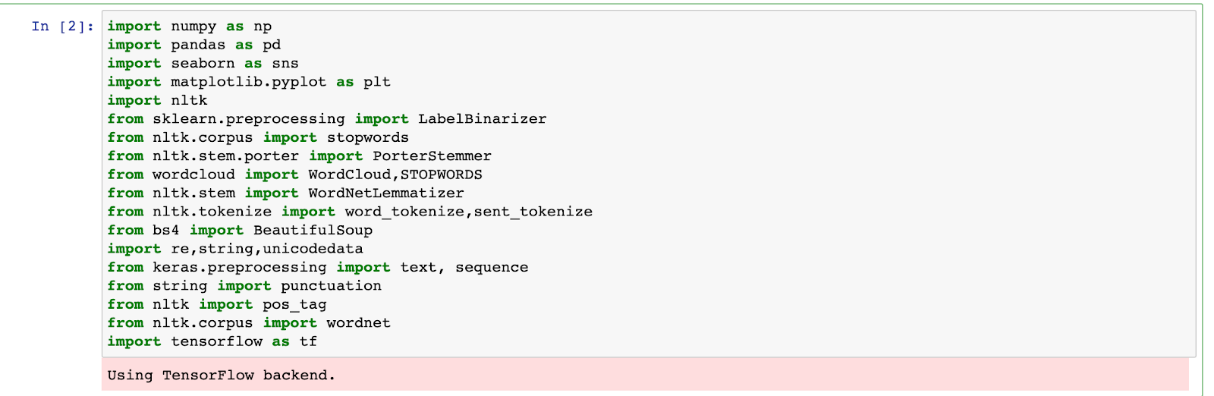
**IMPLEMENTATION AND RESULTS**

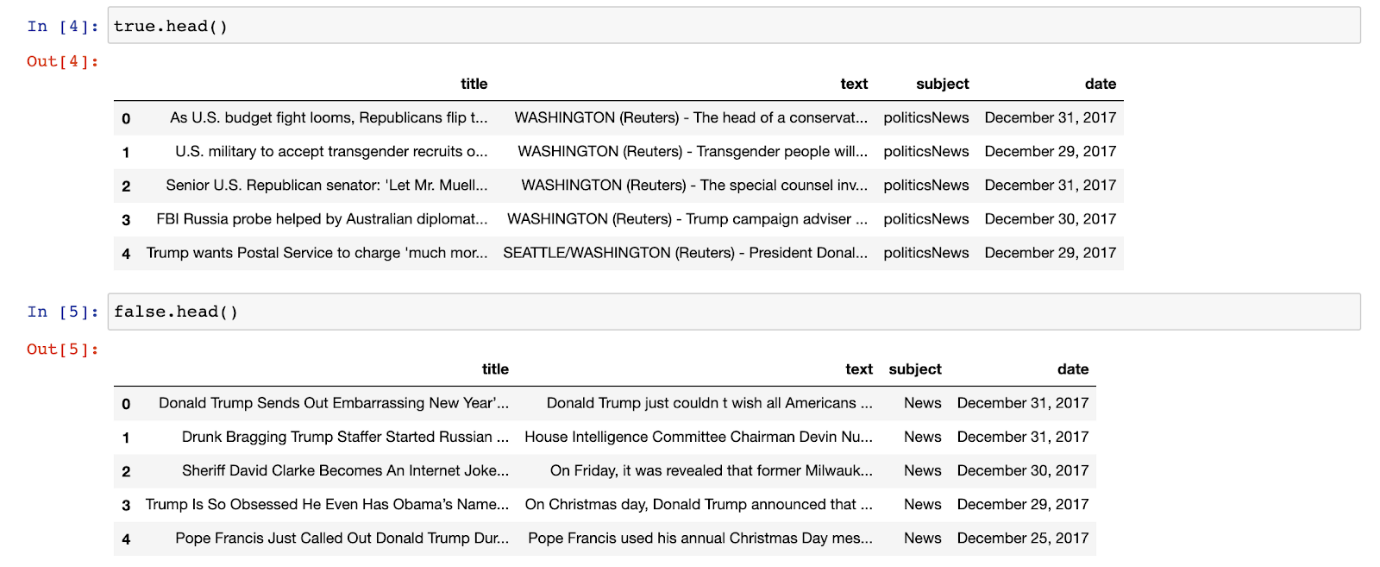
Packages we used were



First we imported our dataset into the notebook using pandas as shown below

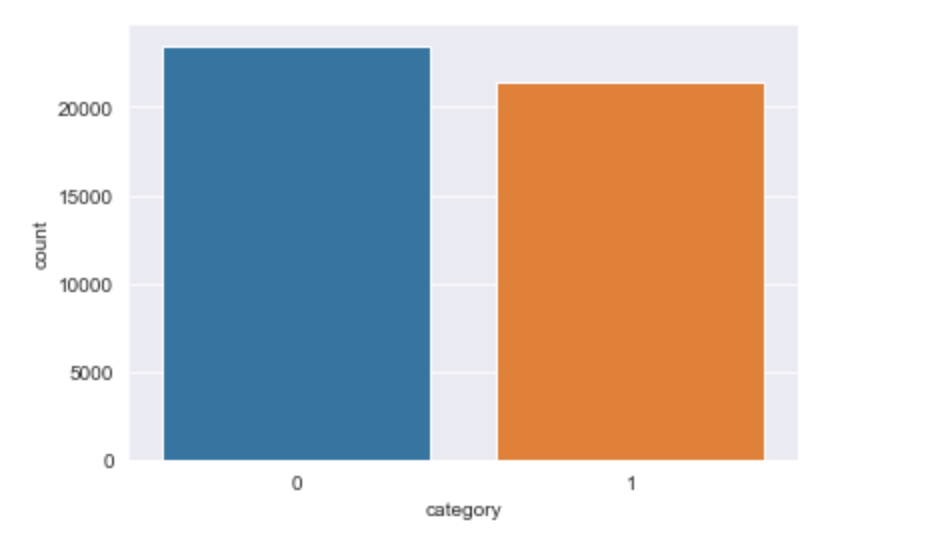


Viewing our true and false dataframes



We append both true and false dataframes to get all of the data into one dataframe

