FAKE NEWS DETECTION PROJECT

Abstract

A report outlining the different Machine Learning models used in classifying fake news, calculating their accuracies, and summarizing the best results.

Syed Hamdan Mustafa
1006193209

Michael Attong
999388283

Muhammad Farhan Riaz
999520193

Matthew Vassov
999561530

Table of Contents

Purpose & Competition Submission 2

Data Approach 2

Exploratory Analysis 3

Model Implementation & Tuning 5

Summary of Results & Recommendation 6

Appendix 8

Purpose & Competition Submission

i. Introduction

Today, as the abundance of information becomes more readily available, it is difficult to determine whether said information is true or false. This information, especially when presented over a medium such as the internet, can be drastically manipulated. As a result, people are unsure of whether the retrieved story is real or not.

The primary purpose of this report is to outline the steps taken into identifying whether a claim has merit – that is to say, whether it is True, Partially True, or False. In order to accomplish this, the data – segregated by claims and related articles, must undergo a process of sentiment analysis.

ii. Competition Submission

The docker image with our Multinomial Naïve Bayes model was uploaded to the competition website and yielded a result of 0.312117, placing us in 44th at the time of preparing this report.

42	Threed	0.317145
43	UcanUup_1624	0.312152
44	Kernel Sanders	0.312117
45	G12	0.307378
46	MIE1624 Group 5	0.302275

Data Approach

The Python program was designed to execute the steps as listed below:

i. Data Preparation

The claims and articles were read from the specified file paths on the competition website. Their respective dataframes were constructed and organized. The text then went through an NLP clean-up process, including:

- Eliminating all stop-words.
- Lowercasing all tokens.
- Removing all punctuations and special characters.
- Removing all hashtags and URL's embedded in the text.
- Removing emojis.

ii. TF-IDF Vectorizing

The next step involved applying a TF-IDF vectorizer to the data. In theory, this works better than other methods such as Bag-Of-Words (ie. Word frequency), especially for cases where the text length is large (large documents, articles of varying length etc.). TF-IDF are word frequency scores that try to highlight words that are more interesting, e.g. frequent in a document but not across documents.

iii. Test-Train Splitting

Once the data was vectorized, it was split into training and testing data. As usual, 70% of the data was allocated to training while 30% was allocated to testing.

iv. Model Testing

Different machine learning algorithms were applied to the training data, in order to retrieve the training accuracy, test accuracy and F1 scores. For the analysis, the F1 score was the most important measure, since accuracy could be misleading in certain situations. Because the data is presenting text, it is more important to measure the precision and recall (F1 takes both into consideration). Since there were more than 2 targets (True, Partially True and False), the macro version of F1 score was utilized which considers the different multi-class targets.

The machine learning models used for the analysis are as follows:

- Decision Tree
- Random Forest
- Naïve Bayes
- Logistic Regression
- Neural Nets (CNNs and Dense)*

*Note: Ultimately it was decided not to use Neural Nets as the final shortlisted set of models, however this analysis was also performed and directly compared to the best model.

v. Hyperparameter Tuning

Once the models were applied to the data and the results were calculated, the model that performed best would undergo a process of hyperparameter tuning. Here, the best parameters for their respective attributes would be selected. This would allow for a recommendation to be made.

Exploratory Analysis

An exploratory analysis was performed in order to better understand the structure of the claims and related articles.

i. Word Cloud Analysis

Using the Wordcloud module, the topmost used words for each label were extracted. For the figures below, from left to right, the first image contained all labels (i.e. True, Partially True and False), the second contained only false labels, the third contained only partially true labels, and the fourth contained only true labels.

While this data (i.e. data for most frequent words) alone is not enough in performing sentiment analysis, it is interesting to note that certain key tokens such “united state” and “president” are consistent across all labels. This is useful to know, since it gives an idea of what topics are being referred to and where they may (or may not!) be occurring.

