# COMP3222 Coursework - MediaEval 2015

Anthi Terezaki
at14g20@soton.ac.uk
(Student ID:32256434)

January 13, 2023

## 1 Introduction and Data Analysis

### 1.1 Introduction

With the rise of social media platforms such as Twitter as a primary source for the transmission of news, fake news has become a prevalent issue in today's society. Using the MediaEval 2015 dataset of tweets, this research intends to investigate methods for automatically classifying Twitter news-related information as fake or real. The described issue is identifying fake posts, which are reposts of legitimate multimedia with false associations, digitally altered data, or synthetic imagery presented as real. The machine learning algorithms used in this problem, however, will focus on natural language processing of textual Twitter data rather than image processing. The data in the dataset will be analysed in this report to acquire insight into the characteristics of false news and the difficulties of recognising it. We will compare five different machine learning algorithms, iterate over them with different preprocessing methods, and determine which algorithm performs the best.

### 1.2 Data Characterisation and Analysis

The dataset consists of tweets related to news content, with each tweet containing a `tweetId`, the text included in the tweet (`tweetText`), a `userId`, an `imageId(s)` which was changed to `imageid` so the Python code didn't clash and get confused with the labelling, a `timestamp`, and a `label` indicating whether the tweet is real, fake or humor. For simplicity purposes, the 'fake' and 'humor' labels where combined into one 'fake' label. The dataset does not include any information about the images themselves, and the machine learning algorithms used will focus on natural language processing of the textual tweet data. The dataset provided includes an already split training and test dataset whose format in a text file (.txt) with each row on a new line and columns separated by a tab character.

The training set contains 14,277 tweets and the testing set has 3,755 tweets, so a total of 18,032 Twitter posts. Posts from events such as Hurricane Sandy, the missing Malaysia Airlines aircraft MH370, and the Boston Marathon bombing are included in the dataset (Figure 1). Figure 1 reveals that there is a large bias towards posts about hurricane Sandy in the training data, with a total of 12,318 posts about it, which indicates that recurring phrases in these posts, such as 'hurricane' and 'Sandy,' would be learned as features that would not apply to any other occurrences.

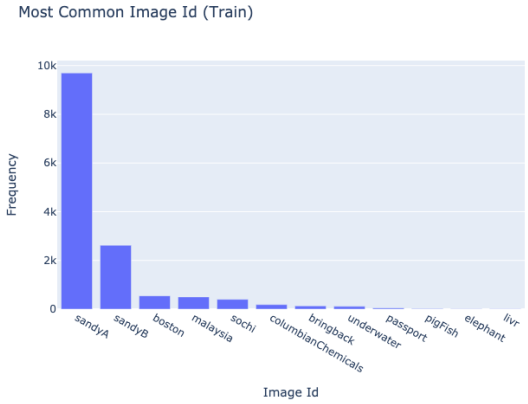



Figure 1: Number of posts per imageId in training set

In the test set, there is a considerably higher coverage of the Syrian boy and the earthquake in Nepal (Figure 2) than any other event. Also, there is no mention of hurricane Sandy, so our training cannot be based on the events of the tweets.

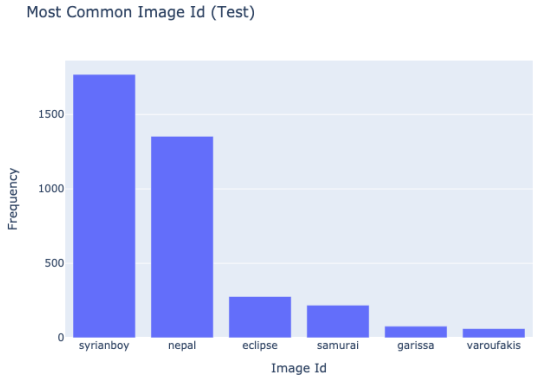
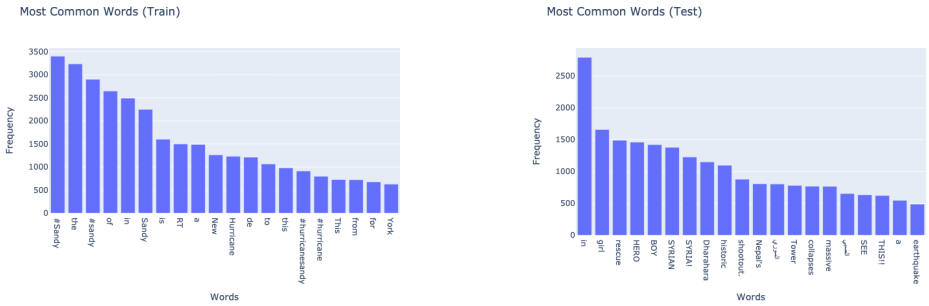



Figure 2: Number of posts per imageId in test set

In addition, in Figure 3, we can see that the most common words in both the training and in the test set, include mix of languages, stopwords taking the first place in most common word in the test set, used 2791 times, and other noise such as special

symbols, especially #. Also, one common word in the training set that's not found in the common words in the test set in 'RT' which means retweet. So, this can give us an idea on what data may require cleaning from so we can get the best possible accuracy and F1 score.