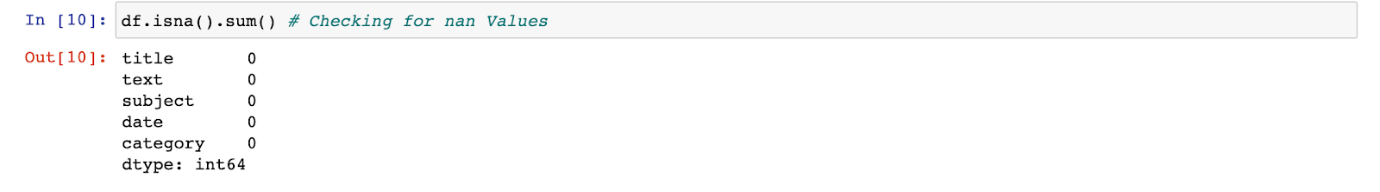
Next, we visualised the ratio of the labels of our data i.e Fake and True News to see if our dataset is balanced



*Figure 1: Ratio of the fake and true news labels where 1 - True news and 0 - Fake news*

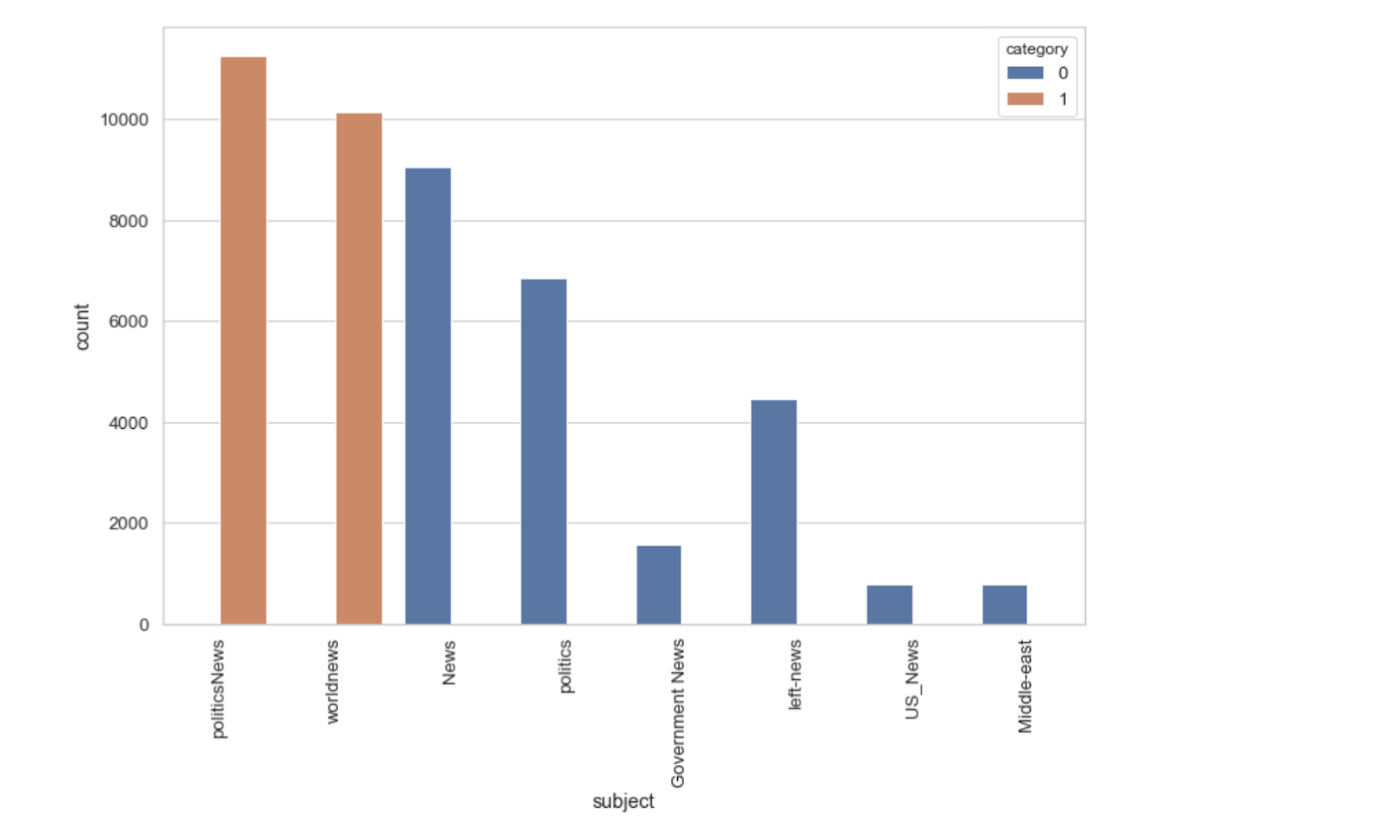
As you can see from the bar graph, our dataset is well balanced with close to equal count of data for both the labels

Next, we checked to see if the dataset contained any null values using df.isna().sum(). The results are shown below:



From this, we can see that our dataset has no null values.