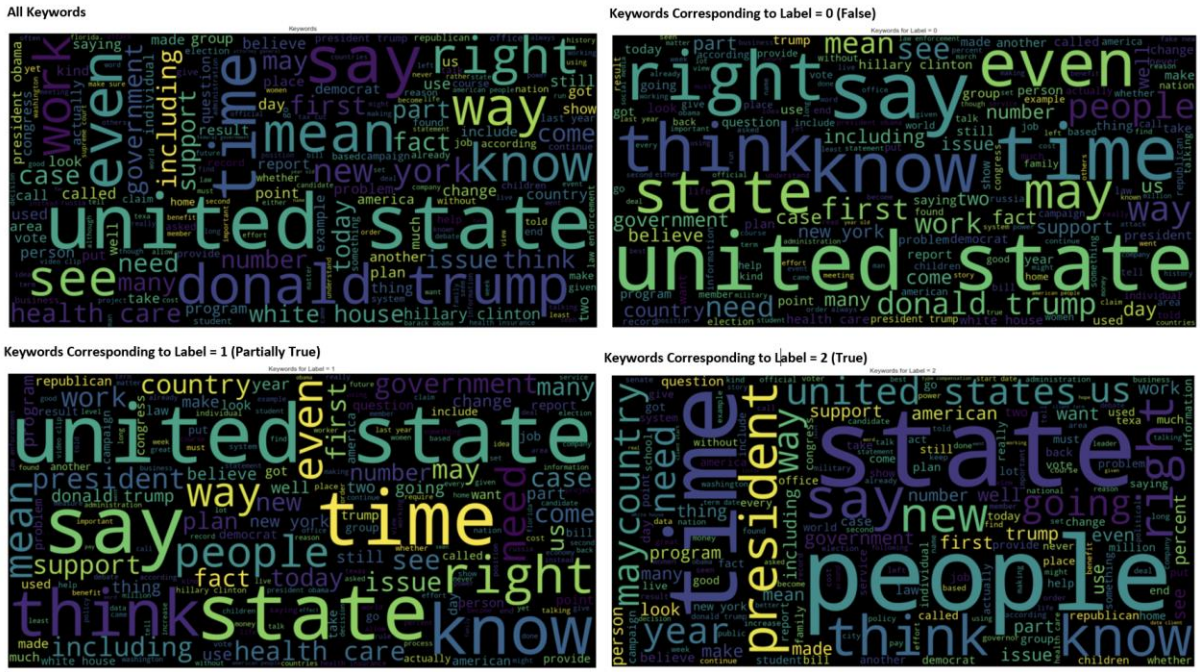

Figure 1: Word Cloud Analysis

ii. Analysis of Labels by Claimant

For the second part of the exploration, the top 10 most active claimants were chosen. Each claimant had their labels measured based on the provided data. It was found that Donald Trump had the highest count of claims, most of which proved to be false (we were equally as surprised!).

It is interesting to note that even though he had the greatest number of claims, his total number of true labels was less than that of Barack Obama and Hillary Clinton. It is also interesting to note that he posts more regularly than "Bloggers", which can be assumed to be a group of people as opposed to one particular person.

