Fake News Detection With Machine Learning

By Austin Willoughby and Victoria Mullin

Introduction

Fake news is false information that takes the appearance of real news with the aim of deceiving individuals and swaying public perception, often in the realm of politics. One well-known fake news website is called The Onion, famous for its fabricated stories. While The Onion's material is indeed untrue, it represents the light side of fake news, as it's intent is satirical in nature rather than deceptive. On the other end of the spectrum fake news takes on a more maleficent nature. This dark side of fake news was evident during the 2016 election where fake news articles were disseminated on social media platforms in the United States by foreign nations seeking to influence the democratic process. Today, nearly 50% of readers report seeing fake news on social media at least once a day, especially on popular sites like Facebook and TikTok. The prevalence of such nefarious media is concerning, and does not require an election to be detrimental. Circumstances such as the ongoing Covid-19 pandemic, where accurate information is important to public health, can make a vulnerable target for misinformation campaigns. Slowing or stopping the spread of fake news is thus desirable to our society as a whole. The goal of this project is to create a natural language processing (NLP) model which accurately distinguishes fake news articles from real news articles.

The data used in this project comes from a Kaggle competition for detecting fake news. As seen in Figure 1, the training dataset consists of an ID variable, the title of the news articles, the author, the text portion of the news article and a label of 1 if the article is unreliable and 0 if the article is reliable. There is also a testing dataset which has the same variables as the training dataset without the labels and a third dataset which is named "submit" which contains all of the labels for the testing dataset. It should be noted that only the training set was used in this project, as it was not explicitly stated that the "submit" dataset was the target of the "test" dataset. There is a relatively balanced number of classes in the training set which consists of 10,385 real news articles and 10,169 fake news articles.

Figure 1

id	title	author	text	label
0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1

Data Preprocessing

When exploring the data, there were 39 records that had a missing text variable out of the 20,800 total records. These NaNs were changed into a string with a single space in order to allow for preprocessing. Next a column named "total" was created by combining the "title," "author," and "text" columns into one string. This column was necessary so that both the author and title of an article could be implemented into the model at the same time as the text.

Next a word count was conducted for every row within the "total" column. Of the 20,800 articles in the training dataset the smallest word count was one, with the largest article containing 24,234 words (including author and title). The overall median word count was 565 words. The distribution of the word counts can be seen in Figure 2 below.

Figure 2

With the outer structure of the articles now known, the inner structure next needed to be assessed. The frequency of individual words was determined, and viewed. It was observed that common words such as "the" and "it" were the most frequent words found, while the least frequent words were those which had non-alphabetic notations included in their string, such as quotation marks or percentage signs. Both of these word types were undesirable, and needed to be removed (Vivek).

To do so, regular expressions were used to remove all non-alphabetic characters, text was set to lowercase, and sentences were then split at white spaces in order to separate individual words. With words now separate, stop words (i.e. common words such as "the") could be removed from articles. These actions left the dataset with lowercase words containing only alphabetic characters (Vivek).

Next a lemmatizing process was completed in order to categorize different inflected forms of the same word. This process groups words such as “express” and “expressly” together and thus allows slightly different words with the same meaning to be counted together. With the data preprocessing completed, the words were joined back into their respective articles and added into a list known as the corpus (i.e. the full collection of texts). The corpus was then used to process the data and create visualizations (Vivek).

With the data preprocessed, the corpus could now be processed into a bag of words. A bag of words model requires the extraction of text features from text into a format which can be used for machine learning. The model entails a vocabulary of all unique words within a corpus and a quantitative measure of the occurrence of these words. Figure 3 below shows an example bag of words taken from a small subset of the Fake News Corpus (Brownlee).

