	3091	109
	real	391	4129

Random Forest			
		True Value	
		fake	*real*
Predicted Value	*fake*	2466	3
	real	1016	4235

Conclusions & Recommendations

Based on the results from our analysis, we conclude that the neural network model is the optimal model for predicting whether or not a news article is fake or real. It has a similar ROC/AUC as the logistic and random forest models, has the highest test accuracy at a 75% prediction threshold, and has the lowest false negative rate. However, it does have some disadvantages. First, it has the second highest false positive rate (Type I Error). Second, neural networks are very complex models, which lead to higher processing times compared to other methods. Third, it has a "Black Box" nature, meaning that while it is good at prediction, it is highly uninterpretable. Furthermore, using our results, we return to our beginning analysis goals:

- **Is it possible to classify the type of news article based on a set of 100 unique words?**
    - Using the neural network model, it is possible to predict the type of news based on the 100 most common words found in Fake News articles.
- **What kinds of words are important predictors for classification?**
    - Using a combination of variable importance plots (in Appendix) from the neural network model along with the estimated beta values of the logistic regression model, it is possible to see what variables are important for classification.
- **Is the time of year an important classifier?**
    - Using a combination of variable importance plots (in Appendix) from the neural network model along with the estimated beta values of the logistic regression model, it is possible to see the effects of the month in which an article was posted.

In conclusion, we can make the following recommendations. First, social media companies should use neural network models to classify news articles as real or fake. Second, social media companies should use a logistic regression model to understand how various words affect the probability/odds of an article being real or fake. From this, they can inform their users so they can tell if they are looking at possible false information. Third, social media companies should set up a flagging system for articles whose prediction probabilities are near the prediction threshold, thereby mitigating a Type II error. Fourth, social media companies should set up an appeal system for human review for organizations who feel their news articles have been misclassified, thereby mitigating a Type I error. For future studies on this topic, we recommend exploring more words and word combinations, extracting the news source as a predictor, and determining the article's topic and using it as a predictor.

Works Cited

Bisaillon, Clément. Fake and real news dataset. 2020 March 26. 7 April 2020. <https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset>.

Ingber, Sasha. Facebook Is Scrapping Its Troubled 'Trending' News Section. 1 June 2018. https://www.npr.org/sections/thetwo-way/2018/06/01/616289583/facebook-is-scrapping-its-troubled-trending-news-section.

Kwartler, Ted. “Predictive Modeling: Using Text for Classifying and Predicting Outcomes.” Text Mining in Practice with R, John Wiley & Sons, Ltd, 2017, pp. 209–36, doi:10.1002/9781119282105.ch7.

Appendix

Variable Descriptions

The raw data: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset

Ture.csv (as found)

- title: The title of the news article
    - Data Type: Text
- text: The text of the news article.
    - Data Type: Text
- subject: The subject of the news article.
    - Data Type: Categorical
        - Levels:
            - politicsNews
            - worldNews
- date: The date at which the article was posted.
    - Data Type: Date

Fake.csv (as found)

- title: The title of the news article
    - Data Type: Text
- text: The text of the news article.
    - Data Type: Text
- subject: The subject of the news article.
    - Data Type: Categorical
        - Levels:
            - a future nuclear family
            - and so is Mr. Katzenbach
            - claimed that hundreds of alternative media websites were producing fake news and conspiracy stories and therefore were unreliable as information sources. It wasn t long before the establishment began referencing these politicized lists
            - decided not to seek re-election. While much of the mainstream media waxed poetic about his 30-year career
            - fell 5.6 percent Monday. Wynn Resortsslipped 1.2 percent. Las Vegas Sands fell as much as 2.1 percent before closing higher. LIVE DRILL Las Vegas has been at the forefront of active shooter training. (Image Source: sinclairstoryline)Las Vegas Active Shooter Drills Back in 2014
            - high taxes
            - Karl Marx is either a villain or a hero of social engineering

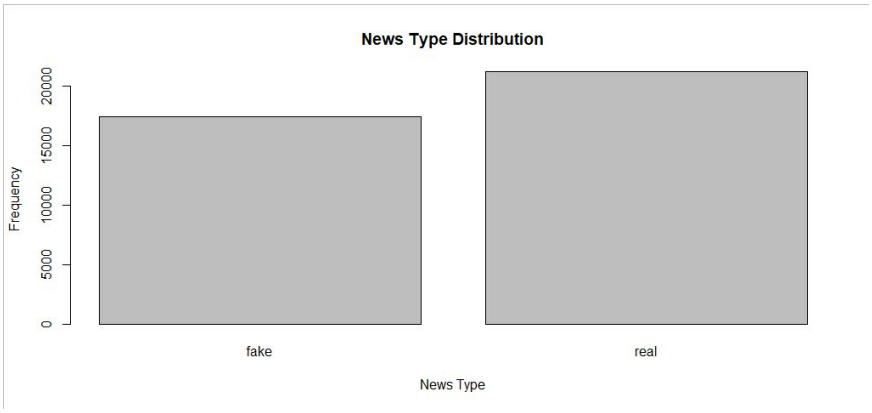
                - of which Soros is a major financier.Mercy Corps: Vis a vis the Arab-Israeli conflict
                - Politics According to the Bible and (with Barry Asmus) The Poverty of Nations: A Sustainable Solution.Via: Townhall"
                - state systems with outsized pretensions to power have reacted to their environments in two ways. The first strategy
                - the defense industry
                - Government News
                - left-news
                - Middle-east
                - News
                - politics
                - US_News
                - (blank)
- date: The date at which the article was posted.
    - Data Type: Date

NewsInfo.csv (created dataset)

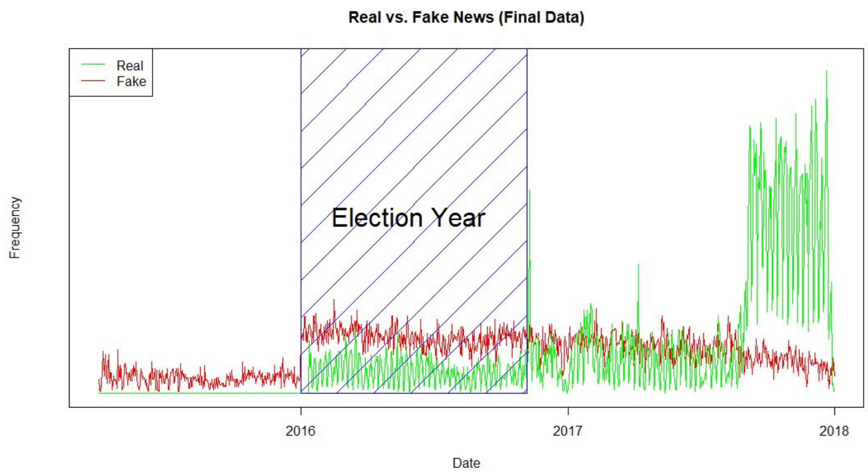
- Type: The class of news article.
    - Data Type: Categorical
        - Levels:
            - real
            - Fake
- month: The month in which the news article was posted.
    - Data Type: Categorical
        - Levels:
            - 1
            - 2
            - 3
            - 4
            - 5
            - 6
            - 7
            - 8
            - 9
            - 10
            - 11
            - 12
- 100 Word Variables: The number of occurrences of each word in each article.

- Data Type: Numeric
  - **Variable Names:** trump said people president will one just donald can like clinton obama also even now new via white time news hillary image state get campaign know house going don republican america first media many make american states right say think country way made told republicans back video years election two united government women police may want last party well much see black take says world political former man year national really fact law never every day public since americans another bill still show didn according presidential doesn support vote good called images actually fox saying something office need russia administration