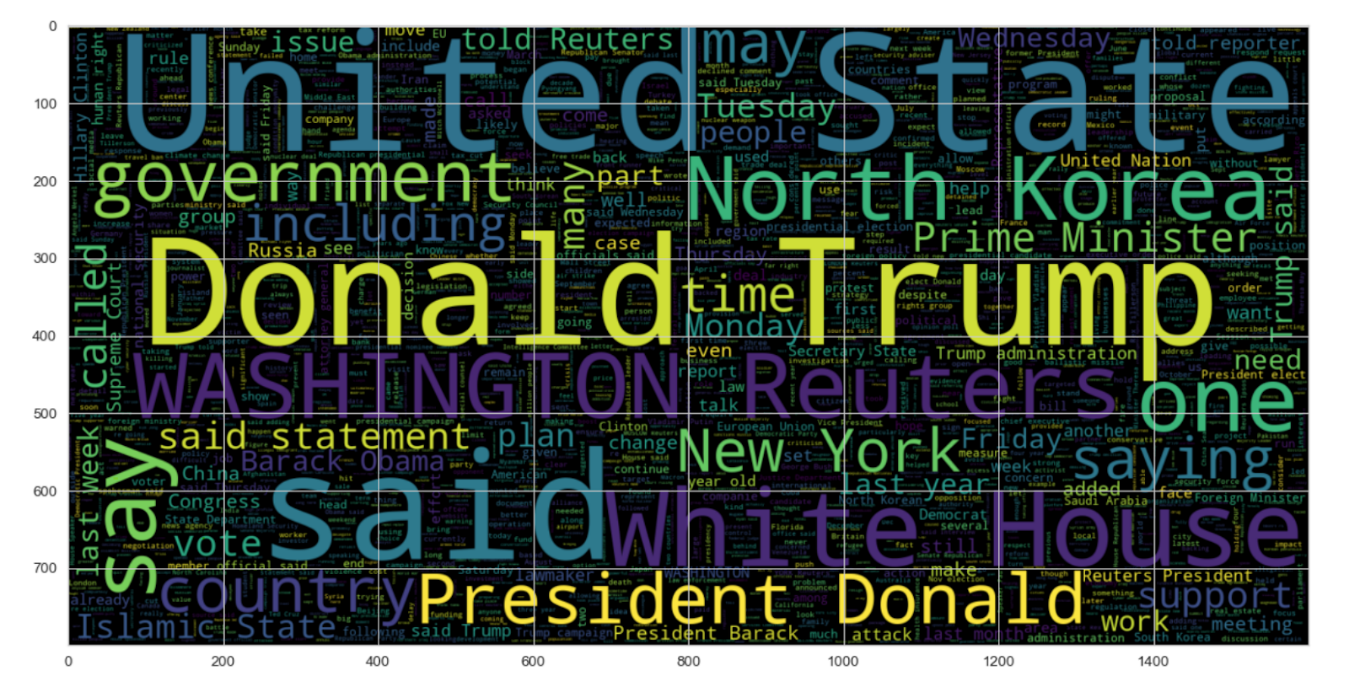
Our dataset has a column called ‘subject’ which basically explains the genre of the news. We perform some exploratory data analysis to see the counts of each type



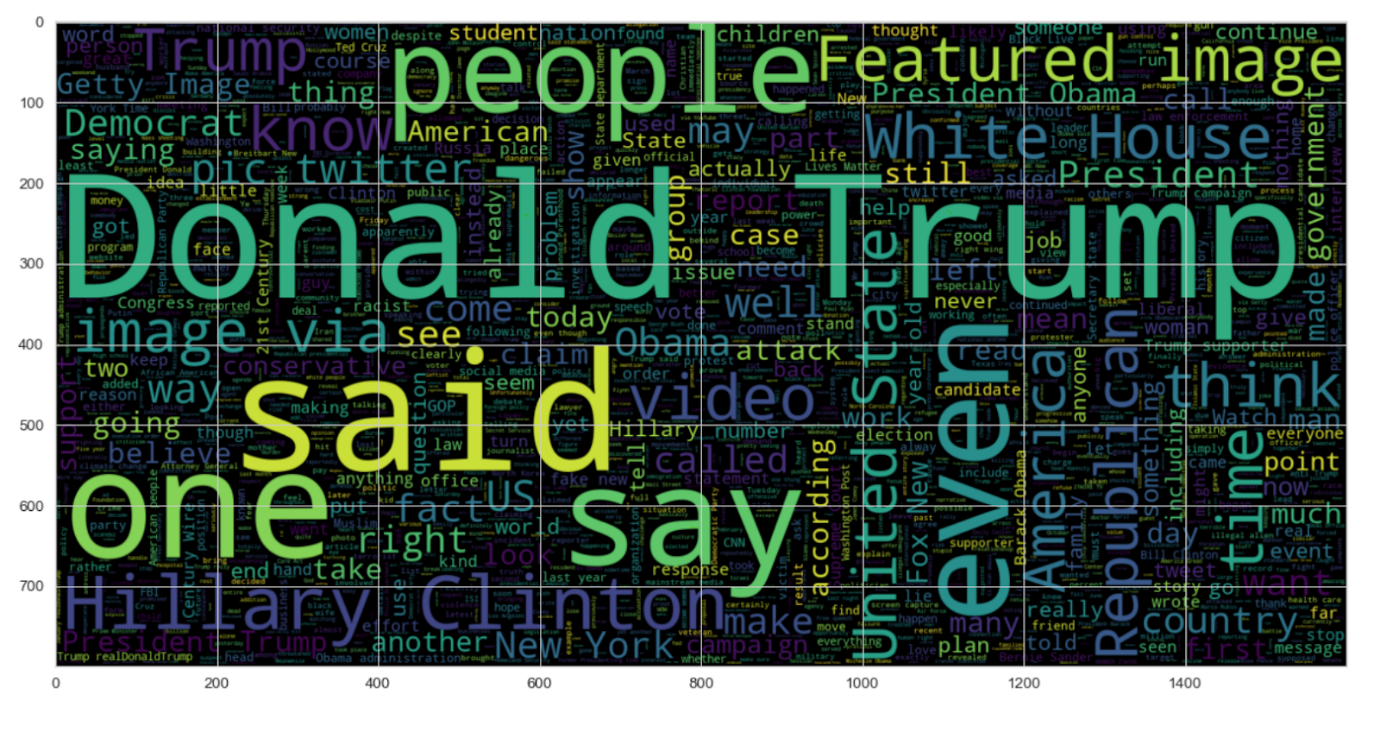
*Figure 2: Subject vs count bar graph of the dataset. Category 0 refers to Fake news while 1 refers to True news. True news has subjects: politicsNews, worldnews and Fake news has subjects: News, politics, Government News, left-news, US\_News and Middle- east*