(a) Training Set (b) Test Set

Figure 3: Most common words in tweetText

Finally, the majority of posts in this dataset are in English, but there are also posts in other languages. English and Spanish combined, make up for 86% of all training set data. However, the accuracy of these numbers may be affected by the presence of slang, misspellings, and emojis in the tweets, which can make it difficult for language detection tools like Google's langdetect to accurately identify the language. Additionally, some tweets contain random characters which further add to the noise in the data. This is demonstrated by the fact that langdetect was unable to identify the language of tweets full of emojis and misspelled words and wouldn't run properly without a try/except block.

# 2 Algorithm Design

## 2.1 Pre-processing

The initial step in preprocessing was to convert the 'humor' labels to 'fake', as the aim was to create a binary classifier that labels posts as either 'fake' or 'real'. To experiment with different data, a copy of the original DataFrame, `df_train`, was kept as `df_keep`, ensuring no entries were manually removed. To address the heavy bias in the dataset towards certain events, caused by numerous retweets of the same posts and the common use of the word 'RT', a separate DataFrame, `df_train_no_rt`, was created to remove these repetitive posts. This helped to avoid overfitting the model to specific features that may have originated from Hurricane Sandy posts and allowed the algorithm to be more flexible for tweets related to any event. It was later discovered that removing retweets had a positive impact on the performance of the

classifiers.

An attempt to deal with foreign languages in the training set using Google Translator was done, but ultimately translated text was not used due to time constraints and also preliminary testing revealed that the results were similar, and at some case worse to the `df_train` that had undergone the same preprocessing.

Initially, emojis, '\n', '&', username mentions(@) and links(http://) were removed from the tweet text as they were found to not contribute much value to the classification algorithm, and whitespace was corrected before proceeding. However, after testing, I decided to only remove symbols and emojis, and replace the '#' symbol with a whitespace. This was done because many tweets had a lot of hashtags next to each other, as shown in Figure 3, which caused many useful words to connect with each other, and then corrected any double spaces that might have occurred. Next, I removed stopwords from all tweets by using stopwords from all languages as seen in the data analysis, as they were heavily used but did not bring any value to our training process.

Finally, an additional column was added to each DataFrame where the refined text would be tokenised and lemmatised. Tokenisation is the process of breaking up a larger body of text into smaller units called tokens and in NLP, tokens are usually words or phrases. Lemmatisation, on the other hand, is the process of reducing a word to its base or root form. For example, the lemma of the word "running" would be "run." The data within the refined_text column was then tokenised and lemmatised and directed to the tl_text column. Spelling correction and POS and NER tagging were not attempted as they were deemed unnecessary for the algorithm and might have had a negative impact on performance.

## 2.2 Feature Extraction

In this text classification problem, various combinations of preprocessing steps, training models, and feature processing methods were evaluated to find the best approach for vectorisation and training. Three different vectorisation techniques which are commonly used in NLP tasks were used for feature extraction: the Bag-of-Words method, N-Grams vectorisation, and TF-IDF vectorisation.

The Bag-of-Words method counts the occurrence of words in a given text, but it forms a collection of unigrams, disregarding any word order dependence. This means that the context in which a word is used is not taken into account, only the frequency of the word. N-Grams vectorisation counts the occurrences of pairs of consecutive words. This technique allows to take into account the context in which the words are used, however it was found that analysing hashtags as standalone features was more effective than considering them pairwise. TF-IDF vectorisation combines CountVectoriser (Bag-of-Words) with TF-IDF transformer. The TF-IDF transformer calculates the term frequency (TF) and the inverse document frequency (IDF) of each word, which

is used to scale down the impact of words that occur very frequently. This allows to emphasize less frequent words which are often more informative. The TF-IDF score of a word in a document is the product of its term frequency and inverse document frequency. With this method, the best tested max df argument value was 0.3, which ignores words that have a frequency strictly higher than the given value.