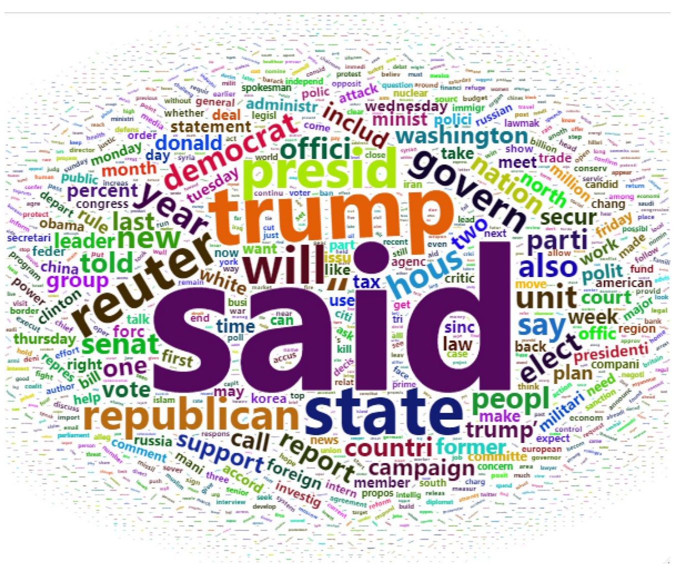
Exploratory Analysis: Plots



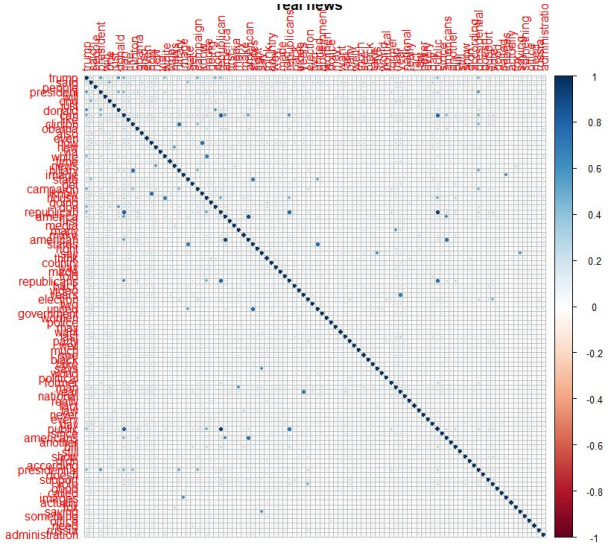

```
fake  real
17413 21191
```



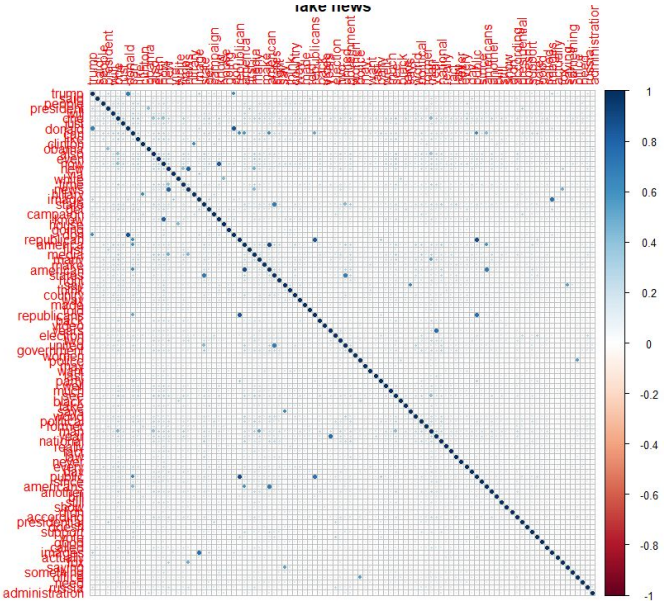
Real News Word Cloud (Final Data)