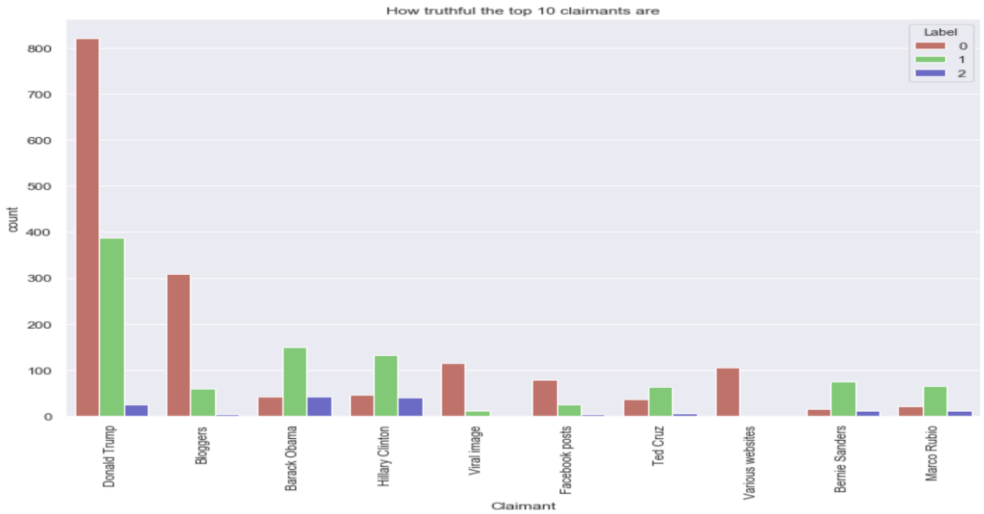

Figure 2: Top 10 Claimant Label Count

Model Implementation & Tuning

A multitude of tests were performed under different feature sets and combinations of claims and articles. Each test is broken out below, between Table 1.1 – Table 1.3. Note that Convolutional and Dense Neural Networks were excluded from these tests, as they would be directly compared to the best model as discussed below.

i. Table 1.1: Claims & Articles Combined

These tests looked at the combination of claims and articles. Their accuracies and F1 scores were calculated based on the number of features. Since it represented the first set of tests in the series, 6 evaluations were performed.

Attempt	Best Model	Features[1]	Accuracy	F1 Score
1	Logistic Regression	1,000	59%	0.412
2	Logistic Regression	1,500	59%	0.426
3	Logistic Regression	1,800	59%	0.428
4	Random Forest	400	59%	0.416
5	Random Forest	200	58%	0.412
6	Logistic Regression	2,500	60%	0.442

ii. Table 1.2: Claims Only

These tests are only looking at claims as the number of features change. Only 4 tests were performed, because it was found that for the most part, an increase in the feature set correlated with better accuracies and F1 scores.

Attempt	Best Model	Features	Accuracy	F1 Score
1	Logistic Regression	1,000	56%	0.419
2	Logistic Regression	1,500	54%	0.415
3	Logistic Regression	400	58%	0.412
4	Naïve Bayes	2,500	59%	0.417

iii. Table 1.3: Articles Only

These tests are only looking at related articles. Only 3 evaluations of different feature sets were performed.

Attempt	Best Model	Features	Accuracy	F1 Score
1	Random Forest	1,000	59%	0.419
2	Random Forest	400	59%	0.413
3	Logistic Regression	2,500	59%	0.444

iv. Best Test Results

After analysing the data for all three test sets, it was found that the Logistic Regression model yielded the best results at an F1 score of 0.444 and an accuracy of 59%. The number of features was set to 2,500 while the model was trained on the related articles data only.

v. Logistic Regression Hyperparameter Tuning

Once the best model from above was selected, it underwent a hyperparameter tuning process. The 2 attributes that were selected for change were the inverse strength regulating coefficient (C) and the multiclass (multi_class) labeller. The below data shows the parameters to be tested for each attribute:

[1] These features refer to the maximum features obtained from the TF-IDF vectorizer.

$$C = [0.001, 0.01, 0.1, 1, 10]$$
$$\text{multi\_class} = [\text{ovr}, \text{multnomial}, \text{auto}]$$
$$\text{solver} = [\text{lbfgs}]$$

Since there were five C values and three multiclass labels to test, the program underwent 15 total iterations. The accuracy and F1 scores were calculated for each iteration and the best was selected.

Summary of Results & Recommendation

i. Logistic Regression Results

Once the Logistic Regression model went through the hyper-parameter tuning phase, it was found that using an inverse strength regulating coefficient at C = 10 and a multiclass labeller at mc = "multinomial" yielded the highest F1 score (at 0.466) with an accuracy of 60.4%.

