Tackling the infodemic: COVID-19 Fake news detection using natural language processing and Naïve Bayes Algorithm

Armela Ligori

Computer Engineering Department

Epoka University, Tirana, Albania

ABSTRACT

Amid the Covid-19 pandemic, when reliable information is vital for public health and safety, COVID-19 related fake news has been spreading even faster than the facts. Considering the negative consequences these news present on various domains, the scientific community has put a lot of effort in tackling this problem. In this paper, I aim to give my contribution in combating the COVID-19 infodemic by training a probabilistic machine learning model for the identification of false information spreading around social media. For this reason, a dataset consisting of tweet news collected from various social media accounts is taken into consideration. This paper presents a tweet pre-processing pipeline tailored for this specific classification. To extract numerical feature vectors from categorical data, BOW and TF-IDF representations are used. The Multinomial Naïve Bayes classifier is trained and evaluated using k-fold cross validation. Then an extra testing dataset is used to predict the final results. The model results in a 91% accuracy and F1-score on the test set. This is a decent result in comparison to the baseline algorithms used for this dataset, considering the relative simplicity of the model.

Keywords: *fake news detection, COVID-19, NLP, Naïve Bayes, machine learning*

1 INTRODUCTION

With the recent situation of COVID-19 and vaccination progress, the amount of Fake News spreading around social media has increased significantly, and has generated situations that negatively impact public health in multiple regions of the world [1]. This fake information can strongly influence people's behaviour [2], causes confusion, sows division, provokes social panic, which directly impact emergency response, treatment, recovery, and mental health during these difficult times [3]. For the common good, it is essential to have a set of tools to be able to combat this infodemic. The first step towards this aim would be to develop models that automatically detect the veracity/falsity of the COVID-19 news circulating around social media.

Fake news detection is a text classification task that has been previously tackled by researchers using different machine learning-based NLP (natural language processing) techniques. However, when applied to a specific domain, such as the novel corona virus, there are a number of issues that arise that make this detection a challenging task, such as the lack of acceptable annotated datasets [4], the ever-changing information as to what we know about this virus etc. To help in tackling the detection of fake news related to COVID-19, Patwa et al. in [4] have described and released a manually annotated dataset of 10,700 fake and real social media news related to COVID-19. Furthermore, they benchmarked this dataset with four ML baselines: Decision Tree, Logistic Regression, SVM and Gradient Boost. The dataset is made available as part of a shared task at CONSTRAINT-2021 workshop. Based on this dataset, different analysing methods are used to study the classification of corona virus related news. On his paper, Felber [5] analysed the performance of some classical ML models by employing also several linguistic features such as n-gram, readability, emotional tone and punctuation. In contrast to classical ML algorithms, Wani et al. [6] took a deep learning approach and developed classification models based on Convolutional [7]Neural Networks (CNN), Long Short

Term Memory (LSTM), and Bidirectional Encoder Representations from Transformers (BERT). Meanwhile, Raha et al. [7] compared using simple baseline models like: Logistic regression, Random Forests, XGBoost and SVM with using state-of-the-art transformer models like BERT, RoBERTa and ELECTRA for this classification. They arrived at the result that the latter outperform the classical machine learning models.

Taking into consideration this dataset, in this project, I aim to develop a classification model for Covid-19 news, using the simple but effective Multinomial Naïve Bayes Algorithm. My aim is to see how this classical ML model will perform in comparison to the baseline approaches (SVM, DT, Logistic Regression, Gradient Boost) and the latest more complex pre-trained language transformers presented above.

The rest of paper is designed as follows: the second section is dedicated to the literature review; the third section describes the materials used in this paper: hardware and software components. Methods, followed by results are shown on the fourth and fifth section respectively. Finally, the findings are concluded in the sixth section.

2 Literature Review

Fake news detection in general is a text classification task that has been previously tackled by researchers using different machine learning-based NLP (natural language processing) techniques. The research of Granik and Mesyura [8] showed, that even a quite simple machine learning (ML) algorithm, such as naive Bayes classifier, may show a good result on such a problem. In recent years, deep learning approaches have achieved premising results on various NLP tasks: Zhang et al. [9] proposed a deep recurrent diffusive neural network to address the problem of fake news detection. Karimi et al. [10] proposed a Multi-Source Multi-class Fake News Detection framework that can do automatic feature extraction using Convolution Neural Network (CNN). On the other hand, in contrast to the traditional RNN and CNN models, the recent pre-trained language models such as BERT (Bidirectional Encoder Representations from Transformers) have outperformed previous models and achieved state-of-the-art results [11] due to their deep-contextualizing nature [12].

Regarding the specific Covid-19 infodemic, Patwa et al. in their paper [4], have described and released a manually annotated dataset of 10,700 fake and real social media news related to COVID-19. Doing so, they aim to help researchers tackle the problem of COVID-19 Infodemia. The dataset is class-wise balanced as 52.34% of the samples consist of real news and 47.66% of the data consists of fake news. The dataset is split into train (60%), validation (20%) and test (20%) by maintaining the class-wise distribution. By doing some Exploratory Data Analysis (EDA), they noticed that there is a significant overlap of important words across fake and real news. As a feature extraction method they applied term frequency–inverse document frequency (TF-IDF) statistical method. Then, they benchmark the developed dataset by applying machine learning algorithms and project them as the potential baselines: Logistic Regression (LR), Support Vector Machine (SVM) with linear kernel, Decision Tree (DT) and Gradient Boost (GDBT). Among the ML models, they noticed that the SVM-based classifier results in a higher accuracy and F1-score of 93.46%.