From the bar graph above we can see that in the category of True news, ‘PoliticsNews’ has the highest count whereas for Fake news, the ‘News’ subject has the highest count. In overall, the subject ‘politicsNews’ has the highest count in the dataset. Next, we do some data cleaning by removing stop words and other unnecessary characters which may be present in our data. To remove stop words, we use NLTK Library: nltk.corpus.stopwords.words("english"). We also concatenate the ‘title’ and ‘text’ column for our data analysis.

Now, we perform more data analysis to understand our dataset better. First, we generated a Word Cloud. A word cloud is a collection, or cluster, of words depicted in different sizes. The bigger and bolder the word appears, the more often it’s mentioned within a given text and the more important it is.

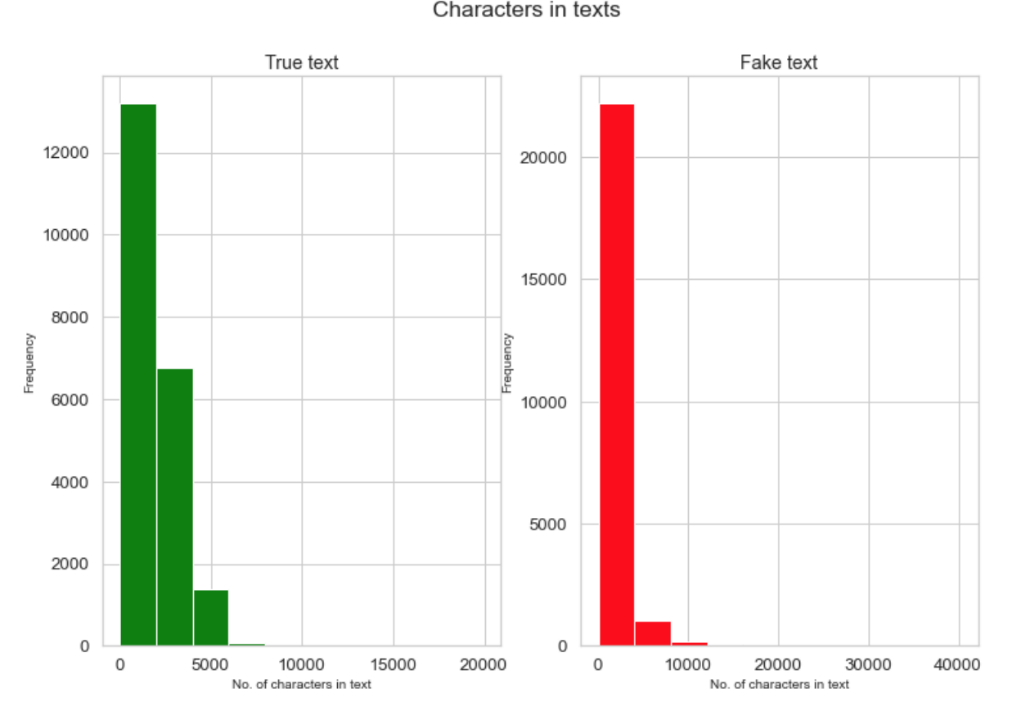


*Figure 3: Word Cloud for True News. You can see words like “Donald”, “Trump” and “United States” seem to occur a lot*