Figure 3

	according	agencies	aide	antonio	assessments	behavioral	big	bush	campus	case	...	war	way	weapons	wedding
0	0.003538	0.000000	0.005898	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.004719	...	0.000000	0.00000	0.000000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002832	0.000000	0.002832	0.000000	...	0.000000	0.00425	0.000000	0.002832
2	0.000000	0.003141	0.000000	0.000000	0.002357	0.003141	0.000000	0.005497	0.000000	0.000000	...	0.000000	0.00000	0.006283	0.000000
142	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.00000	0.000000	0.000000
143	0.000000	0.000000	0.000000	0.004486	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.004486	0.00000	0.000000	0.000000
1200	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.00000	0.000000	0.000000

6 rows × 93 columns

In order to process data into a bag of words a tokenizer can be used. A tokenizer takes a character sequence, such as a paragraph, and separates it into pieces called “tokens.” It also has the option to drop specific characters, such as punctuation; however, this was already completed in the steps mentioned above. A “token” is an occurence of a sequence of characters in text that are parsed and grouped together as a meaningful unit for NLP such as a word, phrase, or even whole sentences (“Tokenization”). Both the CountVectorizer and TF-IDF functions used in this model are examples of common SK-Learn tokenizers (Brownlee).

The CountVectorizer is a tokenization algorithm normally used in NLP. It not only tokenizes collection of text documents but also helps build a vocabulary of known words. It also encodes new documents using that vocabulary (Brownlee).

First we created an instance of the CountVectorizer class, and used the fit() function to learn the vocabulary of the desired document. Next the transform() function was used on the same document to encode words as vectors. After transformation the vector we receive will have the entire length of the vocabulary with a count for the number of times those words appeared in that document. The type of vector we receive will be sparse vectors (Brownlee).

As seen in Figure 3 above, a bag of words dataset will have as many columns as there are unique words in the corpus, with each row representing an individual article. Cell values in a bag of words represent a measure of word frequency, which can range from a word count, to a TF-IDF measure. In the case of this fake news analysis, the TF-IDF measure was used. TF-IDF

is a common quantification measure that stands for term frequency – inverse data frequency, and tends to be much more useful than a simple word count. Term frequency is the number of times a word shows up in an article, divided by the total number of words in that article, or:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

Conversely inverse data frequency is the log of the number of articles divided by the number of articles that contain a given word, or:

$$idf(w) = log(\frac{N}{df_t})$$

The TF-IDF of a given word in a given article is thus the TF multiplied by the IDF, or:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$ (Maklin)

An example of an article processed into its keywords and corresponding TF-IDF values is seen in Figure 4 below (note that only the top 10 keywords are displayed for space).