The confusion matrix for these parameters was plotted:

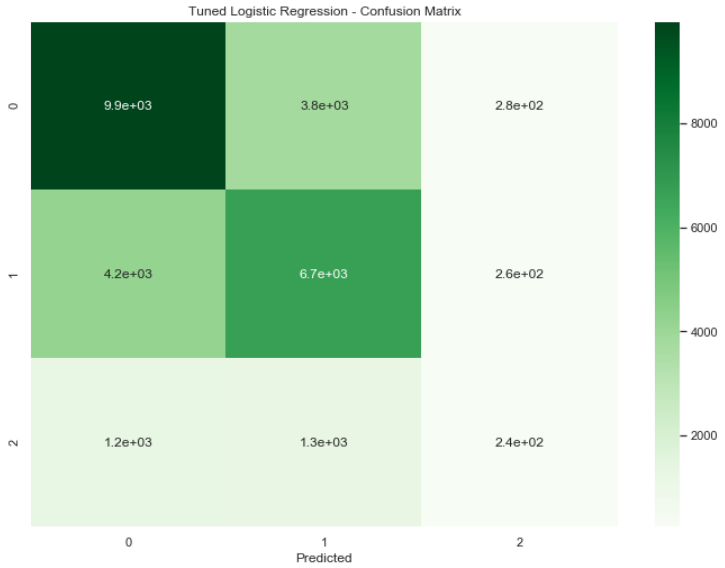

Figure 3: Logistic Regression Model

As displayed in the confusion matrix, the first 2 classes (*False* and *Partially True*) yielded higher correct predictions than the *True* class. This ultimately makes sense, because the number of training examples for false claims and partially true claims are much greater than true claims (see **Figure 4: Count of Labels** in the **Appendix** section below). Note: 0 = False, 1 = Partly True and 2 = True.

In theory, if the distribution was more balanced, the *True* class would have a higher count of correct predictions.

ii. Testing with Neural Networks

The Logistic Regression model was tested against 2 different *Neural Network* models: *Dense* and *Convolutional*. In theory, Neural Networks provide a strong, competitive output when compared to the other models. A Convolutional Neural Network (CNN) is basically a neural-based approach which represents a feature function that is applied to constituting words or n-grams to extract higher-level features. The resulting abstract features have been effectively used for sentiment analysis, machine translation, and question answering, among other tasks.

Between the 2 Neural Networks, it was found that CNN yielded the better results, which can be similarly compared to those of the Logistic Regression (LR) model. CNN resulted in an F1 score of 0.45, which is better than the base LR model of 0.44. Although, once the LR model was tuned, its F1 score surpassed 0.45.

iii. Recommendation

While CNN performed better than the base LR model, after tuning, LR was the optimal choice. It is important to note this for a few reasons:

- While CNN has a good reputation in obtaining a desirable accuracy and F1 score, it is also very computationally expensive. It took an entire night to run the model, which, from a time-sensitive perspective, is not ideal.
- From our results (plots of training vs validation accuracy), we determined that the model was overfitting. To resolve this issue, a custom architecture would be required which, due to the time sensitive nature of the project and the additional expertise required in the topic, was not pursued (however would be a good next step).
- Hyper-parameter tuning on the LR model was relatively quick. Random search hyperparameter optimization was performed for the Neural Networks since grid search takes too long to run.
- Neural Networks are powerful and could be utilized further to deal with the complexities of the model. It would be worth exploring further deep learning to obtain better results.
- In the end, the simpler model had the best results, which is more beneficial in every domain. Hence, Logistic Regression using TF-IDF is the best fake news model in this case.