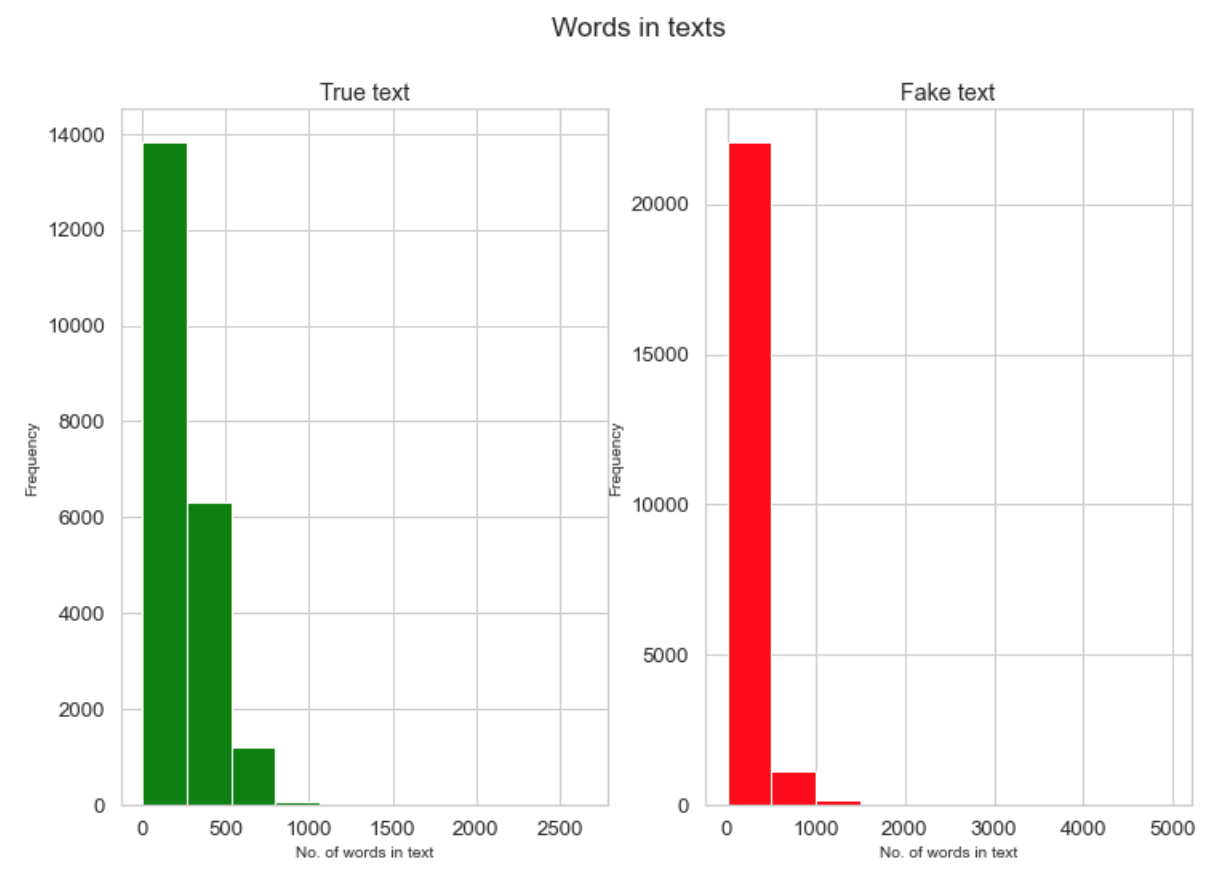
**

*Figure 4: Word Cloud for Fake News. Once again, ‘Donald Trump’ seems to be a common occurring word amongst many others such as ‘people’, ‘Featured’, etc.*

Next we look into the number of characters and words the text has. From the results below, we can see that 2500 characters in text seem to be the most common in the True News category while around 5000 characters in text are most common in the Fake News category.



*Figure 5:  Bar graph depicting No of characters vs Frequency for True and Fake text*

**

*Figure 6:  Bar graph depicting No of words vs Frequency for True and Fake text*

We then see the most commonly occurring words in our dataset. Below are the results:

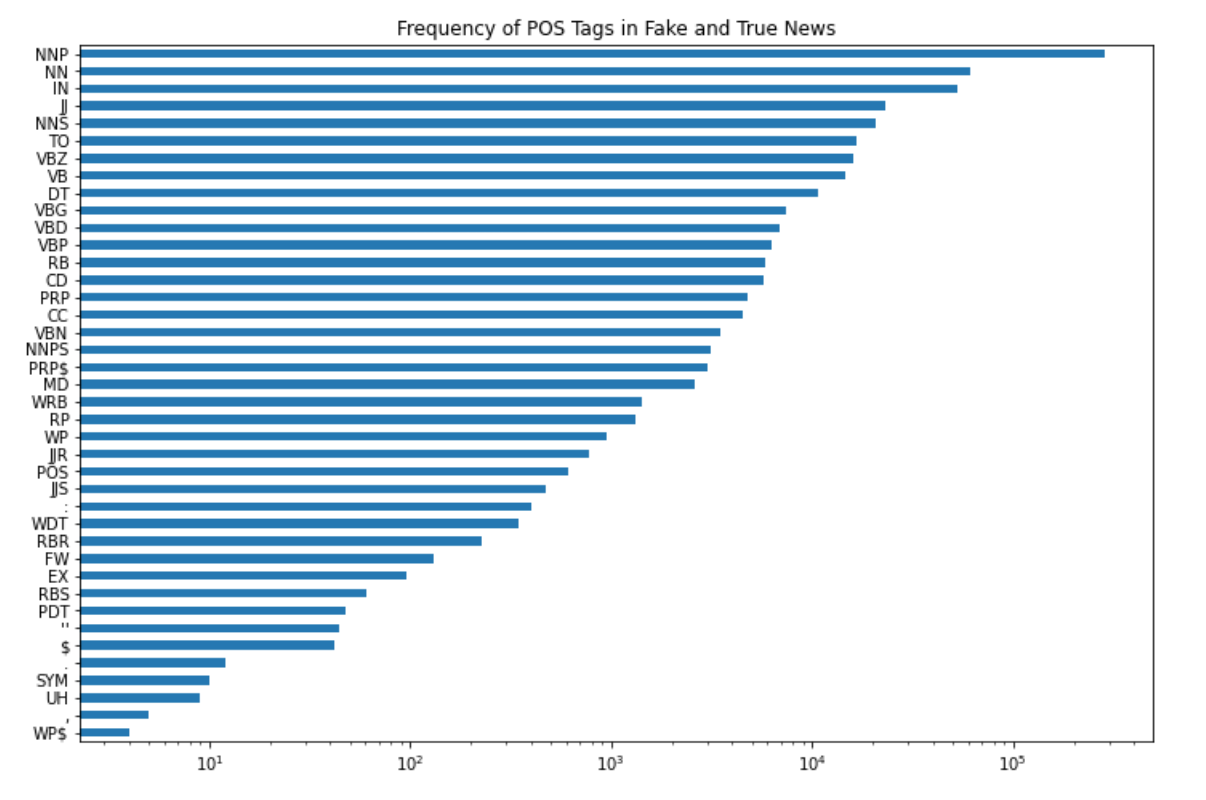


Trump seems to be the most common occurring word present in the dataset with over 111k occurrences.