Figure 4

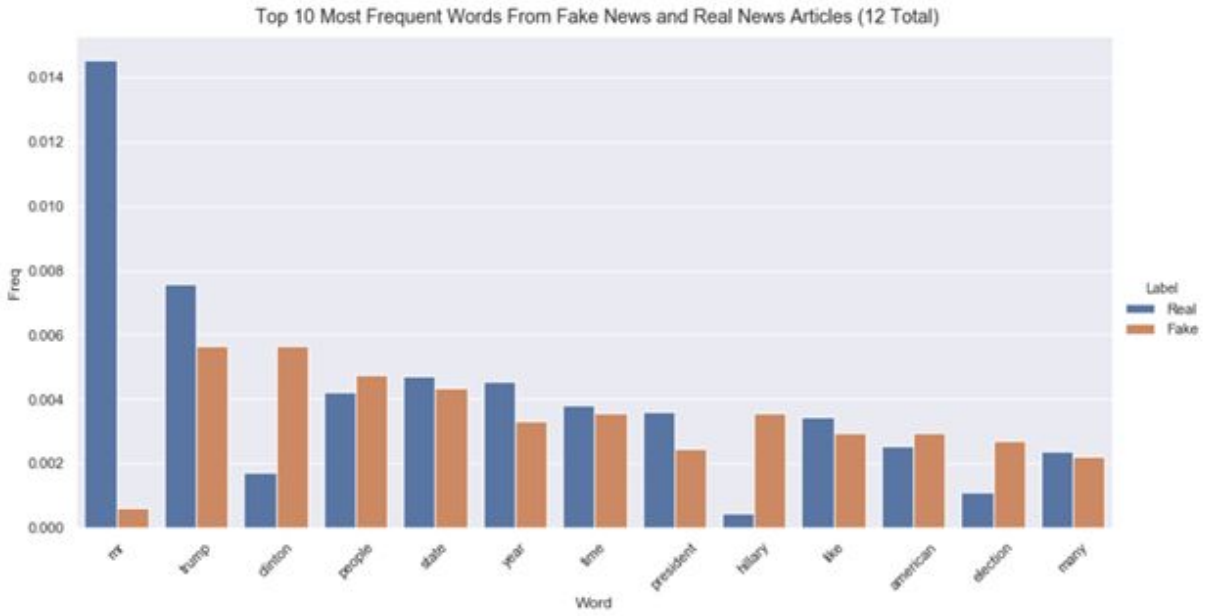
```
Text:
trying time jackie mason voice reason week exclusive clip breitbart news jackie discus looming threat north korea exp
lains president donald trump win support hollywood left u need strike first decides bomb whole country behind everybo
dy realize choice thing jackie say except hollywood left get nauseous trump win left fall love minute bombed better r
eason jackie explains like transgender toilet jackie say surprise hollywood celebrity support trump strike syrian air
field month infuriated say might save life mean anything saved environment climate change happiest people world still
jackie say got nothing hollywood celebs got tough life country watch jackie latest clip follow daniel nussbaum twitte
r dznussbaum

Keywords:
hollywood 0.394
clip 0.232
trump win 0.22
explains 0.204
strike 0.18
say 0.175
left 0.172
mason 0.146
save life 0.139
win 0.139
```

While TF-IDF values of words are useful, they can also be applied to combinations of words. Using word combinations allows for a bag of words model which takes into account some contextual structure. In order to do this, bigrams (i.e. two word combinations) and trigrams (i.e. three word combinations) can be taken into account. It is possible to take into account larger word combinations; however the processing to do so becomes unruly (Vivek).

While TF-IDF is more useful than word counts within the model, word counts are useful for visualizing the differences in text. Figure 5 below displays the top ten most frequent words

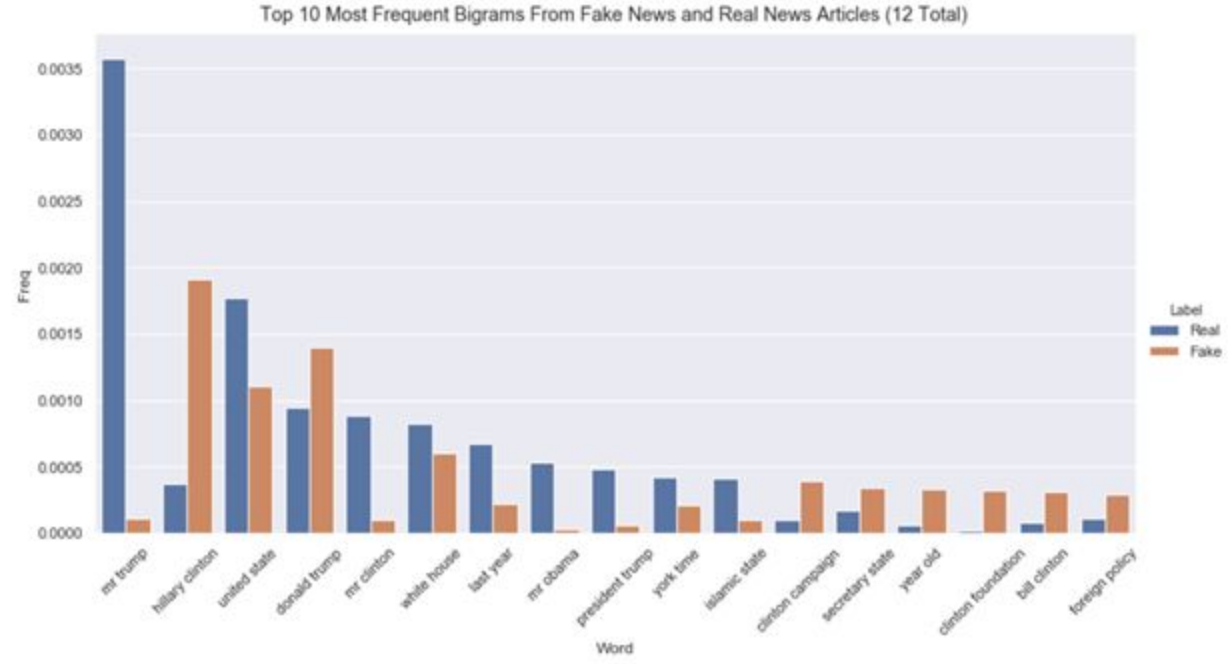
used in both the fake and real news categories (with a total of 12 words present since neither top ten list was exactly the same).