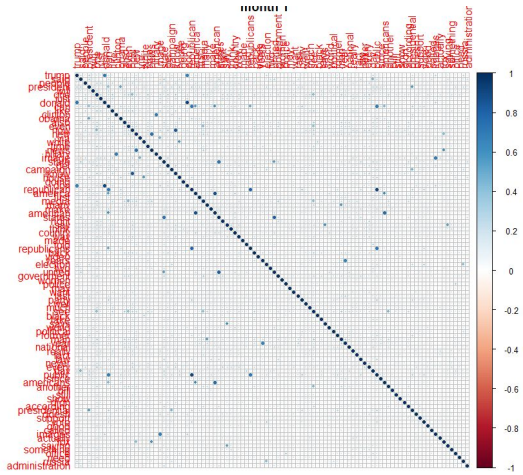
Real news



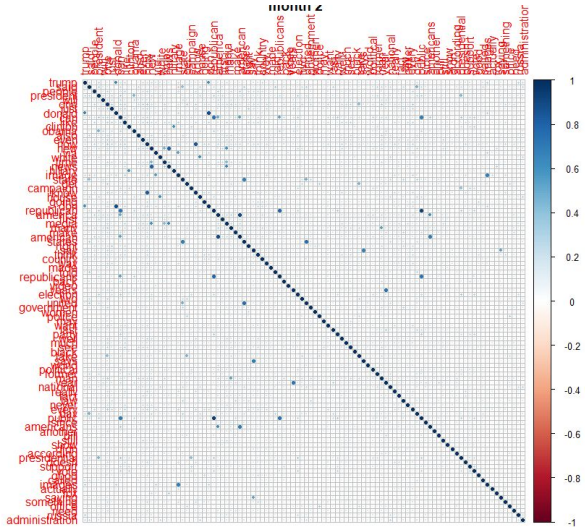
Fake news



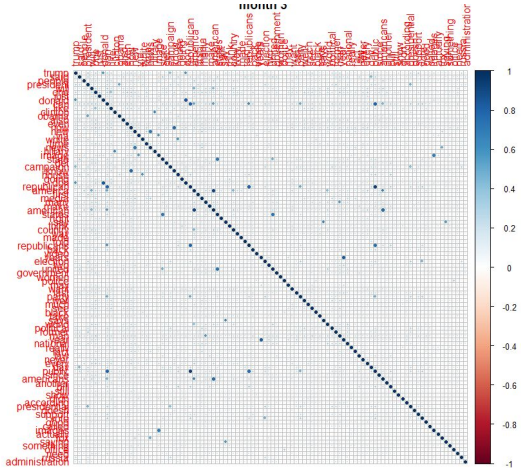
Month 1



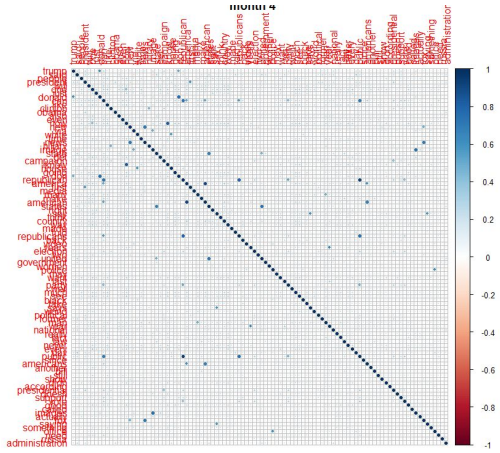
Month 2



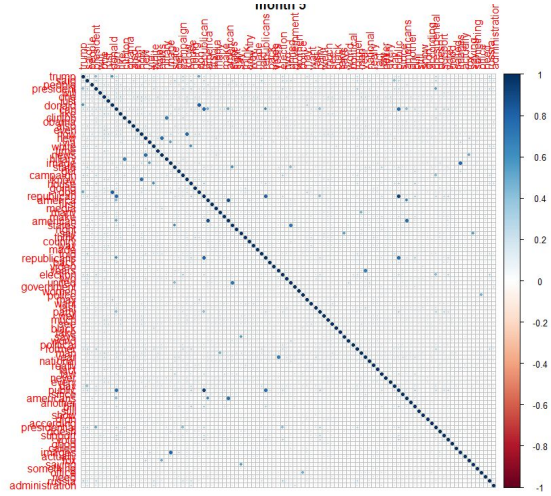
Month 3



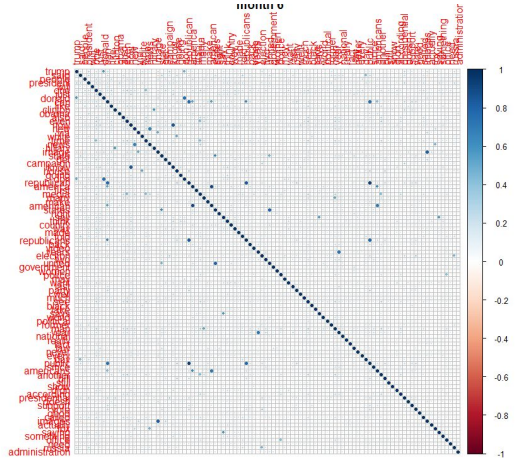
Month 4



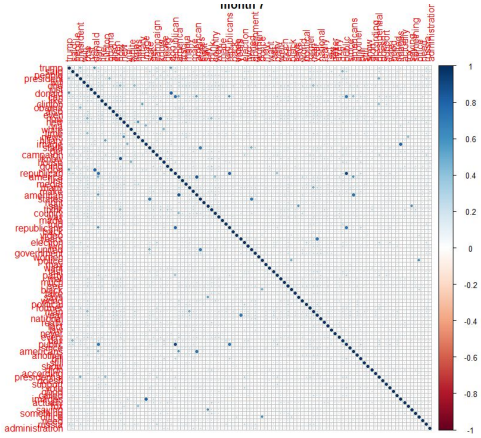
Month 5



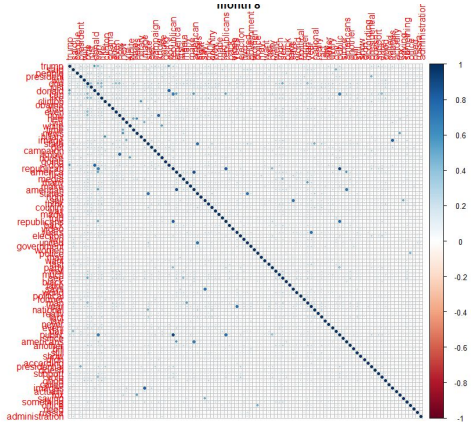
Month 6



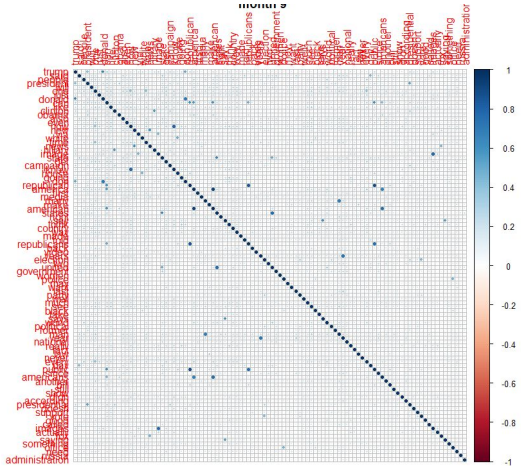
Month 7



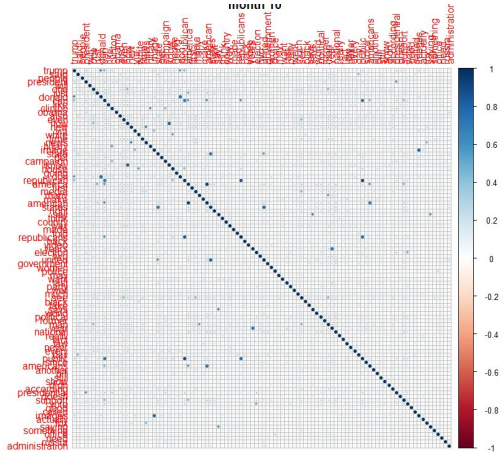
Month 8



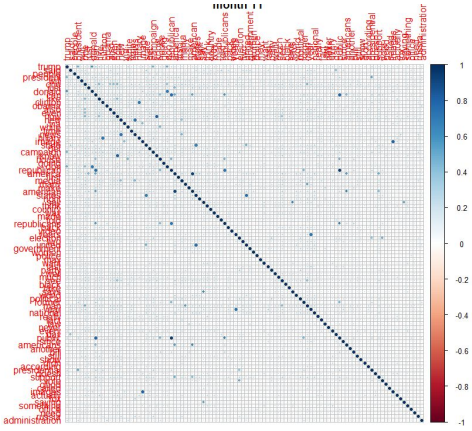
Month 9



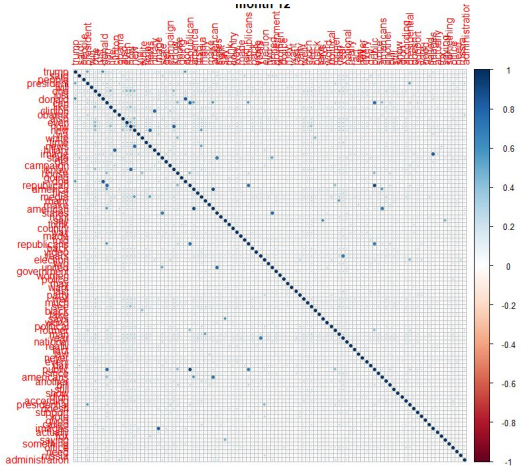
Month 10



Month 11



Month 12



Model Outputs

Optimal Tuning Parameters

Model	Optimal Hyperparmeter(s)
Logistic	NA
NNET	size = 8, decay = 0.5
Tree	cp = 0
RF	mtry = 4

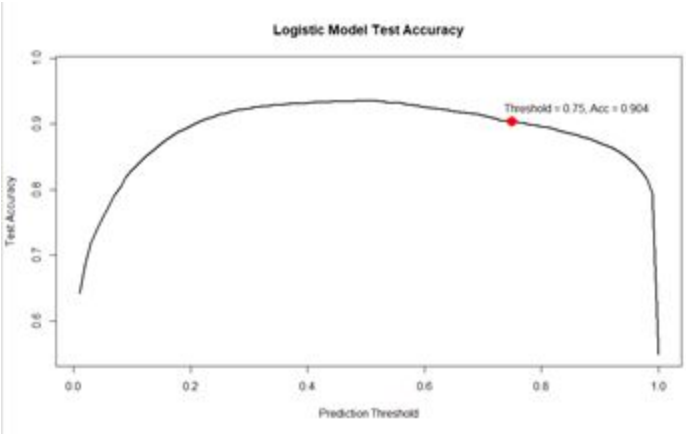
Logistic Regression

```
Generalized Linear Model

30884 samples
  101 predictor
    2 classes: 'fake', 'real'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 23164, 23164, 23164, 23164, 23164, 23164, ...
Resampling results:

  ROC        Sens      Spec
  0.9752436  0.907869  0.9542237
```

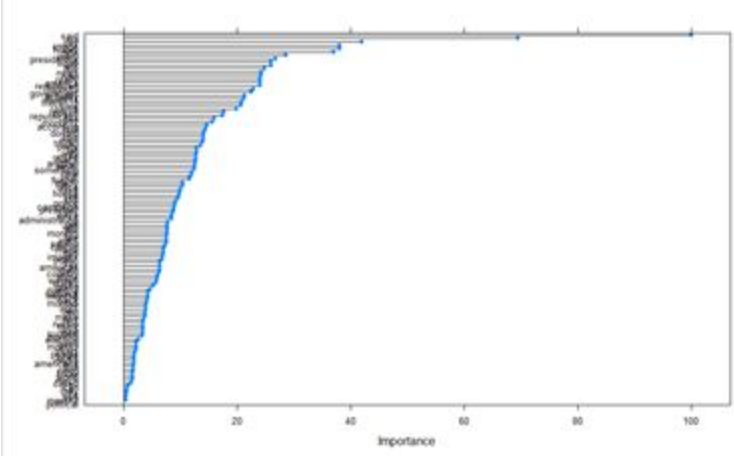
```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.7300  -0.0729   0.0219   0.2898   8.4904

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.296629   0.096399  -3.077 0.002090 **
month2       0.020989   0.129393   0.162 0.871137
month3      -0.114950   0.124953  -0.920 0.357602
month4      -0.234057   0.128135  -1.827 0.067754 .
month5      -0.406591   0.126680  -3.210 0.001329 **
month6      -0.220875   0.127701  -1.730 0.083696 .
month7      -0.253837   0.128016  -1.983 0.047384 *
month8      -0.179114   0.128701  -1.392 0.164011
month9       1.108956   0.115519   9.600  < 2e-16 ***
month10      0.998359   0.114333   8.732  < 2e-16 ***
month11      1.005094   0.112669   8.921  < 2e-16 ***
month12      0.762301   0.116476   6.545 5.96e-11 ***
trump       -0.060295   0.010260  -5.877 4.19e-09 ***
said         0.594063   0.014140  42.014  < 2e-16 ***
people      -0.019843   0.024587  -0.807 0.419642
president    0.062040   0.016548   3.749 0.000177 ***
will         0.058091   0.017889   3.247 0.001165 **
one         -0.103615   0.015461  -6.702 2.06e-11 ***
just        -0.270823   0.024821 -10.911  < 2e-16 ***
donald       0.227872   0.038140   5.975 2.31e-09 ***
can          0.006509   0.017860   0.364 0.715536
like        -0.131271   0.031873  -4.119 3.81e-05 ***
clinton      0.063172   0.020582   3.069 0.002146 **


obama       -0.152850   0.018266  -8.368  < 2e-16 ***
also         0.048100   0.033649   1.429 0.152875
even        -0.167047   0.028319  -5.899 3.66e-09 ***
now         -0.399652   0.039365 -10.152  < 2e-16 ***
new          0.118665   0.023205   5.114 3.16e-07 ***
via         -1.825141   0.062494 -29.205  < 2e-16 ***
white        0.022715   0.029402   0.773 0.439777
time        -0.243692   0.028079  -8.679  < 2e-16 ***
news        -0.383636   0.037498 -10.231  < 2e-16 ***
hillary     -0.994616   0.062237 -15.981  < 2e-16 ***
image       -1.580927   0.098617 -16.031  < 2e-16 ***
state        0.043426   0.016262   2.670 0.007577 **
get         -0.082809   0.025015  -3.310 0.000932 ***
campaign     0.093924   0.024838   3.782 0.000156 ***
know         0.038362   0.056084   0.684 0.493976
house        0.116131   0.027741   4.186 2.84e-05 ***
going       -0.098446   0.044710  -2.202 0.027675 *
don         -0.040139   0.024884  -1.613 0.106725
republican   0.502566   0.049837  10.084  < 2e-16 ***
america     -0.484623   0.047979 -10.101  < 2e-16 ***
first        0.020076   0.027571   0.728 0.466515
media       -0.024781   0.032092  -0.772 0.439995
many         0.167712   0.038364   4.372 1.23e-05 ***
make         0.055318   0.034883   1.586 0.112788
american     0.185223   0.068723   2.695 0.007035 **
states       0.098283   0.040810   2.408 0.016027 *
right       -0.041640   0.025520  -1.632 0.102746
say          0.110953   0.041141   2.697 0.006999 **
think       -0.064472   0.045099  -1.430 0.152842
country      0.096566   0.036629   2.636 0.008381 **
way         -0.165535   0.030626  -5.405 6.48e-08 ***
made        -0.136337   0.046105  -2.957 0.003106 **
told         0.202644   0.034268   5.913 3.35e-09 ***
republicans -0.413112   0.056261  -7.343 2.09e-13 ***
back        -0.059758   0.034076  -1.754 0.079492 .
video       -0.576265   0.052637 -10.948  < 2e-16 ***
years       -0.045017   0.049089  -0.917 0.359109
election     0.028192   0.025774   1.094 0.274036
two          0.088096   0.029907   2.946 0.003223 **
united       0.206821   0.038496   5.373 7.76e-08 ***
government   0.244847   0.025826   9.481  < 2e-16 ***
women       -0.067098   0.026968  -2.488 0.012844 *
police      -0.095032   0.024923  -3.813 0.000137 ***
may          0.006734   0.027096   0.249 0.803741
want        -0.124376   0.039136  -3.178 0.001483 **
last         0.225231   0.037002   6.087 1.15e-09 ***
party        0.139594   0.024613   5.672 1.41e-08 ***
well        -0.128759   0.039606  -3.251 0.001150 **
```

