Motivation

Fake news is one of the most critical problems facing the internet since its early days. Fake news is false or misleading information intentionally disguised as legitimate news, and can have devastating consequences. The most recent incident of fake news causing such consequences was during the COVID-19 pandemic, in which there was a rapid spread of fake news containing conspiracy theories and public health misinformation about COVID-19 and its vaccine. Such misinformation led to people taking ineffective or dangerous remedies, ignoring health guidelines, and for some not to get vaccinated.

The Dataset

We chose the Fake and real news dataset from kaggle. This dataset contains 40,000 news articles. There exist 2 csv files “Fake.csv” and “True.csv”. Fake.csv contains 20,000 fake news articles, while True.csv contains 20,000 true news articles.

The dataset contains the following columns:

- **title** : the title of the article
- **text** : the content of the article
- **subject** : the subject of the article

  This column can have the following values:
  - politicsNews
  - worldnews
  - News
  - politics
  - Government News
  - left-news
  - US_News
  - Middle-east
- **date** : The date at which the article was posted

Our Goal

We are going to utilise this dataset to build a classifier which given the content of an article, can classify whether said article is fake or not.

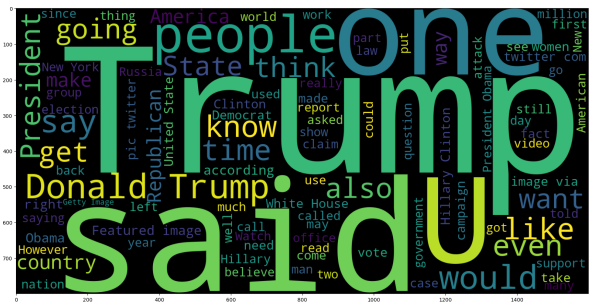
EDA

Before anything else, we would like to inspect the data, and analyse it, which will help us in the later stages to come up with a successful model.
In this stage, we will try to see which good assumptions we can make about the data, come up with questions, and try to answer them through inspecting the data.
And those were the 10 most common words:
(the number besides each words represents how many times it was used)

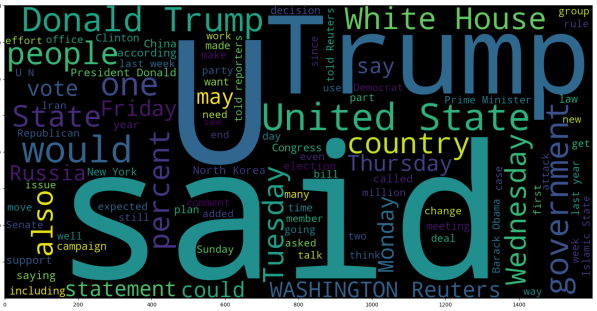
1. Are the words used in fake articles different from those used in true ones ?

To answer this question, we have made a word cloud for the 100 most common words used in both fake articles and true articles..

Fake Articles



True Articles



trump	73933
said	31013
people	25963
president	25586
would	23427
one	22935
us	22049
clinton	18011
obama	17813
like	17621

said	98985
trump	42577
us	1137
would	31514
reuters	28306
president	25533
state	18753
government	17976
states	17634
new	16785

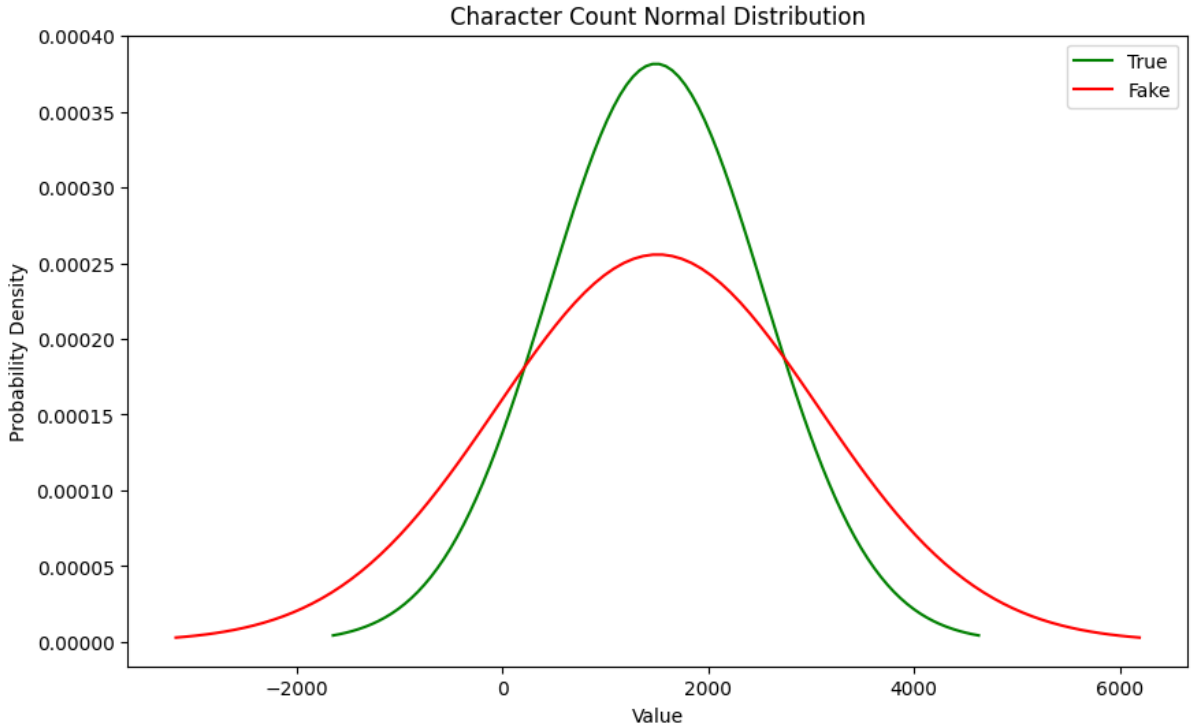
Conclusion:

Although we can clearly see that there are words that do appear a lot in both real and fake articles like the word “trump” and “said”. But there do exist words that appear far more in fake articles such as “obama” and “clinton”, and words that appear far more in real articles such as “state” and “government”.

So we can conclude that the BOW (bag of words) technique for text vectorization would be a valid option.

2. Does the length of fake articles differ from the length of true articles ? (length = character count)

We have computed the mean and standard deviations for the character counts for the fake articles, and for the true articles, and here is the plot of the normal distributions:



Answer: yes, the distribution of the length of fake articles does differ from the distribution of the length of true articles. Although the average length of an article is the same, the fake article’s character count has higher variation.

Conclusion:

Although we are going to use BOW to vectorize the text, an alternative and a very interesting approach would be to extract features directly from each article such as: the character count, the average word length, and other features like this . From our analysis we have shown that at least for the character count, this feature would be a differentiating factor. And this approach would also be viable.

Text Preprocessing

We do execute the following 5 steps in-order to clean the text and preprocess it before building the model.

1) remove non alphabetical character such as digits, punctuations and special characters
2) Normalise all characters to lowercase
3) tokenize the text: split the text into a list of tokens (the words that make up the text)
4) remove the stop words from the text
5) lemmatize the text: convert each token into its proper lemma

Text Vectorization

We need to represent each article with a vector. We use the BOW (bag of words) technique to do so, more specifically we use the CountVectorizer from sklearn library, which is a variant of BOW. Each word present in the training dataset will be a feature, and the value of the feature will be the number of occurrences of said word in the article.

Model Training

We have split the data into a training set (80%) and a testing set (20%).
We have chosen 2 algorithms: Logistic regression, and Adaboost consisting of tree stumps (one split decision trees) as base learners.
We have used grid search on the training set for both models in-order to tune the hyperparameters.
The following hyperparameters were found to give the best results:
Logistic regression: {'C': 0.1, 'penalty': 'l1', 'solver': 'liblinear'}
Adaboost: {'learning_rate': 1.0, 'n_estimators': 100}

Test Results

Logistic Regression

Accuracy: 0.9952115812917595

Precision: 0.9969795037756203

Recall: 0.9937634408602151

F1-score: 0.99536887452881

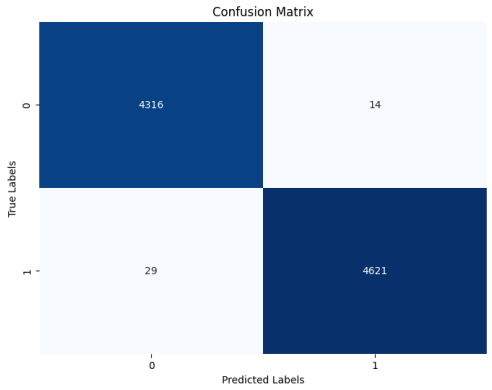
Adaboost

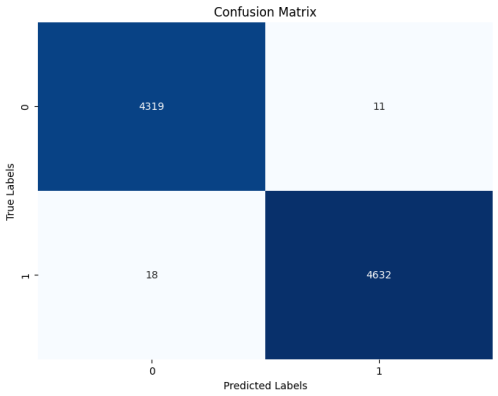
Accuracy: 0.9967706013363029

Precision: 0.9976308421279345

Recall: 0.9961290322580645

F1-score: 0.996879371569999





Conclusion:

Both algorithms achieved great results on the testing set, both achieving accuracy of 99%. The adaboost algorithm performed slightly better, but the difference is negligible.