Appendix

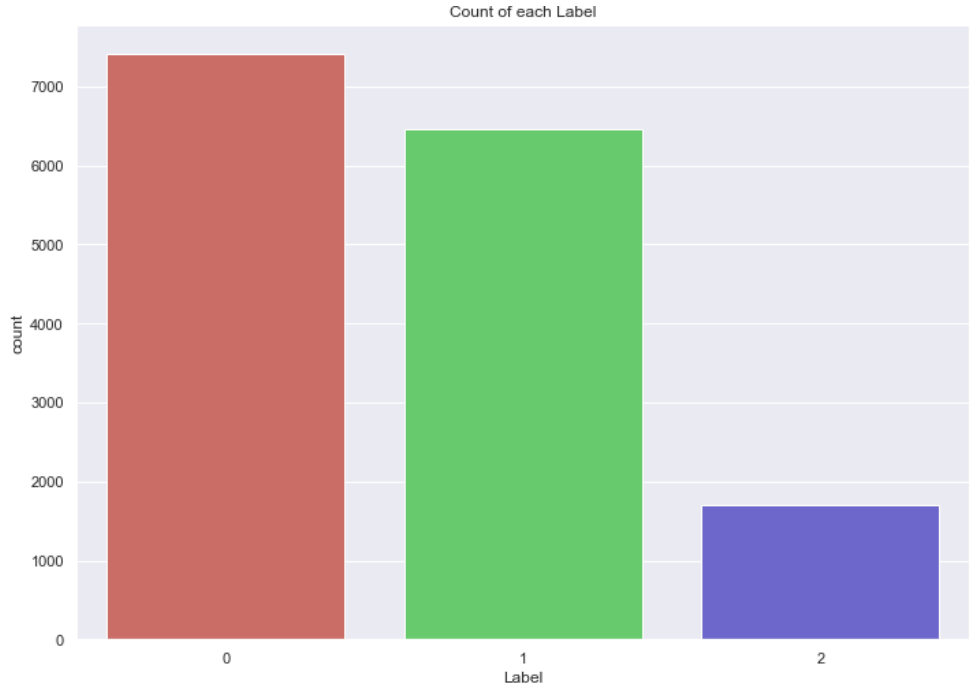

Figure 4: Count of Labels

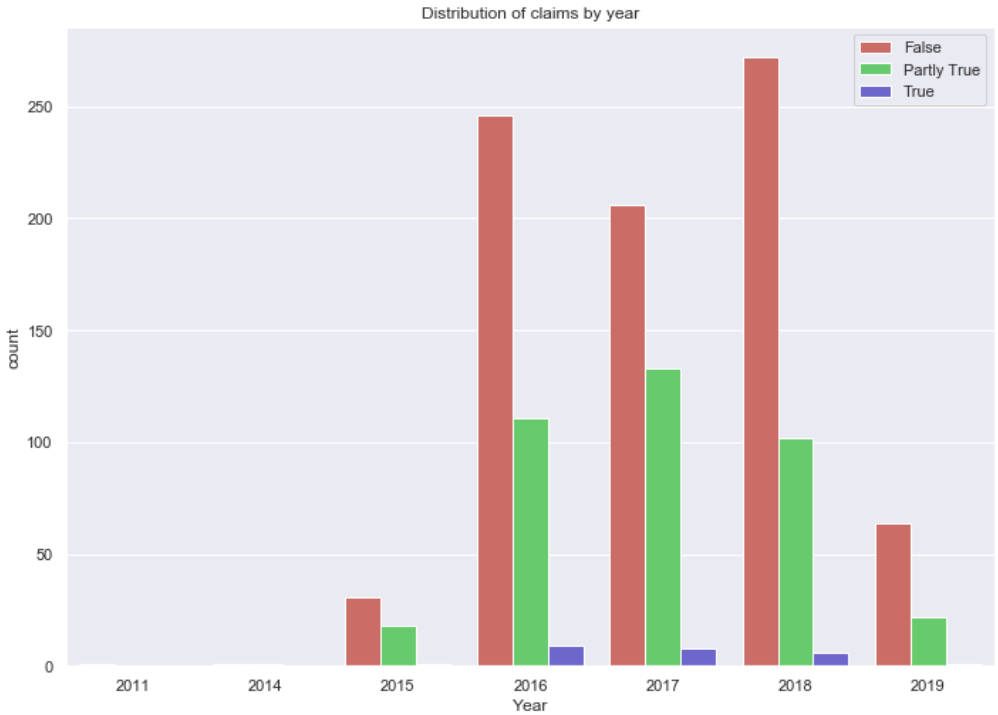

Figure 5: Distribution of Claims by Year