**POS Tagging**

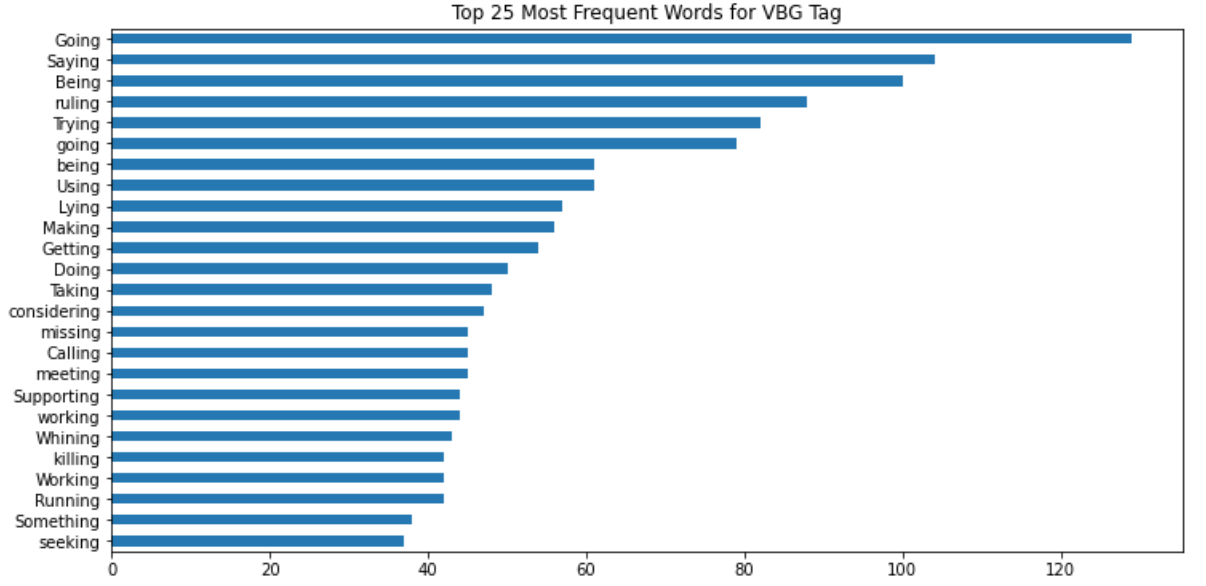
Next we analyse the various POS tags present in our dataset which contains Fake and True news.

We iterate through the words in our dataset, tag their part of speech and finally get the total count of all the tags. We import pos\_tag from the nltk library.

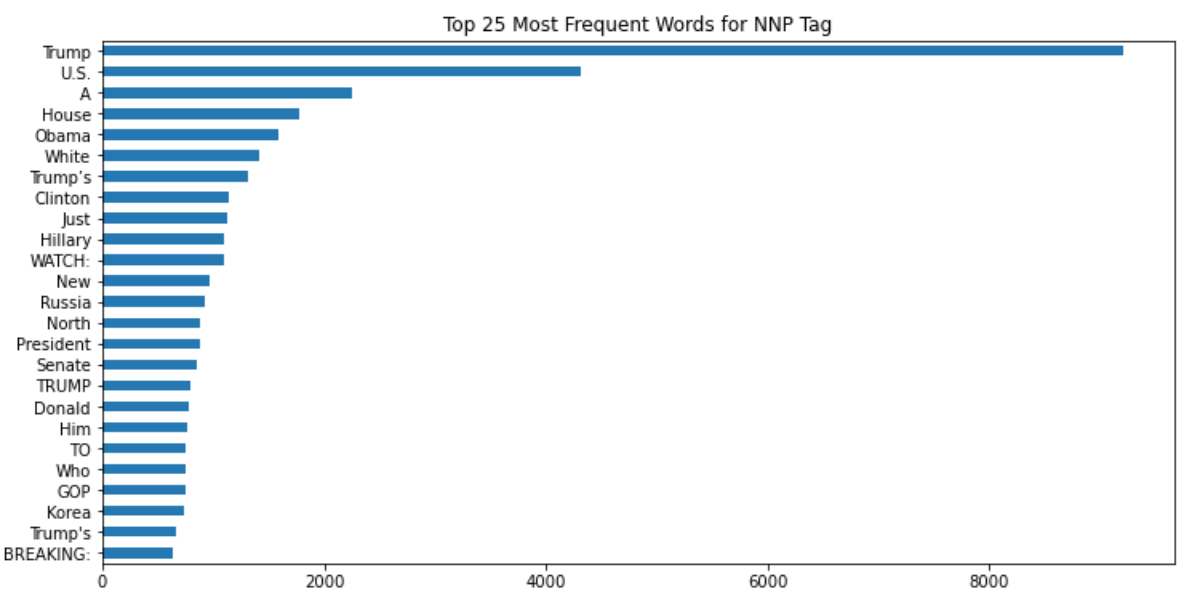


*Figure 7: The Frequency of POS tags present in our dataset*

We also go one step deeper and see the most common words occurring for a few POS Tags. Below you can find the top 25 words for the POS Tags VBG and NNP. VBG is Verb, gerund or present participle and NNP is Proper noun, singular. (From Penn Treebank)