Figure 5



Here we can see that most words tend to have similar frequencies; however the words “mr” and “hillary” vary drastically between real and fake news. Figure 6 below shows a similar graph of bigrams.

Figure 6



The bigram graph displays a much more diverse usage of word combinations between fake news and real news than the single word graph showed. These differences go to show the value of using multi word combinations when creating a bag of words model, as they generally introduce more variance into the dataset.

Corpus word counts can also be visualized through a word cloud as in Figure 7 and 8 below. A word cloud is an assembly of words within an image where each word's text size corresponds to its frequency within a corpus. Figure 7 and 8 below represent word counts from a fake news and real news article respectively.

Figure 7

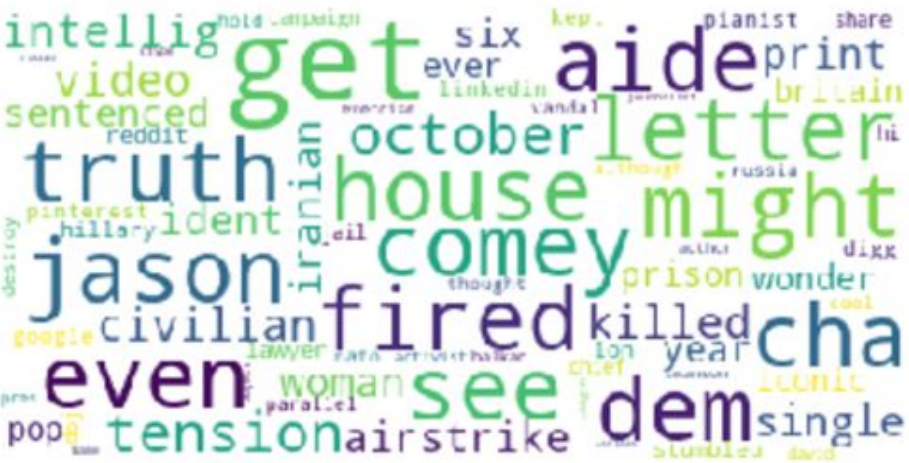
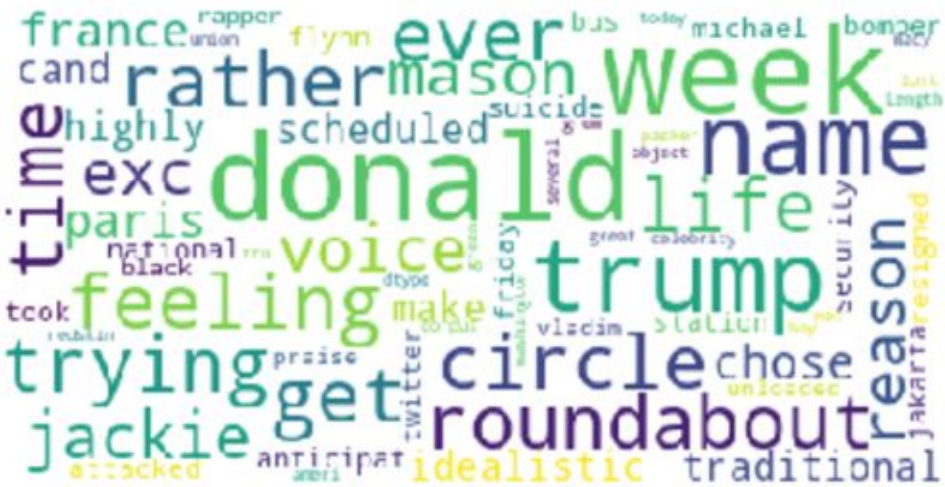


Figure 8



Naive Bayes

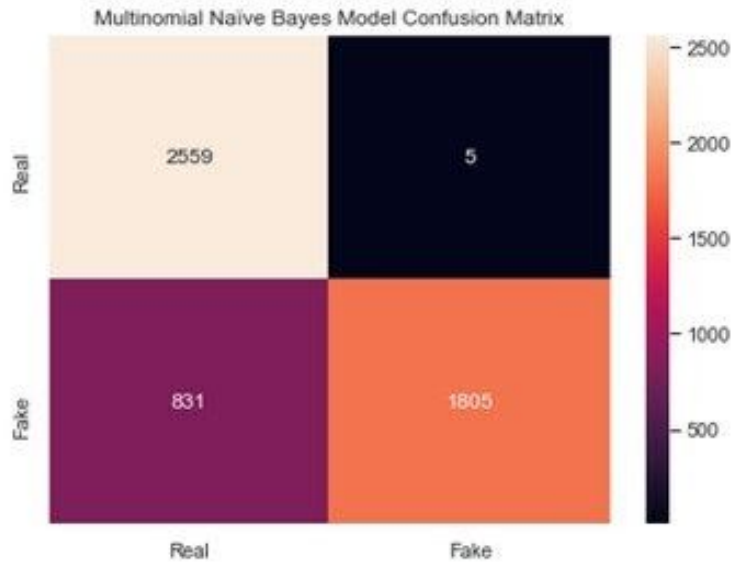
Naive Bayes is an algorithm based on applying Bayes theorem with an assumption that every feature is independent of the others, in order to predict the category of a given sample. They are probabilistic classifiers, therefore will calculate the probability of each category using Bayes theorem, and the category with the highest probability will be output. Naive Bayes algorithms have been demonstrated to be fast, reliable and accurate in a number of applications of NLP (Sanghi).

Multinomial Naive Bayes

Multinomial Naive Bayes classifier is a specific instance of a Naive Bayes classifier which uses a multinomial distribution for each of the features. This is the event model typically used for document classification. So,we decided to use a multinomial naive bias model for our project ("Multinomial Distribution").

The confusion matrix after using countVectorizer as the tokenizer for the words and Multinomial Naive Bayes as the classifier algorithm can be seen in Figure 9 below.

Figure 9



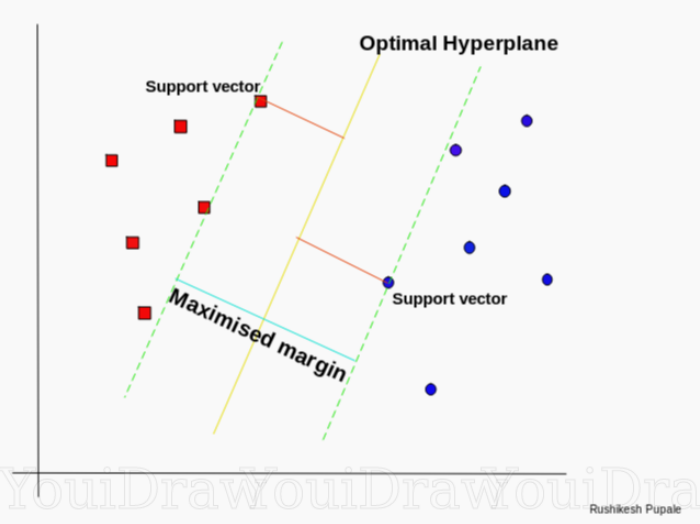
Multinomial naive bayes resulted in an accuracy of 83.9% with only 5 false positives (i.e. real news misclassified as fake) and 831 false negatives. This classifier is much more likely to misclassify real news as fake than the opposite which is likely better since in these cases the author can simply report and fix this misclassification whereas a false negative will likely never be exposed. Although it does not surpass the 90% threshold that is typically desirable for classification accuracy, this model is somewhat effective at determining whether a given article

is real or fake. However, it may not be the best classifier for this problem and will be compared to support vector classifiers next.

Support Vector Machine

Support vector machine (SVM) is a machine learning algorithm that can be used for both classification and regression problems but is mostly used for classification. Each record is plotted as a point in n-dimensional space, where n is the number of features in the dataset. The main goal of SVM is to find a line, plane, or hyperplane for higher dimensions that best separates the two classes. This is done by first finding the nearest points to the line from each class - these points are called the support vectors. Next, the distance between the dividing line and the support vectors is calculated - this is called the margin. The goal of SVM is achieved when the margin is maximized for both classes, meaning that the separation between the two classes is maximal. This can be visualized in the Figure 10 below (Pupale).

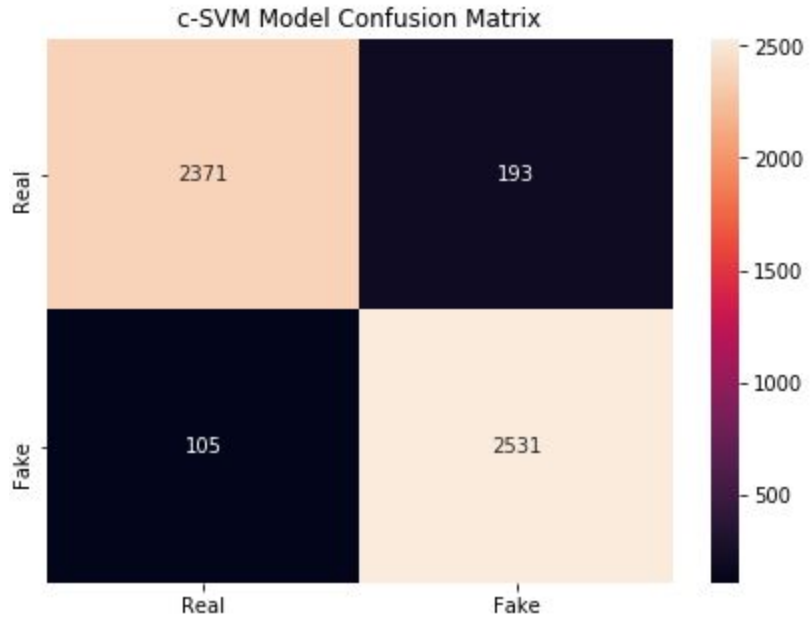
Figure 10



There are two types of SVC algorithms available in SK-Learn: c-SVC and nu-SVC. These two different versions of SVC are very similar to each other. nu-SVC uses a hyperparameter nu that is the upper bound on the percentage of points that can be in margin and a lower bound on the percentage of points that are support vectors. For example, if Nu is set to 0.02 it is guaranteed that at most 2% of the training samples will be misclassified and at least 2% of the training samples will be support vectors. Similarly, c-SVC uses the hyperparameter C to regularize the model which introduces a squared 2l penalty on misclassification. Nu is bounded between 0 and 1 compared to C which ranges from 0 to infinity and therefore is more interpretable than the C parameter (sklearn). Both models are essentially the same exact model, but with different regularization values.

In order to determine the optimal nu value, kernel function, and degree (for the polynomial kernel function “poly”), a grid search was used. The highest accuracy for both the train and test data resulted from using a nu value of 0.5, degree of 1, and the ‘rbf’ kernel. The accuracy for the training data was 99.96% and the accuracy for the testing data was 94.27%, indicating that there is no overfitting or underfitting. The Nu-SVC confusion matrix can be seen below in Figure 11.

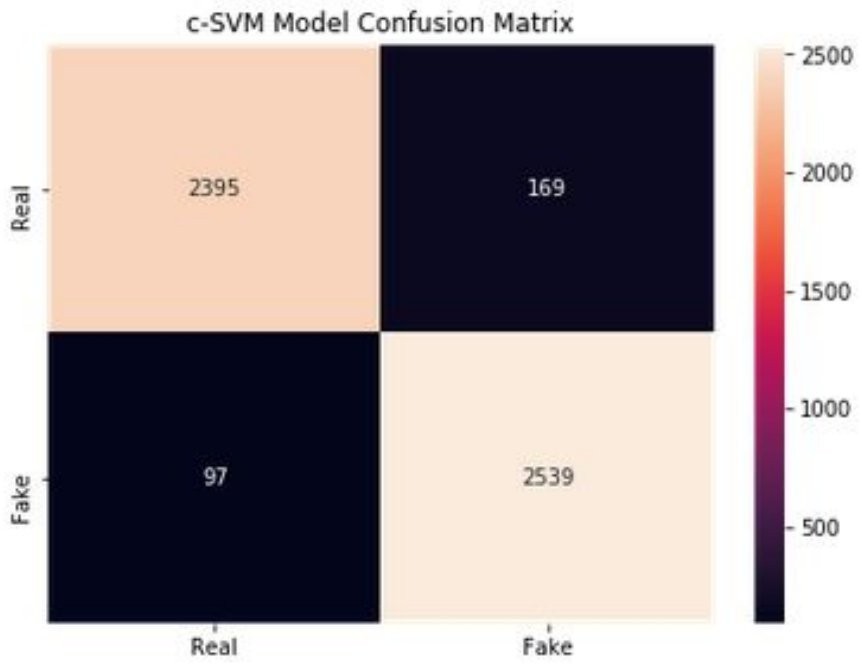
Figure 11



Nu-SVC was successful at classifying real and fake news articles by using the title, author, and the text. There were more cases where real news articles were misclassified as fake news articles, also known as false positive, than vice versa. Although misclassifications are undesirable, in this case, it is better to have more false positives than false negatives. If a real news article is misclassified as fake, this can be fixed by the author reporting it, however, a false negative will likely never be exposed. With such a high accuracy, this model can effectively be used to classify fake and real news articles.

Similarly to nu-SVC, a gridsearch was run on the C value of c-SVC in order to determine the optimal regularization coefficient. The best C value was determined to be 3, with an accuracy of 94.9%. The c-SVC confusion matrix can be seen in Figure 12 below.

Figure 12



As seen in Figure 10, the c-SVC model combined with a bag of words dataset including bigrams and trigrams was able to effectively distinguish fake news articles from real news articles. The model was more likely to misclassify real news articles as fake news articles than the reverse. This classification error is a false positive. In the case of fake news classification, a false negative (i.e. classifying fake news as real news) is likely more costly than a false positive, as the author of a real news article could simply contact the publishing authority in the event of a false positive, while a false negative results in the dissemination of misinformation to the public. The c-SVC model misclassified fake news as real news 1.9% of the time. If determined acceptable by media outlets, future applications of this model could potentially allow for more false positives in order to reduce false negatives.