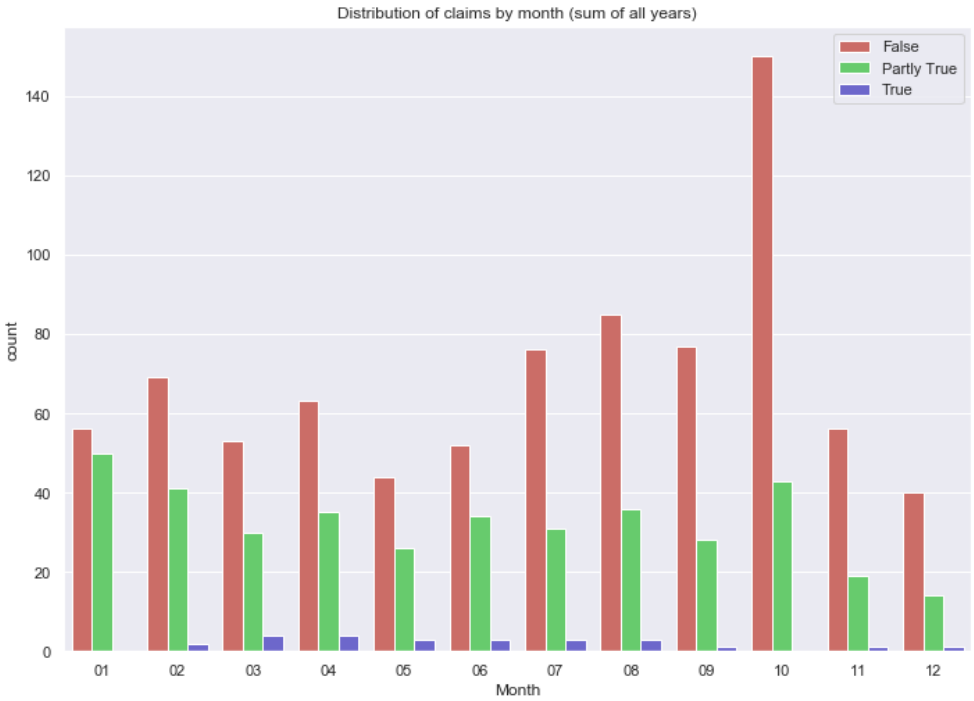

Figure 6: Distribution of Claims by Month across all Years

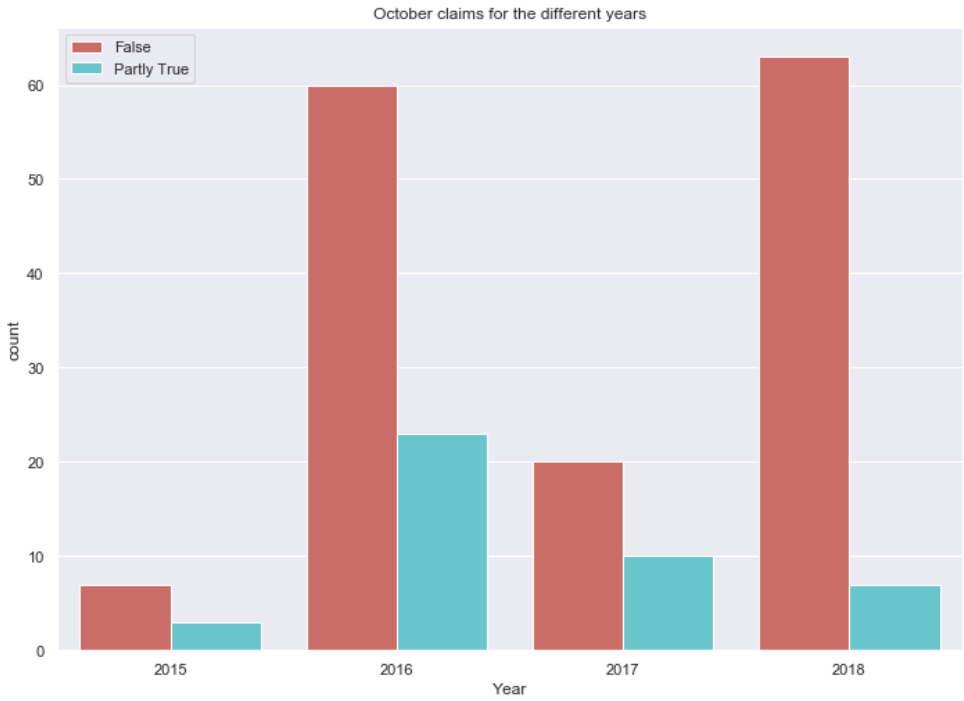

Figure 7: Trump's Claims in October over each Year