*Figure 8: The top 25 words with the POS tag VBG in our dataset*

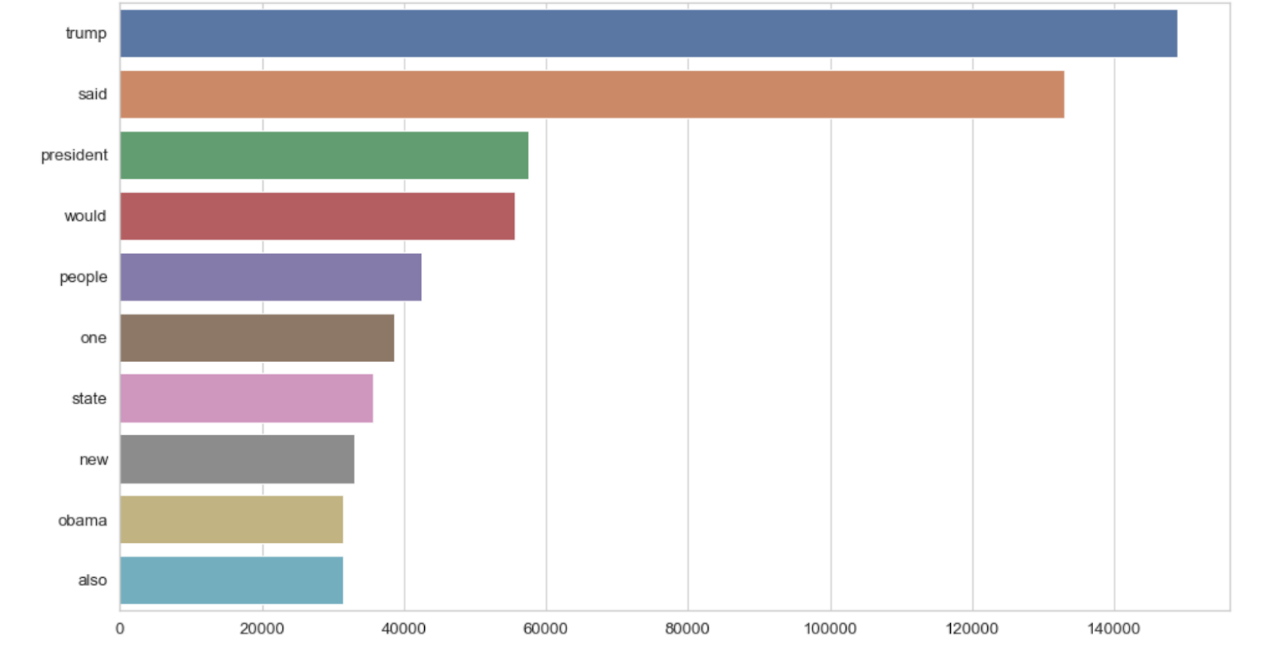
**

*Figure 9: The top 25 words with the POS tag NNP in our dataset*

**N-gram Analysis**

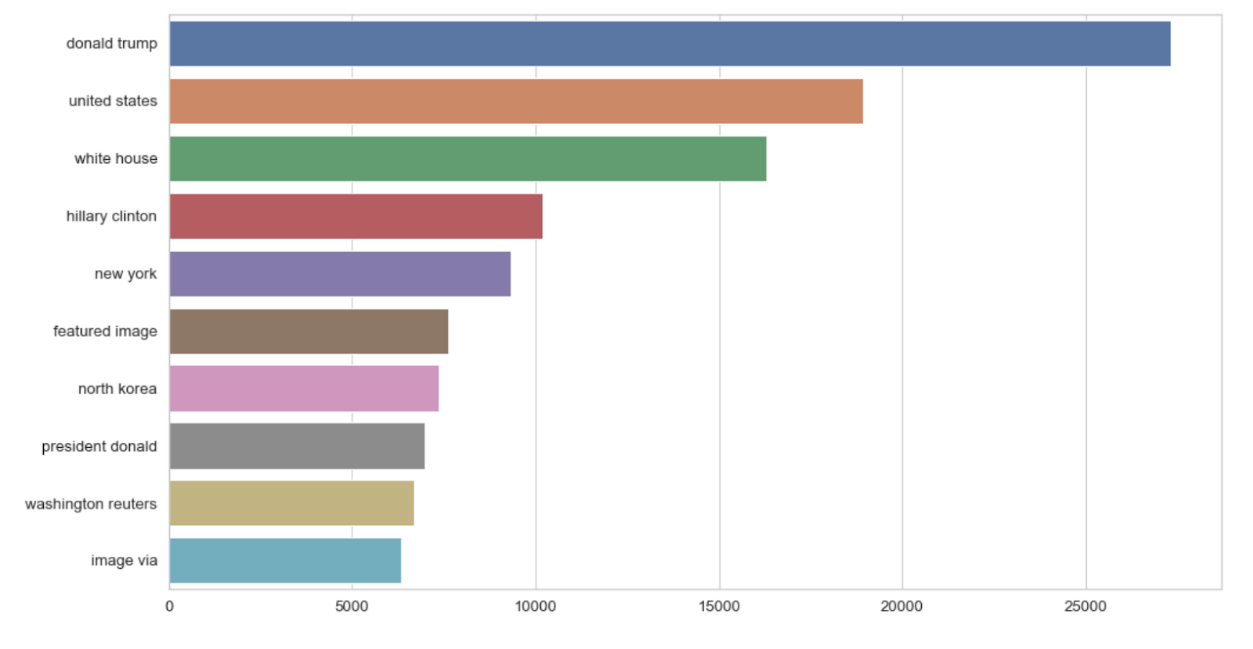
N-grams of texts are extensively used in text mining and natural language processing tasks. An n-gram is a contiguous sequence of n items from a given sample of text or speech. an n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram"; size 3 is a "trigram".