### 2.3 Training Models

The performance of two popular Naive Bayes models was evaluated in this testing: the Multinomial Naive Bayes(MultinomialNB) and the Bernoulli Naive Bayes(BernoulliNB). The Multinomial model is designed to work with discrete data such as integer word counts, while the Bernoulli model is designed to work with binary or boolean features such as the presence or absence of certain words in a document. The Multinomial Naive Bayes is often used in text classification problems due to its efficiency and good performance. However, the Bernoulli model can still perform well even with small vocabulary sizes. Comparing both models, the Multinomial Naive Bayes usually performs better at larger vocabulary sizes and provides a significant decrease in error over the Bernoulli model at any vocabulary size.

In addition, a Stochastic Gradient Descent classifier, a Ridge classifier, and a Linear Support Vector Classifier (LinearSVC) were also tested. The Stochastic Gradient Descent classifier is a linear classifier that uses the stochastic gradient descent algorithm to optimise the parameters of the model, which makes it efficient and suitable for large datasets and high-dimensional feature spaces. The Ridge classifier is a linear model that uses Ridge Regression to classify data, which adds a regularisation term to the cost function to help prevent overfitting, and it can provide better generalization performance than a standard linear classifier when the data is noisy or has multicollinearity. LinearSVC is a type of Support Vector Machine (SVM) that is commonly used in NLP tasks and it is known for its ability to find the best boundary or hyperplane that separates the different classes in high-dimensional feature spaces and it provides global guarantees of optimality.

## 3 Evaluation

The algorithms were evaluate using a confusion matrix and the F1 score derived from it. A correct identification of a fake tweet is considered a True Positive(TP), while a real tweet incorrectly identified as fake is a False Positive(FP). A correct identification of a real tweet is a True Negative(TN) and a fake tweet identified as real is a False Negative(FN). The F1 score was calculated using the formula:

$$F1 = (2 * precision * recall)/(precision + recall),$$

where precision and recall are equal to:

$$precision = TP/(TP + FP)$$

$$recall = TP/(TP + FN)$$

Five classifiers (Multinomial Naive Bayes, Bernoulli Naive Bayes, Stochastic Gradient Descent Classification, Ridge Classification and Linear Support Vector Classification) were evaluated using three feature extraction methods (Bag-of-Words, N-Grams, and TF-IDF) with 3 different ways of preprocessing.

## 3.1 Iteration 1

The first iteration employed many normalisation techniques, including lowercasing, punctuation removal, elimination of commonly occurring symbols, and URL elimination. Following this, all stopwords were removed and the feature extraction and classification process began.

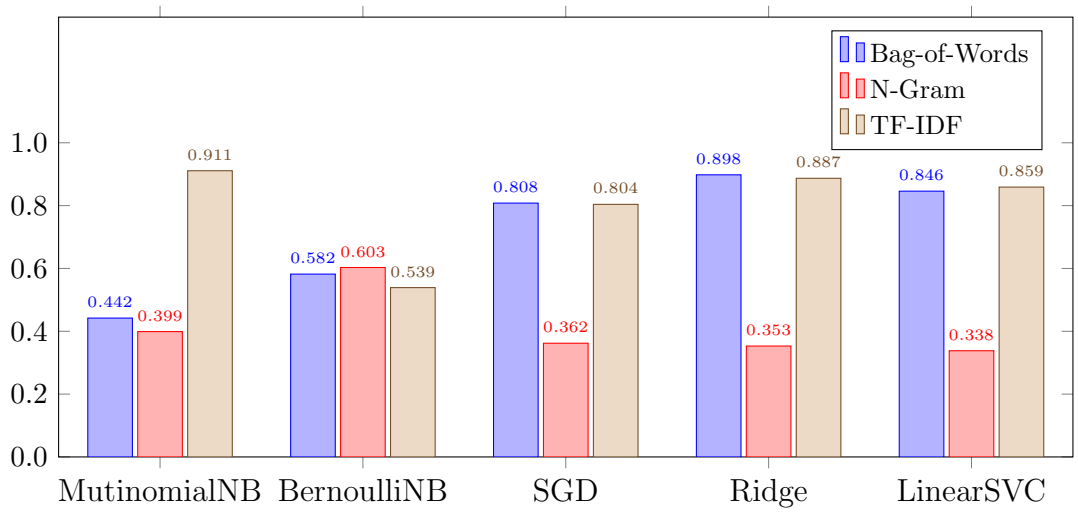



Figure 4: Classifier F1 scores with first iteration

When using Bag-of-Words as a feature extraction approach, the Naive Bayes models performed poorly, whereas SGD, Ridge, and LinearSVC achieved high F1 scores. As the results of this iteration show, N-Grams with n = 2 performed quite poorly when compared to the prior iteration. Bernoulli Naive Bayes was the only classifier to have a rise in F1 score, with 0.603. As a result, that technique was not worth pursuing any further and will be dropped after the next two iterations, however the results will be displayed. When TF-IDF was used as a feature extraction method, all

classification algorithms performed well, with MultinomialNB having the highest F1 score with 0.911. All non-Naive Bayes models performed almost the same way with Bag-of-Words and TF-IDF.

## 3.2 Iteration 2

The second iteration used again the same normalisation techniques and removal of stopwords was applied, but retweets where removed. Following this, tokenisation and lemmatasation where applied to the tweets.

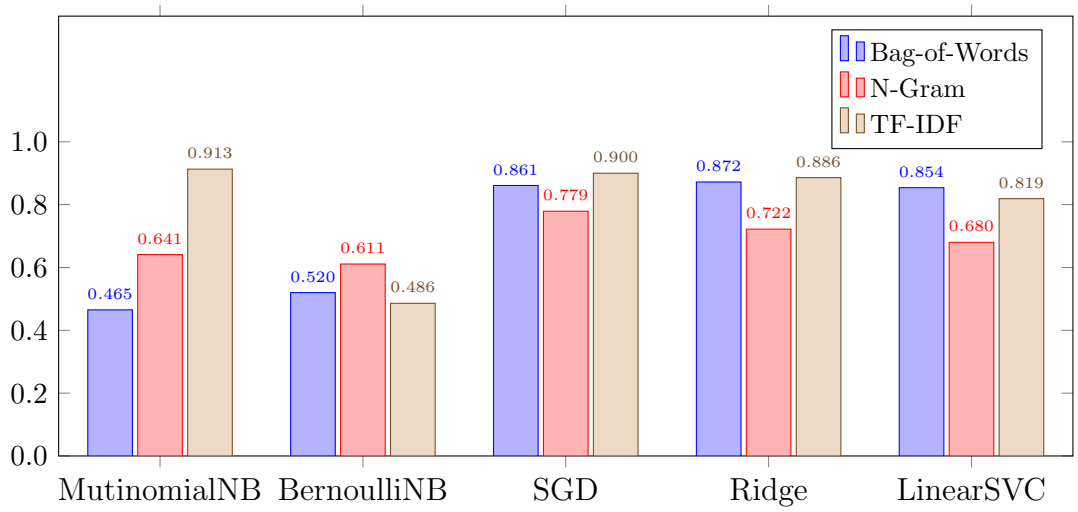



Figure 5: Classifier F1 scores with second iteration

Bag-of-Words and TF-IDF give us almost the same results as the first iteration with some increases and decreases here and there, but again the highest F1 score is found again when using MultinomialNB with TF-IDF with a slight increase in score, achieving 0.913. Surprisingly, all N-Gram scores increased a lot, but still they cannot compare to the other feature extraction methods.

## 3.3 Iteration 3

In my final iteration, I decided to have as normalisation methods the replacement of hashtags (#) with whitespaces, and removing all double whitespaces later, the overall removal of any symbols and number, and the removal of these specific mentions & and \\ n.

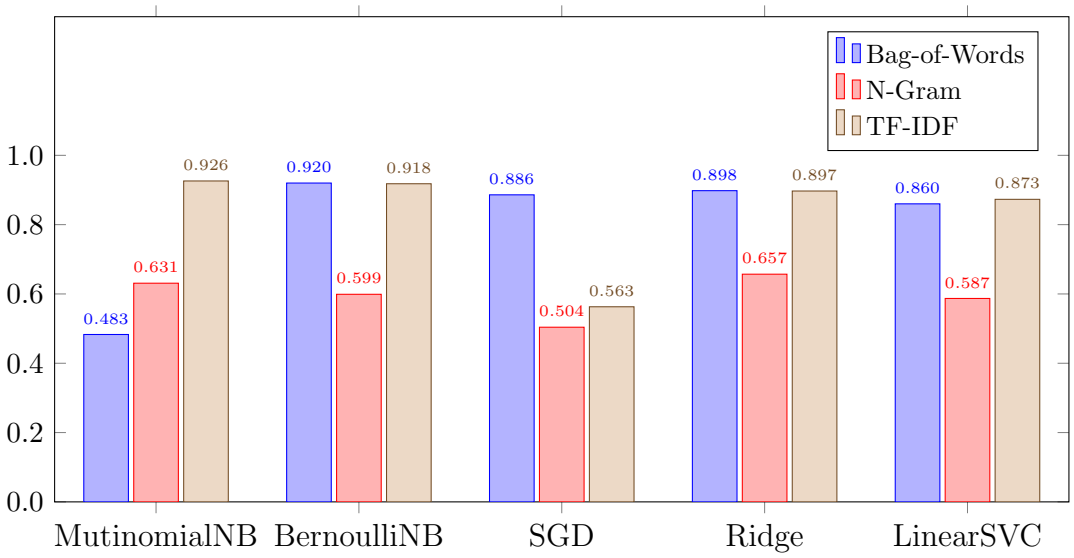
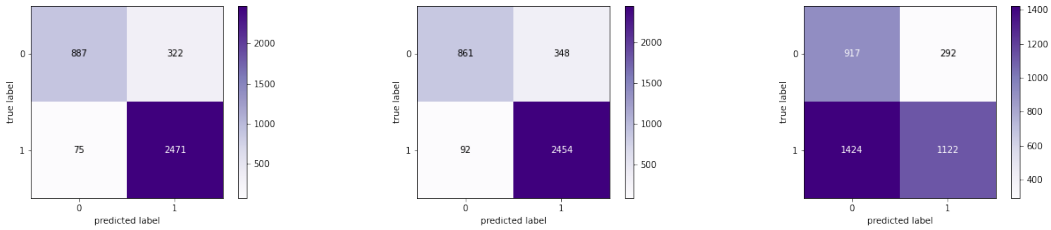