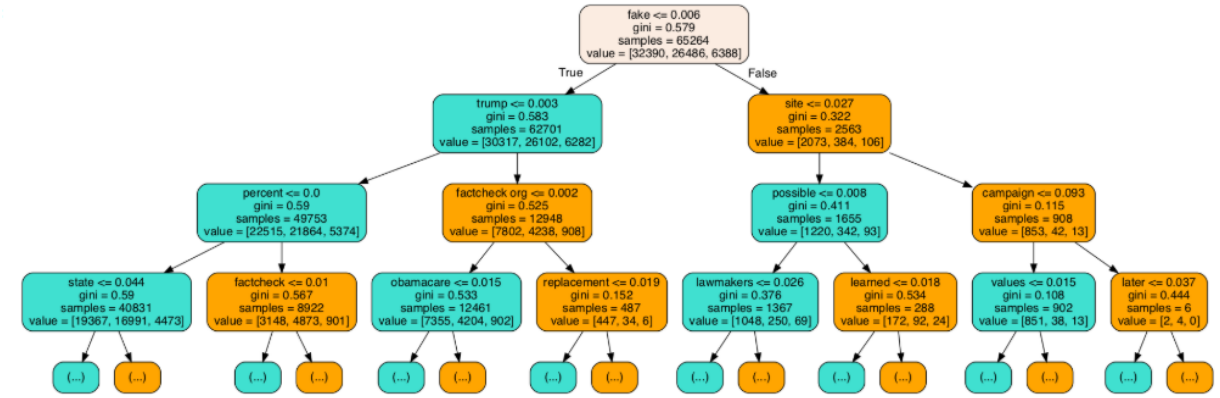

Figure 8: Decision Tree Results

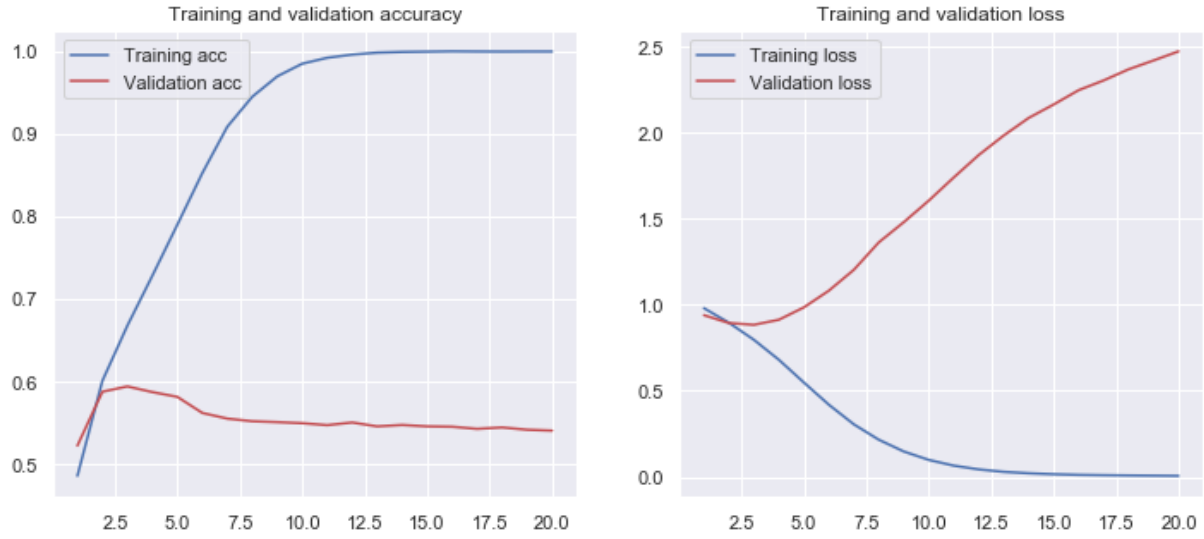

Figure 9: Training and Validation accuracies for each Neural Net