First, we see the unigrams of our dataset



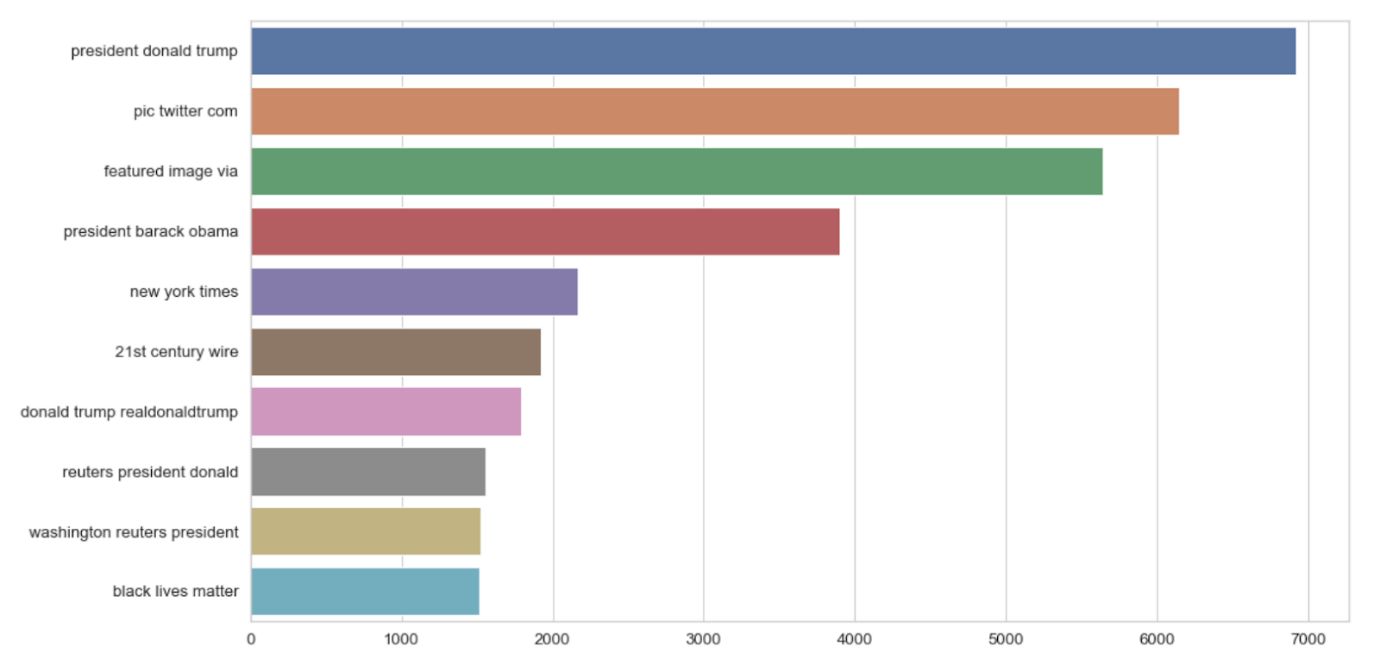
*Figure 10: Unigrams present in the dataset. You can see Trump is the most frequently occurring Unigram*

Next, we looked into the bigrams present as show below



*Figure 11: Bigrams present in the dataset. You can see ‘donald trump‘ is the most frequently occurring bigram*

Finally, we see the trigrams present in the dataset as shown below:

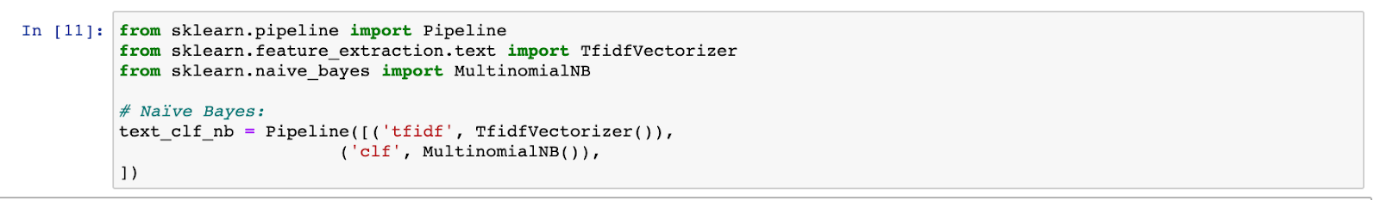
**

*Figure 12: Trigrams present in the dataset. You can see ‘president donald trump’ is the most frequently occurring trigram*

**Implementing our first Model : Naïve Bayes**

We split our dataset into our train and test set with a 70-30 ratio. X\_train and X\_test contain the title of the news and y\_train and y\_test contain their respective labels, i.e, whether True or Fake.

We create a pipeline where first TF-IDF is first applied followed by the Naive Bayes classifier.



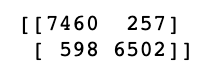
Text data requires special preparation before you can start using it for predictive modeling. The text must be parsed to remove words, called tokenization. Then the words need to be encoded as integers or floating point values for use as input to a machine learning algorithm, called feature extraction (or vectorization).

To calculate word frequencies, we use the popular method, TF-IDF. This is an acronym that stands for “Term Frequency – Inverse Document Frequency” which are the components of the resulting scores assigned to each word. Term Frequency summarizes how often a given word appears within a document. Inverse Document Frequency on the other hand downscales words that appear a lot across documents.

After feature extraction, we pass our transformed set to the Naive Bayes classifier. We train it using our Train set and finally check how the model performed using the test set.

Below are our results for the Naive Bayes model. Our model attained an accuracy of 94.2%.

Confusion matrix of our model



Actual Fake, Predicted Fake: 7460 (TP)

Actual Fake, Predicted True: 257 (FP)

Actual True, Predicted Fake: 598 (FN)

Actual True, Predicted True: 6502 (TN)

Classification report of our model:

