

## 1. Introduction and data analysis

### 1.1 Introduction

The BMJ, a peer-reviewed medical trade journal published by the British Medical Association, released a letter titled ‘Going viral: doctors must tackle fake news in the covid-19 pandemic’ [1]. It stated that misinformation is defined as false information that is communicated without deliberate malice while disinformation is communicated with intent to deceive. The letter’s conclusion of ‘evidence shows that healthcare professionals can stop the spread of false information by refuting or rebutting misleading health information on social media and by providing appropriate sources to accompany their refutation’, highlights the urgent need for engineers to develop scalable tools that could help media organisations or social media users directly, to reduce the detrimental effects of fake news.

In general, it is difficult to ascertain the motivation behind people who spread either misinformation or disinformation. Although, the term ‘fake news’ is popularised in recent times, this problem of people spreading false information is an age-old issue. That being said, the widespread use of social media platforms has amplified this problem. The exact cause of the creation of echo-chambers is a widely debated topic especially since the Brexit referendum and the Trump election. For example, DiFranzo et al. stated that the filter bubble effect is due to the user’s network and past engagement behaviour, such as clicking only on certain news stories [2]. His study concluded that the existence of echo chambers is not the fault of the news-feed algorithm of social media platforms but the choices of users themselves. This debate alone could form an entire study and is not within the scope of this report, but this example provides an important context to the goal of this project.

In short, regardless of the exact motivations of social media users who spread fake news or the cause of the prevalence of echo chambers, understanding the use cases of machine learning algorithms under the umbrella of Natural Language Processing (NLP), could be an important first step of mitigating the damaging effects of widespread fake news in social media.

This project goal is to design and implement an algorithm that could classify social media post as ‘fake’ with a high accuracy. During the process of developing an algorithm, justifications have to be made to explain why certain models were selected and others are discarded. The process of evaluating the use cases of different models and iterating the chosen algorithm is crucial to develop a solid understanding.

### 1.2 Problem Characterisation

As the problem at hand is a classification problem between a real or fake tweet, defining what constitutes a fake tweet is a logical first step. Firstly, a fake tweet could consist synthetic multimedia such as artworks or snapshots presented as real imagery. An infamous type of synthetic multimedia is known as ‘deep fakes’ where a celebrity’s face could be overlayed by a face generated by a deep-learning method called generative adversarial networks to represent the real celebrity. Secondly, a fake tweet could contain a digitally manipulated image. These images are typically edited via a software like Adobe Photoshop. Lastly, a fake tweet could contain a real but out-dated photograph that is re-posted in order to be associated with a current event.

It is important to note that this project involves analysing and training textual data only. It does not involve any analytical work on images or videos.

### 1.3 Data Characterisation

This project’s data comes from the dataset of social media posts used in the ‘MediaEval2015’ event. It contains 14,483 training set posts, 3,781 test set posts. Within those posts, there are 399 unique images/videos that are referenced. These posts are related to 11 news events. Each post contains text

and mentions a linked or embedded image or video. However, as stated in Section 1.2, MediaEval2015 image features are not used to limit the scope of the project.

The training and test data are formatted in a tab delimited UTF-8 CSV form which contains columns of ‘tweetId’, ‘tweetText’, ‘userId’, ‘imageId(s)’, ‘username’, ‘timestamp’ and ‘label’.

Table of basic info of training and test data:

	Training data	Test data
Event count	<div> <div>images</div> <div> <div>boston</div> <div>bringback</div> <div>columbianChemicals</div> <div>elephant</div> <div>livr</div> <div>malaysia</div> <div>passport</div> <div>pigFish</div> <div>sandyA</div> <div>sandyB</div> <div>sochi</div> <div>underwater</div> </div> <div> <div>546</div> <div>131</div> <div>185</div> <div>13</div> <div>9</div> <div>501</div> <div>46</div> <div>14</div> <div>9695</div> <div>2621</div> <div>402</div> <div>112</div> </div> </div> <div>Name: tweetId, dtype: int64</div>	<div> <div>images</div> <div> <div>eclipse</div> <div>garissa</div> <div>nepal</div> <div>samurai</div> <div>syrianboy</div> <div>varoufakis</div> </div> <div> <div>277</div> <div>77</div> <div>1353</div> <div>218</div> <div>1769</div> <div>61</div> </div> </div> <div>Name: tweetId, dtype: int64</div>
Frequency of mentions		
Label count	<div> <div>fake</div> <div>real</div> <div>humor</div> </div> <div> <div>6742</div> <div>4921</div> <div>2614</div> </div> <div>Name: label, dtype: int64</div>	<div> <div>fake</div> <div>real</div> </div> <div> <div>2546</div> <div>1209</div> </div> <div>Name: label, dtype: int64</div>
Label bar chart		

Table 1.

In the first row of the Table 1., denoted by ‘event count’ counts the respective events contained in the tweets in both the training data (left column) and the test set (right column). This counting of events is achieved by utilising the ‘nunique’ function in conjunction to the splitting function for returning the number of unique elements in the specified axis. The splitting the string of the ‘imageId(s)’ is done with the use of underscores as the positions of splits. This method is used because the image IDs of tweets are displayed in the structure of ‘EventName\_etc’ such as ‘SandyA\_fake\_46’. Whether the tweet is classified as real or fake is represented with the ‘label’ category already. Hence, the only useful information worth keeping from each image ID is the event name itself.

To clarify the context of the eleven events mentioned in the training data: ‘boston’ refers to the Boston Marathon bombing; ‘bringback’ refers to the event of Nigerian school girls being abducted by Boko Haram; ‘columbianChemicals’ refers to the hoax that claimed there was an explosion at a chemical plant in Louisiana; ‘elephant’ refers to a rock formation that looks like an elephant situated off the coast

of Iceland; ‘livr’ refers to a mobile application; ‘malaysia’ refers to the missing MH370 flight accident; ‘passport’ refers to a passport hoax; ‘pigFish’ refers to a fake pigfish photo that claimed to be newly discovered; ‘sandyA’ and ‘sandyB’ both refer to Hurricane Sandy which was the most destructive hurricane in 2012; ‘sochi’ refers to the 2014 Winter Olympics; and finally, ‘underwater’ refers to an underwater bedroom.

The same counting operation was applied to the test set which contains six events: ‘eclipse’ refers to a solar eclipse; ‘garissa’ refers to militant attack on a Kenyan University in 2015; ‘nepal’ refers to the severe Nepal earthquake in 2015; ‘samurai’ refers to the ‘samurai ghost’ photo hoax; ‘syrainboy’ refers to a photo of a Syrian boy in a war-torn country; ‘Varoufakis’ refers to a fake video of the then Greek finance minister Yanis Varoufakis produced by a German satirical TV host.

In the second row of Table 1., two bar charts based on the numeric data from the table’s first row are shown. They give visual representations of the frequency of mentions of events in both training and test sets. For the third row of Table 1., the number of fake and real tweets are counted. For the training set, there is an extra label of humour tweets. This row shows that there is sampling bias because both training and test sets are skewed towards having more fake tweets than real ones. 66% of the training data are fake tweets after converting humour tweets as fakes, and 68% of the test data are fake tweets. In the last row of the table, bar charts based on the label counts from the row above are plotted. Finally, as not all tweets are in English, the viability of adding a language translation will be explored in the Algorithm Design section.

## 2. Algorithm Design

### 2.1 Pre-processing data

After understanding the basic composition of both training and test sets in the Data Characterisation section, a detailed process of cleaning data is needed. This pre-processing stage forms the majority of this project because the implementation of algorithms does not require much code. That being said, a significant amount of time is also spent on studying academic papers on the respective advantages and disadvantages of different algorithms before trying several algorithms in the implementation stage to compare and iterate in order to arrive at a final algorithm with the highest accuracy.

For the cleaning data stage, the first step involved converting ‘humour’ labels to ‘fake’ ones in the training data. There is no need to convert any labelling for the test set as there is only ‘real’ and ‘fake’ labels. However, the label count for test set does change as duplicated tweets and rows with missing indices were removed. The changes in label counts are shown in the first two rows of Table 2.

Secondly, a logical step is to remove duplicated tweets. This was done with the use of the ‘drop\_duplicates’ function. Tweets with missing indices were also removed by using the ‘drop\_na’ function. These operations were performed on both training and test sets. The outcomes are shown in the third and fourth row in Table 2. below. The ‘shape’ function returns the dimensions of the data frame in the form ‘(row, columns)’. Clearly, the number of columns does not change as the categories of data (e.g. tweetId, tweetText, etc.) have not been modified.

	Training set	Test set
Label count before cleaning	fake 6742 real 4921 humor 2614 Name: label, dtype: int64	fake 2546 real 1209 Name: label, dtype: int64
Label count after cleaning	fake 8597 real 3779 Name: label, dtype: int64	fake 2507 real 1199 Name: label, dtype: int64
Dimensions before cleaning	(14277, 7)	(3755, 7)
Dimensions after cleaning	(12376, 7)	(3706, 7)

Table 2.

Moreover, the 're' library was used which provides regular expression matching operations. This library enables the conversion of all letters to lower case, removal of twitter handles and URLs. The conversion of all characters to lower case is needed because inconsistent capitalisation could cause significant problems when classifying from a large data set [3]. This decrease in accuracy comes from the scenario where the machine learning model treats a capitalised word (e.g. word at the start of a sentence) differently from the same word but lower case, appearing later in the sentence. As the two words stem from the same word, the designer should make sure that the model treats them equally. Finally, the removal of non-ASCII Unicode characters is important for removing symbols such as the '@' symbol and hashtags, which are very common in tweets but do not contain useful information for the machine learning models explored later on. In academic terms, removing unnecessary characters is known as noise removal. Critical punctuation and special characters are important for human understanding of texts, but they could be detrimental for classification algorithms [3].

The next step involved the removal of all Emojis from tweets. Although there are studies that investigated the use of Emoji characters in sentiment analysis, the author concluded that developing a framework to integrate the analysis of Emojis in classifying fake and real tweets was too complex and may not be suitable in this specific task. Sentiment Analysis utilises NLP to extract opinions from users' text and classify them into positive, neutral and negative classes. These extracted opinions could be then used by firms to help track user satisfaction levels when using products and services, or even predict upcoming events [5].

At first glance, one might immediately believe that integrating sentiment analysis with the use of Emojis is a good idea. However, it is important to consider the specific application at hand. The author decided that it was too complex and thus too time-consuming to study if a particular sentiment may automatically cause a tweet to more likely be real or fake. For example, would a tweet about Hurricane Sandy filled with Emojis associated with negative emotions innately mean that this tweet is more likely to be real? This degree of complexity is still under research and is highlighted in a recent paper by M. Shiha and S. Ayvaz, which states that the usage of Emoji characters in sentiment analysis appeared to have higher impact on overall sentiments of the positive opinions compared to the negative opinions [5].

Other pre-processing methods, including stemming with the use of the 'nltk' module, translation of foreign languages into English using the 'Googletrans' library, were tried and tested. However, they were removed from the final implementation stage because the F1 scores obtained with these additional pre-processing operations worsened.