Figure 6: Classifier F1 scores with third iteration

Bag-of-words surprisingly, gives us an amazing F1 score in BernoulliNB of 0.920 which is the best we have seen so far. But again, the highest F1 score is found when using MultinomialNB with TF-IDF with 0.926. Note that the rest of scores remained similar as before, with the exception of SGD, where we can see a significant decrease in F1 score when using TF-IDF, scoring only 0.563, compared to 0.900 and 0.804 in the previous two iterations.

BernoulliNB was drastically affected by the inclusion of http links and replacement of # with whitespace as while in the first two iterations it was the worst algorithm, in the third iteration, both with Bag-of-Words and TF-IDF, it got surprisingly good results, surpassing all the other F1 scores in all iterations, except MultinomialNB with TF-IDF. Multinomial Naive Bayes was the only classifier of the five that was so drastically affected by introducing TF-IDF transformation in all three iterations. Finally, All non-Naive Bayes models performed almost the same way throughout all three iterations, with the exception of the SGD that was talked about previously.




(a) MultinomialNB classifier (b) BernoulliNB classifier (c) SGD classifier

Figure 7: Confusion matrices with TF-IDF vectorisation

# 4 Conclusion

The task at hand was to identify fake posts, and an F1 score of 0.926 was achieved by using a Multinomial Naive Bayes classifier with a Term Frequency-Inverse Document Frequency feature vectorization. To our surprise, we also achieved an F1 score of 0.92 using a Bernoulli Naive Bayes classifier. We had expected Linear Support Vector Classification and Stochastic Gradient Descent to have higher F1 scores, but the results were still satisfactory. We attribute the better performance of the Naive Bayes models to the unconventional way of normalization by keeping the links of the tweets.

Twitter is widely used by people from various backgrounds and this leads to the use of many abbreviations, poor spelling, and grammar in tweets. This made the process of testing more challenging, and it was more of a trial and error problem to get to a good F1 score. Translation to this kind of text is also difficult, as it would require a lot of computational power and even then, inconsistencies would still appear.

Further analysis of the algorithm's performance showed that the two best-performing models overpredict posts as fake. Since classifying them as fake is the more important task, this is a better outcome than overpredicting them as real, which could lead to news professionals taking fake posts into account as real ones. In conclusion, our research shows that Multinomial and Bernoulli Naive Bayes classifiers are effective in identifying fake posts on Twitter, and the results achieved are satisfactory.

# References


[1] Christina Boididou, Symeon Papadopoulos, Yiannis Kompatsiaris, Steve Schifferes, and Nic Newman. Challenges of computational verification in social multimedia. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 743–748, 2014.

[2] Jose Camacho-Collados and Mohammad Taher Pilehvar. On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. *arXiv preprint arXiv:1707.01780*, 2017.

[3] Jingnian Chen, Houkuan Huang, Shengfeng Tian, and Youli Qu. Feature selection for text classification with naïve bayes. *Expert Systems with Applications*, 36(3):5432–5435, 2009.

[4] Bijoyan Das and Sarit Chakraborty. An improved text sentiment classification model using tf-idf and next word negation. *arXiv preprint arXiv:1806.06407*, 2018.

[5] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[6] Milan Straka and Jana Straková. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 shared task: Multilingual Parsing from raw text to universal dependencies*, pages 88–99, 2017.

[7] Xinyi Zhou and Reza Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys (CSUR)*, 53(5):1–40, 2020.