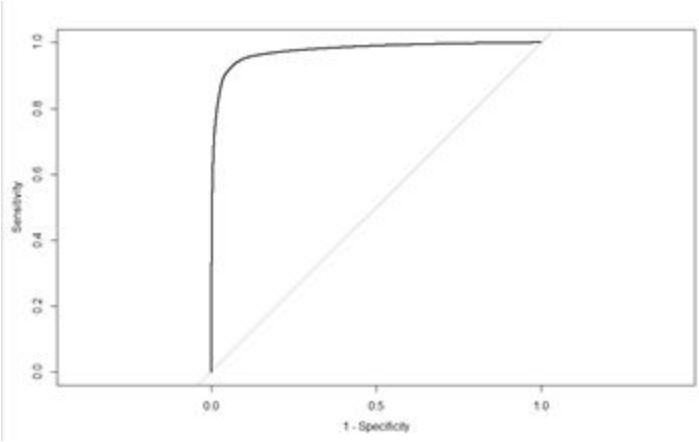
```
much            -0.040241   0.062581  -0.643 0.520209
see              0.005885   0.030520   0.193 0.847091
black           -0.215282   0.040239  -5.350 8.79e-08 ***
take            -0.155766   0.037118  -4.197 2.71e-05 ***
says            -0.229066   0.062336  -3.675 0.000238 ***
world            0.025454   0.038864   0.655 0.512495
political       -0.001397   0.037069  -0.038 0.969935
former           0.168305   0.034563   4.870 1.12e-06 ***
man             -0.023811   0.014650  -1.625 0.104096
year             0.143810   0.028896   4.977 6.46e-07 ***
national         0.049561   0.027100   1.829 0.067425 .
really          -0.844601   0.083562 -10.108  < 2e-16 ***
fact            -0.062315   0.042180  -1.477 0.139582
law              0.061618   0.019368   3.181 0.001466 **
never           -0.334620   0.063474  -5.272 1.35e-07 ***
every           -0.615729   0.051156 -12.036  < 2e-16 ***
day              0.314958   0.017831  17.664  < 2e-16 ***
public          -0.097126   0.033764  -2.877 0.004019 **
since            0.168488   0.047220   3.568 0.000359 ***
americans       -0.055766   0.072706  -0.767 0.443077
another         -0.311958   0.058490  -5.334 9.63e-08 ***
bill             0.124236   0.022922   5.420 5.96e-08 ***
still            0.230829   0.057498   4.015 5.96e-05 ***
show            -0.086221   0.034243  -2.518 0.011805 *
didn            -1.553918   0.099961 -15.545  < 2e-16 ***
according       -0.235061   0.037965  -6.192 5.96e-10 ***
presidential     0.551055   0.049005  11.245  < 2e-16 ***
doesn           -0.740846   0.099854  -7.419 1.18e-13 ***
support         -0.038746   0.027112  -1.429 0.152968
vote             0.004203   0.018546   0.227 0.820731
good            -0.051209   0.054738  -0.936 0.349517
called           0.027932   0.048472   0.576 0.564447
```

```
images           0.521956   0.179572   2.907 0.003653 **
actually        -1.061744   0.118609  -8.952  < 2e-16 ***
fox             -0.602840   0.057882 -10.415  < 2e-16 ***
saying           0.215685   0.067980   3.173 0.001510 **
something       -0.440889   0.084164  -5.238 1.62e-07 ***
office          -0.132254   0.030315  -4.363 1.29e-05 ***
need             0.032262   0.044683   0.722 0.470288
russia           0.023826   0.016647   1.431 0.152373
administration   0.127686   0.035844   3.562 0.000368 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 42518  on 30883  degrees of freedom
Residual deviance: 12075  on 30772  degrees of freedom
AIC: 12299

Number of Fisher Scoring iterations: 8
```



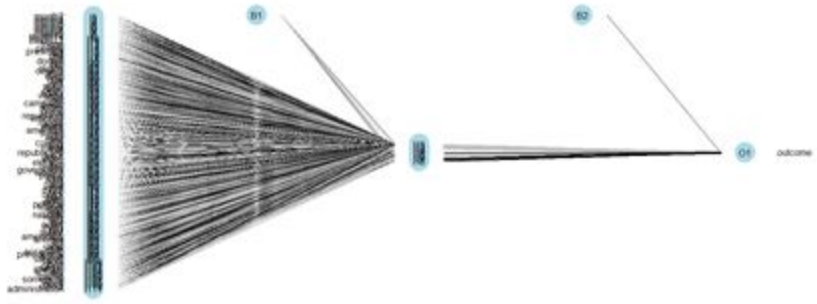
```
glm variable importance

  only 20 most important variables shown (out of 111)

             Overall
said          100.00
via            69.49
day            41.99
image          38.10
hillary        37.98
didn           36.94
every          28.58
presidential   26.70
video          25.99
just           25.90
fox            24.72
news           24.28
now            24.10
really         23.99
america        23.97
republican     23.93
month9         22.78
government     22.50
actually       21.24
month11        21.16
```

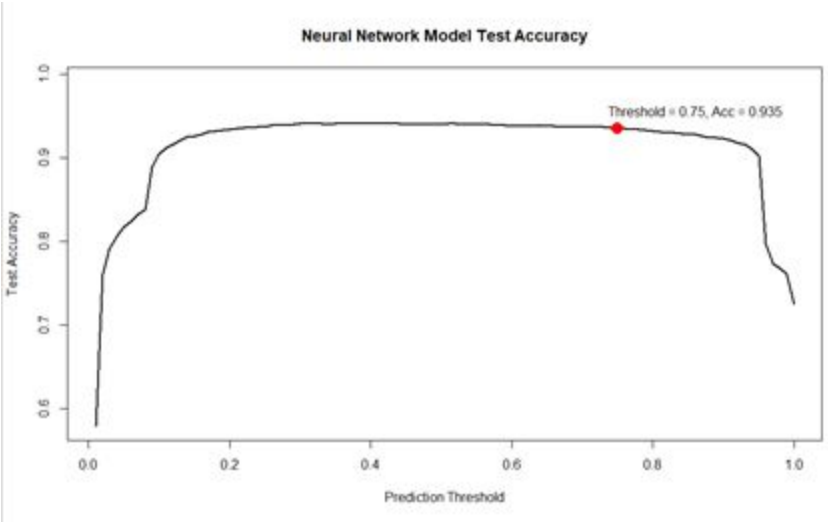




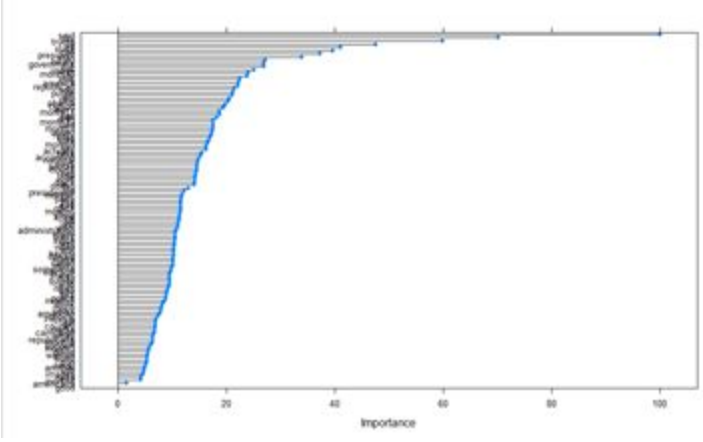
Neural Network

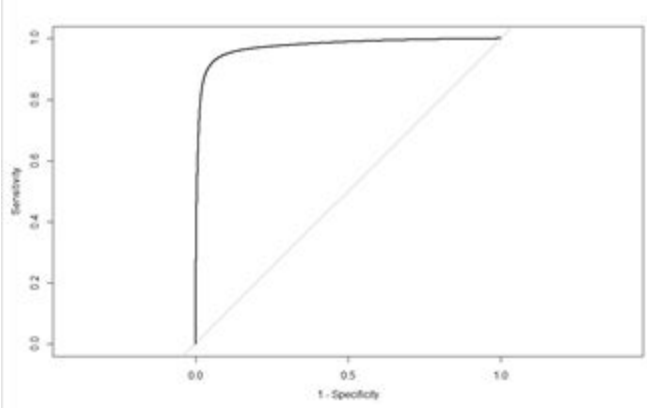
```
Neural Network

30884 samples
  101 predictor
    2 classes: 'fake', 'real'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 23164, 23164, 23164, 23164, 23164, 23164, ...
Resampling results across tuning parameters:

  size  decay  ROC        Sens       Spec
   1    0.00   0.9424773  0.8690178  0.9558849
   1    0.01   0.9502013  0.9040551  0.9364983
   1    0.10   0.9631626  0.9061689  0.9503445
   1    0.50   0.9653216  0.9011488  0.9501652
   2    0.00   0.9662964  0.9103963  0.9490420
   2    0.01   0.9707572  0.9088225  0.9500425
   2    0.10   0.9745847  0.9181735  0.9470599
   2    0.50   0.9752962  0.9152441  0.9482397
   3    0.00   0.9549866  0.8848133  0.9445399
   3    0.01   0.9764560  0.9200574  0.9466163
   3    0.10   0.9768383  0.9212177  0.9471449
   3    0.50   0.9784868  0.9235956  0.9481359
   4    0.00   0.9769859  0.9238599  0.9439169
   4    0.01   0.9774635  0.9159678  0.9500708
   4    0.10   0.9672369  0.8897300  0.9456819
   4    0.50   0.9803471  0.9232165  0.9530061
   5    0.00   0.9783570  0.9202068  0.9476923
   5    0.01   0.9785990  0.9211832  0.9480604
   5    0.10   0.9785706  0.9180816  0.9494856
   5    0.50   0.9817286  0.9267777  0.9524210
   6    0.00   0.9792663  0.9245491  0.9466258
   6    0.01   0.9800744  0.9245032  0.9490042
   6    0.10   0.9808119  0.9242160  0.9510901
   6    0.50   0.9824557  0.9265020  0.9546201
   7    0.00   0.9799264  0.9233084  0.9483436
   7    0.01   0.9801371  0.9245146  0.9480321
   7    0.10   0.9812176  0.9240437  0.9503351
   7    0.50   0.9831456  0.9283285  0.9532704
   8    0.00   0.9800792  0.9257898  0.9461916
   8    0.01   0.9795871  0.9238139  0.9476734
   8    0.10   0.9813406  0.9263871  0.9498820
   8    0.50   0.9838924  0.9288685  0.9549599
   9    0.00         NaN        NaN        NaN
   9    0.01         NaN        NaN        NaN
   9    0.10         NaN        NaN        NaN
   9    0.50         NaN        NaN        NaN
  10    0.00         NaN        NaN        NaN
  10    0.01         NaN        NaN        NaN
  10    0.10         NaN        NaN        NaN
  10    0.50         NaN        NaN        NaN

ROC was used to select the optimal model using the largest value.
The final values used for the model were size = 8 and decay = 0.5.
```






```
nnet variable importance

  only 20 most important variables shown (out of 111)

           Overall
said        100.00
day          70.09
trump        59.94
via          47.50
state        41.01
can          39.61
president    37.25
man          33.91
hillary      27.15
government   26.91
just         26.84
image        25.01
month10      24.04
year         23.67
one          22.42
america      22.16
republican   21.98
now          21.43
public       21.17
will         21.02
```



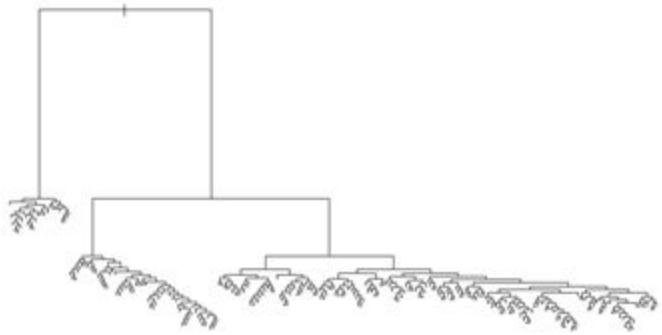


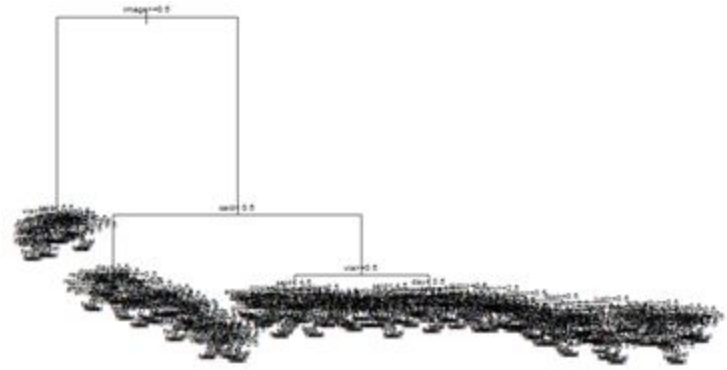
Decision Tree

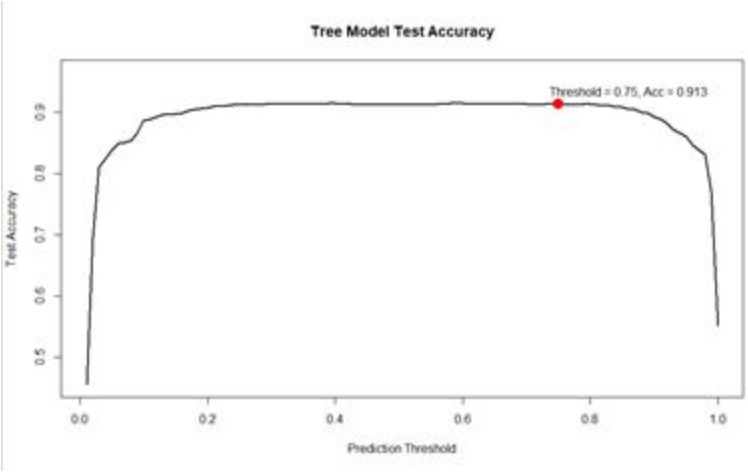
```
CART

30884 samples
  101 predictor
    2 classes: 'fake', 'real'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 23164, 23164, 23164, 23164, 23164, 23164, ...
Resampling results across tuning parameters:

  cp    ROC        Sens        Spec
  0.00  0.9608208  0.89209650  0.9257669
  0.01  0.8891232  0.83582998  0.8935158
  0.02  0.8884594  0.85588742  0.8663143
  0.03  0.8884594  0.85588742  0.8663143
  0.04  0.8523063  0.76493969  0.9112978
  0.05  0.8509347  0.76143596  0.9133365
  0.06  0.8509347  0.76143596  0.9133365
  0.07  0.8509347  0.76143596  0.9133365
  0.08  0.8509347  0.76143596  0.9133365
  0.09  0.8509347  0.76143596  0.9133365
  0.10  0.8509347  0.76143596  0.9133365
  0.11  0.8509347  0.76143596  0.9133365
  0.12  0.8509347  0.76143596  0.9133365
  0.13  0.8509347  0.76143596  0.9133365
  0.14  0.8509347  0.76143596  0.9133365
  0.15  0.8468234  0.75191269  0.9155923
  0.16  0.7544349  0.53406088  0.9748089
  0.17  0.7544349  0.53406088  0.9748089
  0.18  0.7544349  0.53406088  0.9748089
  0.19  0.7544349  0.53406088  0.9748089
  0.20  0.7544349  0.53406088  0.9748089
  0.21  0.7544349  0.53406088  0.9748089
  0.22  0.7544349  0.53406088  0.9748089
  0.23  0.7544349  0.53406088  0.9748089
  0.24  0.7544349  0.53406088  0.9748089
  0.25  0.7544349  0.53406088  0.9748089
  0.26  0.7544349  0.53406088  0.9748089
  0.27  0.7544349  0.53406088  0.9748089
  0.28  0.7544349  0.53406088  0.9748089
  0.29  0.7544349  0.53406088  0.9748089
  0.30  0.7544349  0.53406088  0.9748089
  0.31  0.7544349  0.53406088  0.9748089
  0.32  0.7544349  0.53406088  0.9748089
  0.33  0.7544349  0.53406088  0.9748089
  0.34  0.7544349  0.53406088  0.9748089
  0.35  0.7544349  0.53406088  0.9748089
  0.36  0.7544349  0.53406088  0.9748089
  0.37  0.7544349  0.53406088  0.9748089
  0.38  0.7544349  0.53406088  0.9748089
  0.39  0.7544349  0.53406088  0.9748089
  0.40  0.7544349  0.53406088  0.9748089
  0.41  0.7544349  0.53406088  0.9748089
```

0.42	0.7544349	0.53406088	0.9748089
0.43	0.7544349	0.53406088	0.9748089
0.44	0.7544349	0.53406088	0.9748089
0.45	0.7544349	0.53406088	0.9748089
0.46	0.7544349	0.53406088	0.9748089
0.47	0.7544349	0.53406088	0.9748089
0.48	0.7544349	0.53406088	0.9748089
0.49	0.7544349	0.53406088	0.9748089
0.50	0.7333173	0.48989087	0.9767437
0.51	0.5196199	0.04124067	0.9979991
0.52	0.5000000	0.00000000	1.0000000
0.53	0.5000000	0.00000000	1.0000000
0.54	0.5000000	0.00000000	1.0000000
0.55	0.5000000	0.00000000	1.0000000
0.56	0.5000000	0.00000000	1.0000000
0.57	0.5000000	0.00000000	1.0000000
0.58	0.5000000	0.00000000	1.0000000
0.59	0.5000000	0.00000000	1.0000000
0.60	0.5000000	0.00000000	1.0000000
0.61	0.5000000	0.00000000	1.0000000
0.62	0.5000000	0.00000000	1.0000000
0.63	0.5000000	0.00000000	1.0000000
0.64	0.5000000	0.00000000	1.0000000
0.65	0.5000000	0.00000000	1.0000000
0.66	0.5000000	0.00000000	1.0000000
0.67	0.5000000	0.00000000	1.0000000
0.68	0.5000000	0.00000000	1.0000000
0.69	0.5000000	0.00000000	1.0000000
0.70	0.5000000	0.00000000	1.0000000
0.71	0.5000000	0.00000000	1.0000000
0.72	0.5000000	0.00000000	1.0000000
0.73	0.5000000	0.00000000	1.0000000
0.74	0.5000000	0.00000000	1.0000000
0.75	0.5000000	0.00000000	1.0000000
0.76	0.5000000	0.00000000	1.0000000
0.77	0.5000000	0.00000000	1.0000000
0.78	0.5000000	0.00000000	1.0000000
0.79	0.5000000	0.00000000	1.0000000
0.80	0.5000000	0.00000000	1.0000000
0.81	0.5000000	0.00000000	1.0000000
0.82	0.5000000	0.00000000	1.0000000
0.83	0.5000000	0.00000000	1.0000000
0.84	0.5000000	0.00000000	1.0000000
0.85	0.5000000	0.00000000	1.0000000
0.86	0.5000000	0.00000000	1.0000000
0.87	0.5000000	0.00000000	1.0000000
0.88	0.5000000	0.00000000	1.0000000
0.89	0.5000000	0.00000000	1.0000000
0.90	0.5000000	0.00000000	1.0000000
0.91	0.5000000	0.00000000	1.0000000
0.92	0.5000000	0.00000000	1.0000000
0.93	0.5000000	0.00000000	1.0000000
0.94	0.5000000	0.00000000	1.0000000
0.95	0.5000000	0.00000000	1.0000000
0.96	0.5000000	0.00000000	1.0000000
0.97	0.5000000	0.00000000	1.0000000
0.98	0.5000000	0.00000000	1.0000000
0.99	0.5000000	0.00000000	1.0000000
1.00	0.5000000	0.00000000	1.0000000

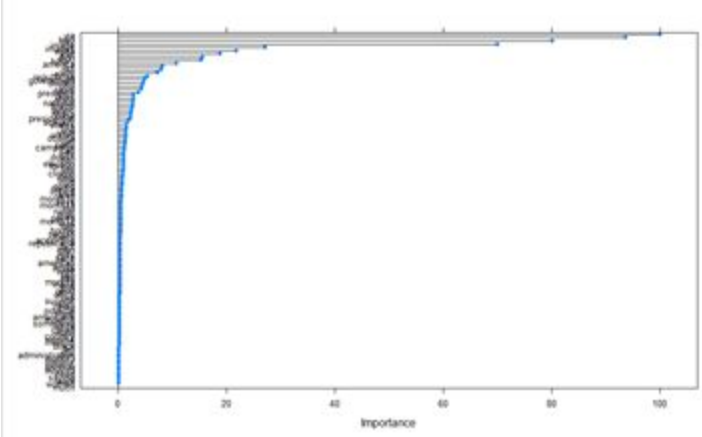
ROC was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.