Conclusion

Overall, Naive bayes resulted in an accuracy of 83.9% which is the lowest of the three models. SVC type 1, which uses the C-parameter, achieved the highest accuracy which was 94.9%. Similarly, SVC type 2 – which uses the Nu parameter -resulted in an accuracy of 94.27%, which is very close but just slightly worse than type 1. Naive Bayes was more likely to misclassify a false negative (i.e. fake news classified as real news) than a false positive. On the other hand, both SVC models had higher false positive rates than false negative, meaning they both misclassified real news as fake news more often than the opposite. This is worth noting if either models were to be used in a real world application since, in this case, a false negative is more detrimental than a false positive– which could be reported and fixed by the author.

SVC likely performed better than Naive Bayes on this dataset since the latter uses linear separation while SVC uses the ‘rbf’ kernel which is nonlinear and incorporates interactions between the fields. Additionally, SVC is effective in cases where the number of features is greater than the number of samples, which is the case with our tokenized bag of words dataset. This high dimensionality leads to different classes being located in two very different areas of the feature space. An SVC can thus easily generate a hyperplane between the high dimensional classes (Li et al).

Moving forward, other classification techniques could be used - such as logistic regression and K-nearest neighbors - to compare our results to. However, we have surpassed our goal of having an accuracy of at least 90% with c-SVC which effectively distinguishes between real and fake news. SVC is also efficient in higher dimensional spaces, such as our dataset.

References

Brownlee, J. (2019, August 7). How to Prepare Text Data for Machine Learning with scikit-learn. Retrieved April 24, 2020, from https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

Fake News: Fake News & Social Media. (2020, February 25). Retrieved April 22, 2020, from https://libguides.com.edu/c.php?g=649902&p=4556540

Heidenreich, H. (2018, August 24). Natural Language Processing: Count Vectorization with scikit-learn. Retrieved April 18, 2020, from https://towardsdatascience.com/natural-language-processing-count-vectorization-with-scikit-learn-e7804269bb5e

Heidenreich, H. (2018, August 16). Introduction to Word Embeddings. Retrieved April 18, 2020, from https://towardsdatascience.com/introduction-to-word-embeddings-4cf857b12edc

Kumar, D. (2019, September 17). Demystifying Support Vector Machines. Retrieved April 15, 2020, from https://towardsdatascience.com/demystifying-support-vector-machines-8453b39f7368

Multinomial distribution. (2020, April 9). Retrieved April 24, 2020, from https://en.wikipedia.org/wiki/Multinomial_distribution

Tokenization. (2009, July 4). Retrieved April 17, 2020, from https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

Pupale, R. (2019, February 11). Support Vector Machines(SVM) - An Overview. Retrieved April 15, 2020, from https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

Sanghi, D., & Sanghi, D. (2019, January 14). Applying Multinomial Naive Bayes to NLP Problems. Retrieved April 24, 2020, from https://www.geeksforgeeks.org/applying-multinomial-naive-bayes-to-nlp-problems/?ref=rp

sklearn.svm.NuSVC. (n.d.). Retrieved April 23, 2020, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.NuSVC.html

Tf-idf :: A Single-Page Tutorial - Information Retrieval and Text Mining. (n.d.). Retrieved April 23, 2020, from http://www.tfidf.com/

Maklin, Cory (2019, May 5). TF IDF | TFIDF Python . Retrieved April 26th 2020, from Example Https://towards datascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76

Vivek, Sowmya (2018, December 17). Automated Keyword Extraction from Articles using NLP. Retrieved 4/19/2020, from https://medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34

Brownlee, Jason (2019, August 7). A Gentle Introduction to the Bag-of-Words Model. Retrieved 4/22/2020, from https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Li, Yaoyong et al. Adapting SVM for Natural Language Learning: A Case Study Involving Information Extraction. Retrieved 4/30/2020, from https://gate.ac.uk/sale/nle-svm/svm-ie.pdf