In the case of stemming, the functions 'PorterStemmer' and 'word\_tokenizer' were used. This type of NLP technique has been commonly used by researchers when dealing with tasks such as text mining, text classification, and information retrieval. The purpose is to extract the root of words by removing affixes and suffixes without affecting the meaning. To obtain the root of words in English, there are several stemming algorithms such as Lovins, Dawson, and Porter [6]. As an example, the Porter stemming algorithm which was trialled to obtain a F1 score, would classify 'consisted', 'consistency', 'consistent' and 'consistently' as 'consist'. This example highlights the purpose of stemming as it converts several slightly different forms of a specific word into the same stem without losing the word's meaning. Although this example would suggest that including stemming in the pre-processing stage would likely increase the accuracy of the machine learning model, this is often not the case. In a recent tweet classification study conducted by A. Hidayatullah et al., it concluded that it is observed that all accuracy results in tweet classification tend to decrease [6]. Stemming task does not raise the accuracy either using SVM or Naive Bayes algorithm. The limitation of these findings for this project is that the study was researching the Indonesian language.

However, there are other studies that have similar findings. In an empirical text classification study of English literature text such as early American novels, B. Yu concluded that there was negligible effect

of stemming on classification performance [7]. In addition, T. Gaustad et al. found that the use of stemming (more specifically, they used Porter's algorithm) for classification of Dutch email texts does not consistently improve classification accuracy [8]. Clearly, there are certain scenarios that stemming may be useful but the studies mentioned align with the weaker F1 scores when stemming was attempted.

Other data cleaning methods that are mentioned above, were also discarded in the final implementation process due to worse F1 scores.

### 3. Implementation

Most text classification problems involve four phases – feature extraction, dimension reduction, classifier selection and evaluation [3]. Therefore, the author follows this convention. Note that the dimension reduction step is an optional part of the pipeline which could be part of the classification system. However, in some cases this dimension reduction step could be crucial to optimise computation time. As this project involves a fixed data set and does not involve handling data and displaying results in a live scenario (e.g. giving live updates to a client via a website interface in real time), the computation time factor is treated as negligible.

With regards to the feature extraction step, the Term Frequency-Inverse Document Frequency (TF-IDF) method, which consists of converting feature space to another dimension is an important step for this project. Other than TF-IDF, common techniques of feature extractions include Term Frequency (TF), Word2Vec, and Global Vectors for Word Representation (GloVe).

#### 3.1 Consideration of feature extraction methods

Term Frequency:

The Term Frequency (TF) method is the most basic feature extraction technique under the umbrella of the 'weight word' domain. This method involves mapping each word in a tweet to a number corresponding to the number of occurrences of that word in the entire data set of tweets. The limitation for this method is that specific words that are commonly used could dominate the representations [3].

TF-IDF:

In 1972, K. Sparck Jones proposed Inverse Document Frequency (IDF) as a method to be used in conjunction with term frequency to lessen the effect of implicitly common words in the corpus [9]. IDF works by assigning a higher weight to words with either high or low frequencies term in the text document. The combination of TF and IDF is well known technique called Term Frequency-Inverse document frequency (TF-IDF).

#### 3.2 Consideration of different machine learning models

Consideration of Multinomial Naive Bayes model:

Naive Bayes is a learning algorithm that is regularly employed to tackle text classification problems [4]. It is computationally very efficient and easy to implement. There are two event models that are commonly used: the multivariate Bernoulli event model and the multinomial naive Bayes (MNB) model. The MNB model typically outperforms the Bernoulli model, and is even competitive with more specialised event models.

In the study by A. Kibriya et al., the empirical results for several versions of the MNB classifiers were evaluated [4]. More complex variants of the MNB approach (in this case: Transformed Weight-normalised Complement Naive Bayes classifier was studied) were compared the standard MNB model. The study concluded that the added complexity of the standard MNB model may not be necessary to achieve optimum performance. Therefore, the standard MNB was chosen to be part of the model comparison process in the tweet classification implementation process. Moreover, the study highlighted the importance of TF-IDF conversion and document length normalisation. A TF-IDF transformation

method was added to the pipeline when training the model in the implementation stage as the paper underlined that standard MNB model can be improved significantly by applying a TF-IDF transformation to the word features.

However, the study added that MNB is still typically inferior to the specialised support vector machine (SVM) classifiers with regards to the accuracy of classification on text categorisation problems. The paper concluded that SVMs is the recommended model choice if the longer training duration is acceptable. Therefore, a SVM classifier is included in the model comparison process.

Consideration of the Support Vector Machine model:

In Z. Liu et al.'s study on 'SVM compared with the other text classification methods' summarised the main concepts of SVM neatly [10]. The SVM model is based on structural risk minimisation theory to find the optimal separating hyperplane from the feature space so it can achieve global optimisation.

Firstly, the original data set is compressed to the support vector set, where new knowledge is gained by learning to use the subset. However, in case of a the linearly inseparable scenario, a nonlinearity mapping is applied. This mapping enables the data set to be linearly separable because the inseparable data is converted from a low-dimensional feature space to high-dimensional one.

Consideration of SVM with SGD optimisation function model (referred to as 'the SGD' model for conciseness):

Gradient Descent (GD) is a type of minimisation algorithm. The gradient descent process begins with an initial set of parameters and iterates towards a set of parameters that find minimal point for the function. This search for the minima uses calculus derivations. A potential issue with using a gradient descent model is the long duration of computation, especially with large data sets. This is due to the requirement of the model to predict each instance in a training set per iteration of a gradient descent.

The Stochastic Gradient Descent (SGD) model utilises the same updating procedure for the coefficient as GD, but differs with regards to the updating process which only runs during the training process [11]. This means that unlike GD which consists of summing up the cost over all training patterns, SGD only involves the calculation for one training pattern. As a result, the SGD model requires less computation time.

In a recent study which has a very similar context to this tweet classification problem as it consisted of developing a hoax detection system on Indonesian news sites based on text classification, A. Prasetyo et al., made several useful conclusions [11]. The researchers noted that the SGD model performed better than the Linear SVM model. In addition, they highlighted that the SGD model used in conjunction with TF-IDF lead to an increase in accuracy of classification. Therefore, this combination was implemented in this project as shown in results section.

## 4. Results and Evaluation

Three classification models were implemented for comparison before selecting the best model. The corresponding confusion matrices are shown below. In addition, the Micro F1-score are shown in Table 3. While the confusion matrices with the respective classification reports are shown Table 4. To generate these results, the functions 'plot\_confusion\_matrix', 'f1\_score' and 'classification\_report' were used. These functions were all from the 'sklearn.metrics' library.

Model	Micro F1 score (4 s.f.)
MNB	0.8662
SVM	0.8141
SGD	0.7183

Table 3.

Confusion matrices with Classification reports for comparison of three models:

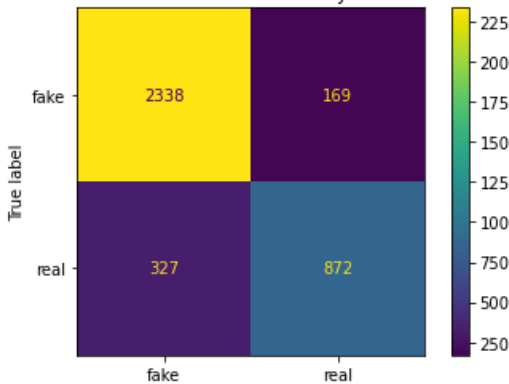
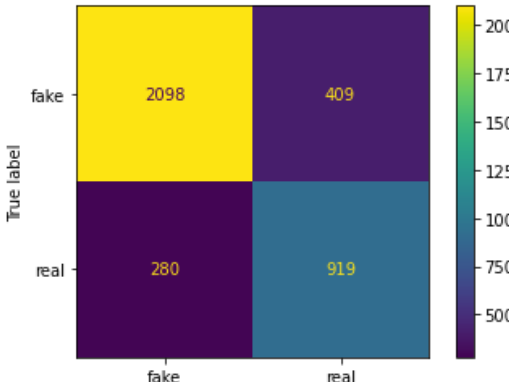
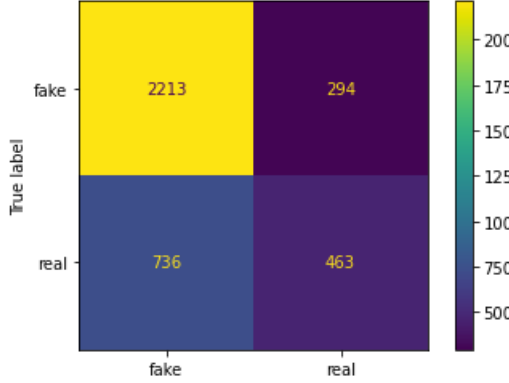
<p>Multinomial Naive Bayes</p>  <p>True label</p> <p>Predicted label</p>		<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>fake</td><td>0.88</td><td>0.93</td><td>0.90</td><td>2507</td></tr><tr><td>real</td><td>0.84</td><td>0.73</td><td>0.78</td><td>1199</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.87</td><td>3706</td></tr><tr><td>macro avg</td><td>0.86</td><td>0.83</td><td>0.84</td><td>3706</td></tr><tr><td>weighted avg</td><td>0.86</td><td>0.87</td><td>0.86</td><td>3706</td></tr></table> <p>Computation duration: 0.2056 seconds</p>		precision	recall	f1-score	support	fake	0.88	0.93	0.90	2507	real	0.84	0.73	0.78	1199	accuracy			0.87	3706	macro avg	0.86	0.83	0.84	3706	weighted avg	0.86	0.87	0.86	3706
	precision	recall	f1-score	support																												
fake	0.88	0.93	0.90	2507																												
real	0.84	0.73	0.78	1199																												
accuracy			0.87	3706																												
macro avg	0.86	0.83	0.84	3706																												
weighted avg	0.86	0.87	0.86	3706																												
<p>SVM Model</p>  <p>True label</p> <p>Predicted label</p>		<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>fake</td><td>0.88</td><td>0.84</td><td>0.86</td><td>2507</td></tr><tr><td>real</td><td>0.69</td><td>0.77</td><td>0.73</td><td>1199</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>3706</td></tr><tr><td>macro avg</td><td>0.79</td><td>0.80</td><td>0.79</td><td>3706</td></tr><tr><td>weighted avg</td><td>0.82</td><td>0.81</td><td>0.82</td><td>3706</td></tr></table> <p>Computation duration: 0.2100 seconds</p>		precision	recall	f1-score	support	fake	0.88	0.84	0.86	2507	real	0.69	0.77	0.73	1199	accuracy			0.81	3706	macro avg	0.79	0.80	0.79	3706	weighted avg	0.82	0.81	0.82	3706
	precision	recall	f1-score	support																												
fake	0.88	0.84	0.86	2507																												
real	0.69	0.77	0.73	1199																												
accuracy			0.81	3706																												
macro avg	0.79	0.80	0.79	3706																												
weighted avg	0.82	0.81	0.82	3706																												
<p>SGD Model</p>  <p>True label</p> <p>Predicted label</p>		<table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>fake</td><td>0.75</td><td>0.88</td><td>0.81</td><td>2507</td></tr><tr><td>real</td><td>0.61</td><td>0.39</td><td>0.47</td><td>1199</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.72</td><td>3706</td></tr><tr><td>macro avg</td><td>0.68</td><td>0.63</td><td>0.64</td><td>3706</td></tr><tr><td>weighted avg</td><td>0.71</td><td>0.72</td><td>0.70</td><td>3706</td></tr></table> <p>Computation duration: 0.2700 seconds</p>		precision	recall	f1-score	support	fake	0.75	0.88	0.81	2507	real	0.61	0.39	0.47	1199	accuracy			0.72	3706	macro avg	0.68	0.63	0.64	3706	weighted avg	0.71	0.72	0.70	3706
	precision	recall	f1-score	support																												
fake	0.75	0.88	0.81	2507																												
real	0.61	0.39	0.47	1199																												
accuracy			0.72	3706																												
macro avg	0.68	0.63	0.64	3706																												
weighted avg	0.71	0.72	0.70	3706																												

Table 4.

The confusion matrix is a neat graphical representation of True-Positive (TP), False-Positive (FP), True-Negative (TN) and False-Negative (FN). The top left quadrant represents the TP, which is the number of tweets that is classified correctly as fake. This is the most important quadrant as the goal of the project is to identify fake tweets with high accuracy. The bottom left quadrant shows the FP, which is when there is an incorrect prediction of tweets being fake when they are in fact, real. Moreover, the top right quadrant shows the FN, which is when there is an incorrect prediction of tweets being real when they are actually fake. Lastly, the bottom right quadrant represents the TN, which is when there is a correct prediction of real tweets.

As shown above, out of the three models implemented, the MNB model performed the best with a micro F1-score of 0.8662. The SVM model was a close second with a micro F1-score of 0.8141. While the



SGD model performed significantly worse, with a micro F1-score of 0.7183. The potential reasoning behind these results is discussed below.

In a paper which compared three different classification models using the confusion matrix for sentiment analysis on newspaper headlines, C. Rameshbhai et al. summarised that the method combining the use of TF-IDF and Linear SVM provides better accuracy for smaller datasets [12]. Conversely, for larger datasets, SGD and linear SVM model is likely to outperform other models. This suggests that due to the data set of tweets being small, the MNB model in this specific application is competitive with the SGD and SVM models. Perhaps, if a significantly larger data set of tweets were used, the SVM and SGD model would outshine the MNB model.

In a slightly older (2006) paper, F. Colas posed an important question: following the rising interest towards the Support Vector Machine alongside with various studies showing that SVM outperforms other classification algorithms, should researchers simply always opt for SVM [13]? This study's finding was that all the classifiers (SVM, k-Nearest-Neighbour and MNB) achieved comparable performance on most problems. The highlighted result was that SVM was not a clear winner, despite its good overall performance. The MNB model having competitive accuracy results was a noteworthy point as well. This study's finding aligns with the results of the tweet classification as the MNB model is shown to be competitive with more specialised models.

In terms of computation duration, MNB had an advantage when the training data size is small, but as the number of text documents increases, the difference in computation duration decreases. On the other hand, SVM has a significantly longer computation duration. Importantly, this duration increases quadratically with the number of documents in the training set.

As the SDG model scored significantly worse than the other two models, hyperparameter tuning was attempted on this model to see if it could be tuned to achieve a comparable performance. For reference on what processing parameters would likely be helpful, the paper by S. Diab, about optimising SGD in text classification in the context of predicting global terrorist attacks was used [14].

The following data processing parameters were considered:

- 'n\_gram\_range': determine the range of sequence of n items of the text – unigram which means a one-word sequence or bigram which means a two-word sequence.
- 'norm': two normalisation methods investigated - the least absolute deviations and the least square method.
- 'use\_idf': to enable or disable IDF when scoring the count features.
- 'alpha': a constant used to compute the step size and the learning rate.

The best micro F1-score of 0.82 was derived from the following tuning the SGD model with the use of the Grid Search method: using unigram as the 'n\_gram\_range'; using the least absolute deviations method for normalisation; disabling IDF for scoring count features and finally, using an alpha of 0.00001. Although the micro F1-score for the tuned SGD model remains to be lower than the MNB model, it shows that tuning the SGD model could improve its classification accuracy significantly as the one without tuning scored 0.7183. This finding aligns with S.Diab's conclusion that utilising a grid search approach to obtain the hyperparameter would increase the accuracy of the classifiers. Lastly, to confirm that the hyperparameter tuning process does not occupy excessive amount of time, the processing time was found to be 20.6 seconds.

## Conclusion

Due to the MNB model having the highest micro-F1 score, this model is the final choice for this project's application. One main finding that was not intuitive to the author, involves obtaining poorer classification performance as more steps of pre-processing was done. At the start of the project, the author's intuitive thought was that by adding popular cleaning processes such as stemming, removing stop words, language translation, this would automatically lead to producing models that achieve higher



accuracy. This was not the case and perhaps ending the project with this summary is suitable to highlight the main learning points. The conclusion from Z. Jianqiang et al., summarised that removing stop words, numbers, and URLs might be suitable to reduce noise, but could have minimal effect on the performance of classifiers in some applications [15]. If the author would conduct future research on this topic, more types of machine learning algorithms such as Bidirectional Encoder Representations from Transformers (i.e. BERT) would be experimented with for a wider breadth of comparison. Lastly, more focus would be placed on hyperparameter tuning on several different models as it could significantly improve the classification accuracy of the models.

## References

- [1] O'Connor, C. and Murphy, M., 'Going viral: doctors must tackle fake news in the covid-19 pandemic'. *BMJ*, 369(10.1136).
- [2] DiFranzo, D. and Gloria-Garcia, K., 'Filter bubbles and fake news', *Crossroads The ACM Magazine for Students, Association for Computing Machinery*, April 2017.
- [3] Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, and Brown, "Text Classification Algorithms: A Survey," *Information*, vol. 10, no. 4, p. 150, Apr. 2019.
- [4] Kibriya A.M., Frank E., Pfahringer B., Holmes G. 'Multinomial Naive Bayes for Text Categorization Revisited'. In: Webb G.I., Yu X. (eds) *AI 2004: Advances in Artificial Intelligence*. AI 2004. Lecture Notes in Computer Science, vol 3339. Springer, Berlin, Heidelberg.
- [5] Shiha, M. and Ayvaz, S., 2017. The effects of emoji in sentiment analysis. *Int. J. Comput. Electr. Eng.(IJCEE)*, 9(1), pp.360-369.
- [6] Hidayatullah, A.F., Ratnasari, C.I. and Wisnugroho, S., 2016. Analysis of Stemming Influence on Indonesian Tweet Classification. *Telkomnika*, 14(2), p.665.
- [7] Yu, B., 2008. An evaluation of text classification methods for literary study. *Literary and Linguistic Computing*, 23(3), pp.327-343.
- [8] Gaustad, T. and Bouma, G., 2002. Accurate stemming of Dutch for text classification. In *Computational Linguistics in the Netherlands 2001* (pp. 104-117). Brill Rodopi.
- [9] Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* 1972, 28, 11–21.
- [10] Liu, Z., Lv, X., Liu, K. and Shi, S., 2010, March. Study on SVM compared with the other text classification methods. In *2010 Second international workshop on education technology and computer science* (Vol. 1, pp. 219-222). IEEE.
- [11] A. B. Prasetijo, R. R. Isnanto, D. Eridani, Y. A. A. Soetrisno, M. Arfan and A. Sofwan, "Hoax detection system on Indonesian news sites based on text classification using SVM and SGD," 2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), 2017, pp. 45-49, doi: 10.1109/ICITACEE.2017.8257673.
- [12] Rameshbhai, C.J. and Paulose, J., 2019. Opinion mining on newspaper headlines using SVM and NLP. *International Journal of Electrical and Computer Engineering (IJECE)*, 9(3), pp.2152-2163.
- [13] Colas, F. and Brazdil, P., 2006, August. Comparison of SVM and some older classification algorithms in text classification tasks. In *IFIP International Conference on Artificial Intelligence in Theory and Practice* (pp. 169-178). Springer, Boston, MA.
- [14] Diab, S., 2019. 'Optimizing stochastic gradient descent in text classification based on fine-tuning hyperparameters approach. A case study on automatic classification of global terrorist attacks.' *arXiv preprint arXiv:1902.06542* (2019).
- [15] Jianqiang, Z. and Xiaolin, G., 2017. Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access*, 5, pp.2870-2879.