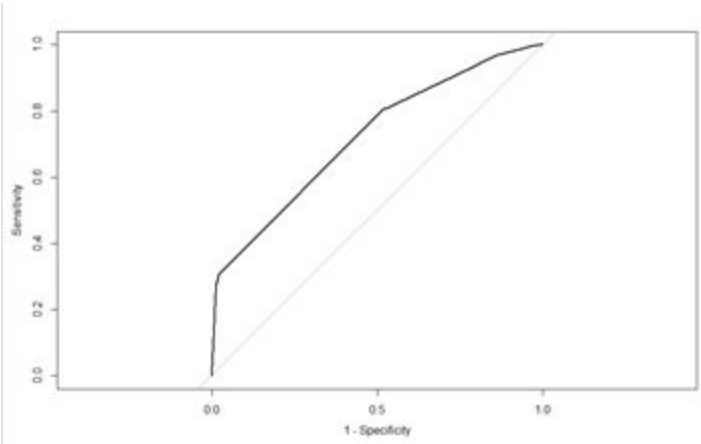





Tree Model Test Accuracy
Threshold = 0.75, Acc = 0.913
Test Accuracy
0.5
0.6
0.7
0.8
0.9
0.0
0.2
0.4
0.6
0.8
1.0
Prediction Threshold

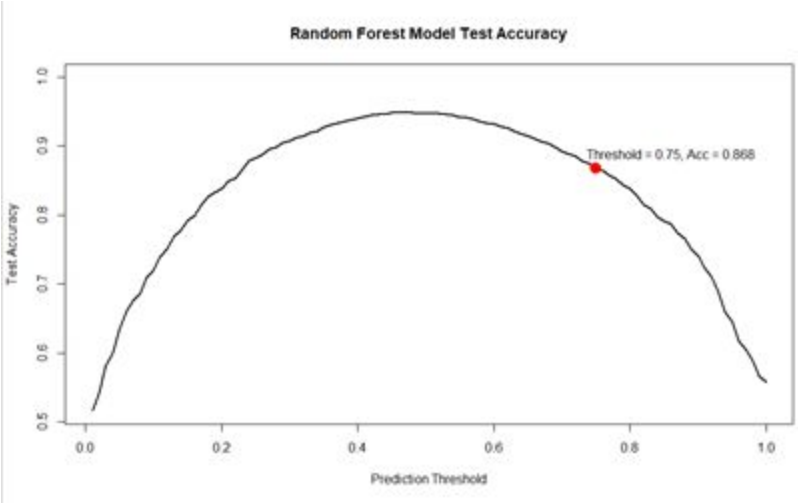
```
rpart variable importance

  only 20 most important variables shown (out of 111)

           Overall
via        100.000
said        93.669
day         80.128
image       70.049
images      27.148
video       21.731
just        18.848
didn        15.553
hillary     15.326
america     10.694
fox          8.084
one          7.932
every        7.211
republican   5.440
government   4.917
donald       4.629
now          4.411
news         4.182
president    3.718
think        2.827
```



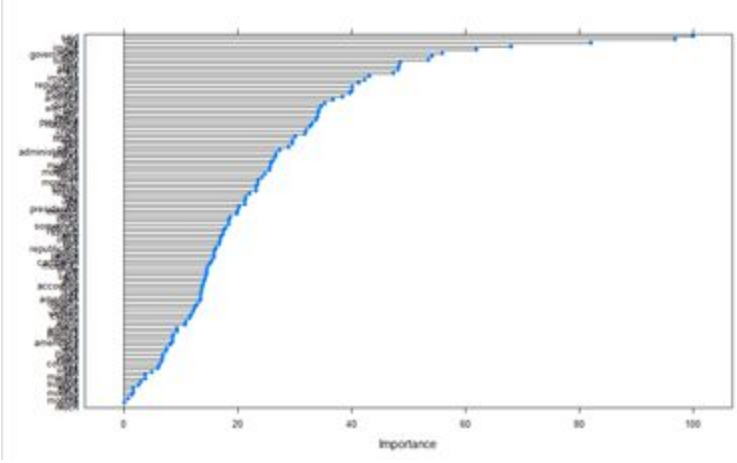


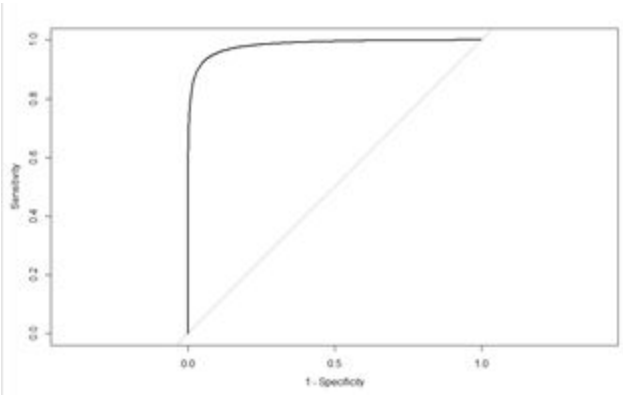
Random Forest

```
Random Forest

30884 samples
  101 predictor
    2 classes: 'fake', 'real'

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (25 reps, 75%)
Summary of sample sizes: 23164, 23164, 23164, 23164, 23164, 23164, ...
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
   1    0.9697761  0.7473176  0.9832279
   2    0.9849671  0.8970821  0.9724965
   3    0.9859347  0.9077886  0.9707692
   4    0.9861150  0.9124296  0.9689476
   5    0.9859807  0.9146123  0.9680227
   6    0.9857436  0.9156462  0.9658235
   7    0.9856096  0.9176106  0.9648608
   8    0.9854315  0.9182769  0.9631996
   9    0.9852710  0.9191729  0.9625484
  10    0.9849695  0.9202412  0.9617744

ROC was used to select the optimal model using the largest value.
The final value used for the model was mtry = 4.
```

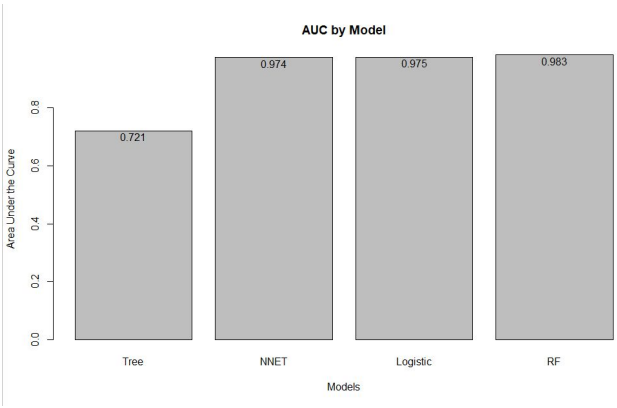


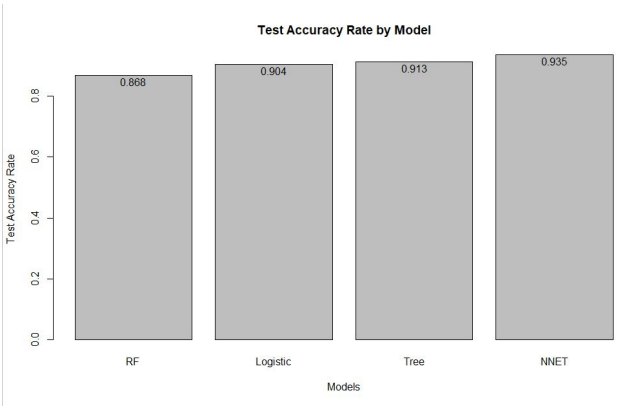

```
rf variable importance

  only 20 most important variables shown (out of 111)

           Importance
said           100.00
day             96.91
via             82.10
image           68.02
video           61.91
government      56.03
didn            54.12
hillary         53.47
just            48.62
state           48.28
every           48.14
fox             47.39
images          43.12
told            42.29
republican      41.25
donald          40.16
actually        40.06
united          39.73
america         38.37
don             36.69
```





Model Comparison: Plots





R Code:

```
### Read in files
real <- read.csv("True.csv", encoding = "UTF-8", as.is = TRUE)
fake <- read.csv("Fake.csv", encoding = "UTF-8", as.is = TRUE)

### Convert date from factor to date
library(lubridate)
table(real$date)
real$date <- mdy(real$date)
class(real$date)

### There is one observation in Fake where the date is not a date.
### It looks to be the title of the article
options(max.print = 2000)
table(fake$date)

fake[which(fake$date == "MSNBC HOST Rudely Assumes Steel Worker Would Never Let His Son Follow in His Footsteps…He Couldn’t Be More Wrong [Video]"),]

### It appears this observations just doesn't have a date.
### Remove this observation
fake <- fake[-18934,]

### There are some dates where they are not of the same format.
### Drop these.
any(grepl("-",fake$date))
```

```
which(grepl("-",fake$date))

fake <- fake[-c(which(grepl("-",fake$date))),]

### Convert to date
fake$date <- mdy(fake$date)
class(fake$date)

### Add month and year varaibles
real$month <- factor(substring(real$date,6,7))
fake$month <- factor(substring(fake$date,6,7))
real$year <- factor(substring(real$date,1,4))
fake$year <- factor(substring(fake$date,1,4))


### Add the class of news article, real or fake
real$Type <- factor("real")
fake$Type <- factor("fake")


### There are some articles where the text is blank.
### Remove these
library(glue)
any(trim(real$title) == "")
any(trim(real$text) == "")
any(trim(fake$title) == "")
any(trim(fake$text) == "")

real <- real[trim(real$text) != "",]
fake <- fake[trim(fake$text) != "",]
real <- real[trim(real$text) != " ",]
fake <- fake[trim(fake$text) != " ",]
real <- real[trim(real$text) != "  ",]
fake <- fake[trim(fake$text) != "  ",]
real <- real[trim(real$text) != "   ",]
fake <- fake[trim(fake$text) != "   ",]



### Combine data
data <- rbind(real, fake)
levels(data$Type)

### Remove duplicates
data <- data[!duplicated(data$text),]

### Sort by time
data <- data[order(data$date),]
class(data$date)

dev.off()

rm(fake)
rm(real)
data <- data[,c(2,7,5)]
data$text <- tolower(data$text)
############# Text Mining #############

### Find 100 most popular words in all data
library(tm)
library(SnowballC)

memory.limit(size = 150000)

fakeText <- data[data$Type == "fake", 1]
```

```
### Text mine the first text
doc <- Corpus(VectorSource(fakeText[1]))
doc <- tm_map(doc, removeNumbers)
doc <- tm_map(doc, removePunctuation)
doc <- tm_map(doc, stripWhitespace)
doc <- tm_map(doc, removeWords, stopwords("english"))
dtm <- TermDocumentMatrix(doc)
m <- as.matrix(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
topwords <- data.frame(word = names(v),freq=v)
head(topwords)

### Create a dataframe to hold words and frequencies
popularWords <- data.frame(word = topwords$word, freq = topwords$freq)

### Now loop over all others and add them to popularWords
for(i in 2:length(fakeText)){
  doc <- Corpus(VectorSource(fakeText[i]))
  doc <- tm_map(doc, removeNumbers)
  doc <- tm_map(doc, removePunctuation)
  doc <- tm_map(doc, stripWhitespace)
  doc <- tm_map(doc, removeWords, stopwords("english"))
  dtm <- TermDocumentMatrix(doc)
  m <- as.matrix(dtm)
  v <- sort(rowSums(m),decreasing=TRUE)
  topwords <- data.frame(word = names(v),freq=v)

  ### Add to popularWords
  popularWords <- rbind(popularWords, topwords)

  rm(doc)
  rm(dtm)
  rm(m)
  rm(v)
  rm(topwords)

  print(paste(i, Sys.time(), nrow(popularWords), sep = "    "))
}


popularWords2 <- aggregate(popularWords$freq,
                           by = list(words = popularWords$word),
                           FUN = sum)
popularWords2Sorted <- popularWords2[order(popularWords2$x, decreasing = TRUE),]
head(popularWords2Sorted, 100)

Top100Words <- as.character(popularWords2Sorted$words[1:100])


Top100WordCounts <- data.frame(matrix(0, nrow = nrow(data), ncol = 100))
colnames(Top100WordCounts) <- Top100Words

library(stringr)

for (i in 1:nrow(data)){

  for (j in 1:100){

    Top100WordCounts[i,j] <- length(str_extract_all(data$text[i], Top100Words[j])[[1]])

    print(paste(i, j, Sys.time(), sep = "    "))
  }
}

############ Finalize data ##########
```

```
dataNew <- cbind(data, Top100WordCounts)
dataNew <- dataNew[,-1]

write.csv(dataNew, "NewsInfo.csv", row.names = FALSE)

rm(list=ls())
library(tm)
library(SnowballC)
library(wordcloud2)

data <- read.csv("NewsInfo.csv")

real <- data[data$Type == "real",2]
rm(data)

################################################

memory.limit(size = 100000000000)
docs <- Corpus(VectorSource(real))
rm(real)
docs <- tm_map(docs, content_transformer(tolower))
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removeWords, stopwords("english"))
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, stemDocument)
dtm <- TermDocumentMatrix(docs)
rm(docs)
m <- as.matrix(dtm)
rm(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
rm(m)
d <- data.frame(word = names(v),freq=v)
rm(v)


wordcloud2(data = d,
        main = "Most Common Words in Real News Data")
### Save this graph and put it on the google drive


############################################
rm(list = ls())

fake <- read.csv("Fake.csv", encoding = "UTF-8", as.is = TRUE)
fake$text <- tolower(fake$text)

fake <- data[data$Type == "fake",2]
rm(data)



docs <- Corpus(VectorSource(fake$text))
rm(fake)
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, removeWords, stopwords("english"))

dtm <- TermDocumentMatrix(docs)
rm(docs)
m <- as.matrix(dtm)
rm(dtm)
v <- sort(rowSums(m),decreasing=TRUE)
rm(m)
d <- data.frame(word = names(v),freq=v)
```

```
rm(v)


wordcloud2(data = d,
          main = "Most Common Words in Fake News Data")
rm(list=ls())
data <- read.csv("NewsInfo.csv")
data$month <- factor(data$month)

library(ggplot2)
library(caret)
library(pROC)
library(corrplot)


############ Data exploration ##############

barplot(table(data$Type), main = "News Type Distribution",
        xlab = "News Type",
        ylab = "Frequency")
table(data$Type)


plot(as.Date(names(table(fake$date))), unname(table(fake$date)),
     type = "l", col = "red", ylim = c(0,max(unname(table(real$date)))+5),
     xlab = "Date", ylab = "Frequency", main = "Real vs. Fake News")
lines(as.Date(names(table(real$date))), unname(table(real$date)),
      col = "green")
rect(xleft = as.Date("2016-1-01"), ybottom = 0,
     xright = as.Date("2016-11-06"), ytop = 200,
     col = "blue", density = 3)
text(x = as.Date("2016-1-01"), y = 100,
     labels = "Election Year", cex = 2,
     adj = -0.2)
legend("topleft", legend = c("Real","Fake"),
       lty = 1,
       col = c("green","red"))



corrplot(cor(data[,-c(1,2)]))


corrplot(cor(data[data$Type=="real",-c(1,2)]), main = "real news")
corrplot(cor(data[data$Type=="fake",-c(1,2)]), main = "fake news")

levels(data$month)
for (i in 1:length(levels(data$month))){
  corrplot(cor(data[data$month == levels(data$month)[i],-c(1,2)]),
           main = paste ("month",i, sep = " "))
}


colSums(data[,-c(1,2)])
colSums(data[data$Type=="real",-c(1,2)])
colSums(data[data$Type=="fake",-c(1,2)])

sumsType <- data.frame(SUM = as.vector(colSums(data[data$Type=="real",-c(1,2)])),
                       vars = names(colSums(data[data$Type=="real",-c(1,2)])),
                       Type = "real")

sumsType <- rbind(sumsType, data.frame(SUM = as.vector(colSums(data[data$Type=="real",-c(1,2)])),
                                       vars = names(colSums(data[data$Type=="real",-c(1,2)])),
                                       Type = "fake"))
```

```
ggplot(data = sumsType, aes(x = reorder(vars, SUM), y = SUM, fill = Type)) +
  geom_bar(stat = "identity") + theme(axis.text.x = element_text(vjust = 0.25,
    angle = 90))

ggplot(data = sumsType, aes(x = reorder(vars, SUM), y = SUM, fill = Type)) +
  geom_bar(stat = "identity", position = position_dodge(), width = 0.8) +
  theme(axis.text.x = element_text(vjust = 0.25, angle = 90))+
  labs(title = "Sum of Words by Type", x = "Words")

########## Modeling ###############

set.seed(210)
intrain <- createDataPartition(data$Type, p = 0.8)[[1]]
Train <- data[intrain,]
Test <- data[-intrain,]


AccPlot <- function(MODEL, THRESHOLD, TESTdata, modelName){

  thresholds <- seq(0.01, 1, 0.01)
  Acc <- c()

  for (i in 1:length(thresholds)){
    Preds <- predict(MODEL, TESTdata, type = "prob")
    Preds <- ifelse(Preds$fake >= thresholds[i], "fake", "real")
    Acc <- append(Acc, 1 - mean(Preds != Test$Type))

  }

  plot(thresholds, Acc, type = "l", lwd = 2,
       xlab = "Prediction Threshold",
       ylab = "Test Accuracy",
       main = paste(modelName, "Model Test Accuracy", sep = " "),
       ylim = c(min(Acc),max(Acc) + 0.05))
  points(THRESHOLD, Acc[THRESHOLD * 100], col = "red", pch = 16, cex = 2)
  text(THRESHOLD + 0.11, Acc[THRESHOLD * 100] + 0.02,
       labels = paste("Threshold = ", THRESHOLD, ", Acc = ",
                      round(Acc[THRESHOLD * 100], 3), sep = ""))

}




############ Logistic ##################

set.seed(210)
ctrl <- trainControl(method = "LGOCV",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

logicTune <- train(Type ~., data = Train,
                   method = "glm",
                   metric = "ROC",
                   trControl = ctrl)
logicTune

summary(logicTune)


### Make graph to compare
logicCoeff <- as.data.frame(summary(logicTune)$coefficients)
logicCoeff <- logicCoeff[-1,]
logicCoeff$Vars <- rownames(logicCoeff)
logicCoeff$Sig <- ifelse(logicCoeff$`Pr(>|z|)` <= 0.05, "*","")
```

```
VJUST <- ifelse(logicCoeff$Estimate <= 0, 1.1, 0.4)
ggplot(data = logicCoeff, aes(x = reorder(Vars, Estimate), y = Estimate,
                              fill = (Estimate >= 0))) +
  geom_bar(stat = "identity", width = 0.4) +
  geom_text(aes(label = Sig), col = "white", size = 10, vjust = VJUST) +
  labs(title = "Logistic Regression Estimates",
    x = NULL, y = "Log(Odds)") +
  theme(legend.position = "none") + theme(axis.text = element_text(hjust = 1),
    axis.text.x = element_text(vjust = 0.25,
        angle = 90)) + theme(panel.grid.major = element_line(colour = "gray26"),
    panel.grid.minor = element_line(linetype = "blank"),
    panel.background = element_rect(fill = "black")) +
  theme(plot.subtitle = element_text(size = 9)) +
  theme(axis.text.y = element_text(size = 12))



### Accuracy Plot
AccPlot(logicTune, 0.75, Test, "Logistic")

### Confusion Matrix
logicPred <- predict(logicTune, Test, type = "prob")
logicPred <- ifelse(logicPred$fake >= 0.75, "fake", "real")
table(logicPred, Test$Type)

### Test error rate
mean(logicPred != Test$Type)

### Importance
varImp(logicTune)
plot(varImp(logicTune))

### Test ROC
logicROC <- roc(response = logicTune$pred$obs,
                predictor = logicTune$pred$fake,
                levels = rev(levels(logicTune$pred$obs)))
plot(logicROC, legacy.axes = TRUE)
logicAUC <- auc(logicROC)


### Save results

Test_Acc <- c(1 - mean(logicPred != Test$Type))
Test_Error <- c(mean(logicPred != Test$Type))
Test_AUC <- c(logicAUC[1])
Models <- c("Logistic")




############# Neural Network #################

nnetGrid <- expand.grid(.size = 1:10,
                        .decay = c(0, .01, .1, 0.5))
set.seed(210)
nnetTune <- train(Type ~., data = Train,
                  method = "nnet",
                  metric = "ROC",
                  tuneGrid = nnetGrid,
                  trControl = ctrl)

nnetTune

library(devtools)
source_url("https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet_plot_update.r")
```

```
plot.nnet(nnetTune)

AccPlot(nnetTune, 0.75, Test, "Neural Network")

nnetPred <- predict(nnetTune, Test, type = "prob")
nnetPred <- ifelse(nnetPred$fake >= 0.75, "fake", "real")

table(nnetPred, Test$Type)

### Test error rate
mean(nnetPred != Test$Type)

### Importance
varImp(nnetTune)
plot(varImp(nnetTune))

### Test ROC
nnetROC <- roc(response = nnetTune$pred$obs,
               predictor = nnetTune$pred$fake,
               levels = rev(levels(nnetTune$pred$obs)))
plot(nnetROC, legacy.axes = TRUE)
nnetAUC <- auc(nnetROC)


### Save results

Test_Acc <- append(Test_Acc, 1 - mean(nnetPred != Test$Type))
Test_Error <- append(Test_Error, mean(nnetPred != Test$Type))
Test_AUC <- append(Test_AUC, nnetAUC[1])
Models <- append(Models, "NNET")


############## Decision Trees ############

library(rpart)
cpGrid <- expand.grid(cp = seq(0,0.1, 0.005))
set.seed(210)
treeTune <- train(Type ~., data = Train,
                  method = "rpart",
                  tuneGrid = cpGrid,
                  metric = "ROC",
                  trControl = ctrl)

treeTune

plot(treeTune$finalModel)
text(treeTune$finalModel, cex = 0.6)

AccPlot(treeTune, 0.75, Test, "Tree")

treePred <- predict(treeTune, Test, type = "prob")
treePred <- ifelse(treePred$fake >= 0.75, "fake", "real")

table(treePred, Test$Type)

### Test error rate
mean(treePred != Test$Type)

### Importance
varImp(treeTune)
plot(varImp(treeTune))

### Test ROC
treeROC <- roc(response = treeTune$pred$obs,
               predictor = treeTune$pred$fake,
               levels = rev(levels(treeTune$pred$obs)))
```

```
plot(treeROC, legacy.axes = TRUE)
treeAUC <- auc(treeROC)


### Save results

Test_Acc <- append(Test_Acc, 1 - mean(treePred != Test$Type))
Test_Error <- append(Test_Error, mean(treePred != Test$Type))
Test_AUC <- append(Test_AUC, treeAUC[1])
Models <- append(Models, "Tree")




############ Random Forest #############

mtryGrid <- data.frame(mtry = floor(seq(1, 10)))

set.seed(210)
rfTune <- train(Type ~., data = Train,
                method = "rf",
                tuneGrid = mtryGrid,
                ntree = 150,
                nodesize = 50,
                importance = TRUE,
                metric = "ROC",
                trControl = ctrl)
rfTune


AccPlot(rfTune, 0.75, Test, "Random Forest")

rfPred <- predict(rfTune, Test, type = "prob")
rfPred <- ifelse(rfPred$fake >= 0.75, "fake", "real")

table(rfPred, Test$Type)

### Test error rate
mean(rfPred != Test$Type)

### Importance
varImp(rfTune)
plot(varImp(rfTune))

### Test ROC
rfROC <- roc(response = rfTune$pred$obs,
             predictor = rfTune$pred$fake,
             levels = rev(levels(rfTune$pred$obs)))
plot(rfROC, legacy.axes = TRUE)
rfAUC <- auc(rfROC)


### Save results

Test_Acc <- append(Test_Acc, 1 - mean(rfPred != Test$Type))
Test_Error <- append(Test_Error, mean(rfPred != Test$Type))
Test_AUC <- append(Test_AUC, rfAUC[1])
Models <- append(Models, "RF")



########## Plots #############
testResults <- data.frame(Test_Acc,Test_Error,Test_AUC,Models)
```

```
### ROC Curves
par(pty = "s")
plot(logicROC, lty = 1, lwd = 3.5,
    main = "ROC Curves by Model")
lines(rfROC, col = "red", lty = 3, lwd = 3.5)
lines(nnetROC, col = "blue", lty = 3, lwd = 3.5)
lines(treeROC, col = "green", lty = 3, lwd = 3.5)

legend("bottomright",
      legend = as.character(c("Logistic","RF","NNET","TREE")),
      lty = c(1,3,3,3),
      lwd = c(3.5,3.5,3.5,3.5),
      col = c("black","red","blue","green"))


dev.off()

### Test error rate, sorted
OrderErr <- testResults[order(testResults$Test_Error),]

bpErr <- barplot(OrderErr$Test_Error, names.arg =  OrderErr$Models,
              xlab = "Models", ylab = "Test Error Rate",
              main = "Test Error Rates by Model")
text(bpErr, OrderErr$Test_Error - 0.005,
    labels = round(OrderErr$Test_Error, 3))



### AUC, sorted
OrderAUC <- testResults[order(testResults$Test_AUC),]

bpAUC <- barplot(OrderAUC$Test_AUC, names.arg =  OrderAUC$Models,
              xlab = "Models", ylab = "Area Under the Curve",
              main = "AUC by Model")
text(bpAUC, OrderAUC$Test_AUC - 0.02,
    labels = round(OrderAUC$Test_AUC, 3))


### Accuracy, sorted
OrderAcc <- testResults[order(testResults$Test_Acc),]

bpAcc <- barplot(OrderAcc$Test_Acc, names.arg =  OrderAcc$Models,
              xlab = "Models", ylab = "Test Accuracy Rate",
              main = "Test Accuracy Rate by Model")
text(bpAcc, OrderAcc$Test_Acc - 0.02,
    labels = round(OrderAcc$Test_Acc, 3))



### Resamples
resamp <- resamples(list(Logistic = logicTune,
                  NNET = nnetTune,
                  Tree = treeTune,
                  RF = rfTune))
dotplot(resamp, metric = "ROC")


modelDifferences <- diff(resamp)
modelDifferences$statistics$ROC
```

```
library(highcharter)

highchart() %>%
  hc_yAxis_multiples(
    list(lineWidth = 3, lineColor = "red", title=list(text="Error")),
    list(lineWidth = 3, lineColor = "green", title=list(text="Accuracy")),
    list(lineWidth = 3, lineColor = "blue", title=list(text="AUC"))
  ) %>%
  hc_add_series(data = testResults$Test_Error, color = "red", type = "column") %>%
  hc_add_series(data = testResults$Test_Acc, color = "green", type = "column",
                yAxis = 1) %>%
  hc_add_series(data = testResults$Test_AUC, color = "blue", type = "column",
                yAxis = 2) %>%
  hc_xAxis(categories = testResults$Models, title = list(text = "Models"))
```