Meanwhile, Felber [5] addressed this classification problem by applying classical machine learning algorithms: SVM, Random Forest, Logistic Regression, Naive Bayes and Multilayer Perceptron, together with several linguistic features, such as n-grams, readability, emotional tone and punctuation. Using these extra linguistic features, they aimed to have better results than the baseline approach of Patwa et al. [4] . Indeed, by employing linguistic features, they managed to achieve the best result of 95.19% F1-score on test data on their SVM model.

On the other hand, Wani et al. [6] evaluated the recent advancements in deep learning based text classification algorithms for the task of COVID-19 fake news detection. The techniques include raw models based on CNN and LSTM and pre-trained transformer-based models like BERT. After the normal pre-processing pipeline, instead of the common TF-IDF statistical method, word embeddings were used for feature extraction. Sequential models (CNN, LSTM) were trained using two types of word embeddings namely Glove and Fast-text. Meanwhile for the transformer-based model, their experiments involved further pre-training using an extra COVID-19 corpus and fine-tuning the transformer-based models. They compared their result with the baseline accuracy in [4] and achieved a maximum accuracy of 98.41% using language model pre-training on BERT over the baseline best accuracy of 93.46%.

Raha et al. [7] continued to see the problem as a binary classification. They implemented a few simple baseline models like: Naive Bayes, Logistic regression, Random Forests, Boosting models (XGBoost) and SVM. In contrast to the previous group of approaches that used classical ML algorithms [4] [5], Raha et al. combined and split the dataset in a slightly different way: train and validation in the ratio of 9:1. Also, for feature extraction and vector representation of the words, they considered both TF-IDF method and word embedding with Word2Vec. It was seen that the results from using TF-IDF feature extraction for the models were better than using Word2Vec. The best F1-score of 94.1% was achieved for SVM Model. This is a slight improvement in comparison to the baseline F1-score achieved by Patwa et al. in [4], but lower than the SVM Model of Felber in [5]. Then, as a second approach, they used the transformer architecture and fine-tuned different pre-trained transformer models like RoBERTa, BERT and Electra on the COVID-19 training dataset. As a result, their RoBERTa model gives a F1-score of 98.64% on the official test set, higher than the BERT model proposed in [6].

In their paper, Das et al. [13], proposed a more advanced approach for the problem of fake news detection. They developed ensemble models consisting of different combinations of pre-trained models followed by a statistical feature fusion network. The ensemble consisting of pre-trained RoBERTa, XLNet, XLM-RoBERTa and DeBERTa produced the best classification result overall on the test set. In contrast to previous approaches, Das et al. argued that by incorporating and analysing various meta-attributes present in news items or tweets, like source, username handles, URL domains and authors as statistical features, their approach could go a long way in creating a robust framework for fake news detection. They evaluated their approach on the COVID-19 Fake News dataset proposed in [4]. The Heuristic Ensemble Model, obtained a F1-score of 98.92 %, the highest discussed so far.

3 Materials

3.1 Dataset

The dataset used in this paper was manually annotated and made publicly available in [4] as part of a shared task at Constraint@AAAI2021 workshop. This dataset consists of tweets representing fake or real news related to the Covid-19 pandemic, collected from various social-media platforms such as Twitter, Facebook, Instagram, etc. For each tweet, the dataset stores information for the id of the tweet (which is irrelevant to our classification), and two categorical values: the tweet text and the class label for that tweet. As mentioned previously, this is a binary classification problem and the two class labels are: fake and real. The dataset consists of 10700 tweet samples. Since it is manually curated from the authors, there are no missing values that we have to deal with. Figure 1 shows a snippet of this dataset.

	id	tweet	label
0	1	The CDC currently reports 99031 deaths. In gen...	real
1	2	States reported 1121 deaths a small rise from ...	real
2	3	Politically Correct Woman (Almost) Uses Pandem...	fake

Figure 1 *Dataset Structure*

The dataset is originally split into train (60%), validation (20%) and test (20%) by maintaining the class-wise distribution. For this work, the split of this dataset as introduced in the original paper is maintained, however the training split and the validation split are merged together into a train-valid dataset to perform k-fold cross validation while training. In Figure 2, the class-wise distribution across training, validation and test split as provided by the authors is presented.

Figure 2 *Class-wise distribution across original splits*

3.1.1 Pre-processing steps

Text pre-processing is considered to be a significant step for natural language processing tasks. The purpose of text pre-processing is to get a 'clean' version of the text suitable for our task in hand, by removing any unnecessary or ambiguous patterns within the text. This sub-section presents the basic pre-processing pipeline that this paper follows. This pipeline is created for an individual tweet and then applied to all the samples in our dataset:

1. **Lowercasing:** The first step in tweet pre-processing, consists on tweet lower-casing. Each word is lowercased so that the same word is not treated differently when encountered uppercased or lowercased.

2. **URL links removal:** News text on social media is often associated with URL mentions that tend to re-direct the user to other sources. These mentions do not contribute to our classification, but instead can perhaps contribute to an incorrect classification if they contain conflicting words.

3. **User Mentions removal:** All user mentions in the tweets do not add value to our classification and for this reason they can be safely removed.

4. **Contractions expansion:** Contractions are shortened versions of the words. They exist both in spoken and written forms of the English language. They are usually formed by removing one vowel from the word. Transforming contractions to their expanded version would help with standardizing the text.

5. **Tokenization:** Tokenization means splitting the sentence into individual tokens (words).

6. **Punctuation removal:** This step is purposely done after the URL and user mentions normalization, because otherwise the links would be split into sub-tokens and our regular expressions would fail to match the links effectively.

7. **Non-alphabetic words removal:** Any word that contains non-alphabetic characters is removed. An exception is made in the case of word 'covid-19'. This word is kept even though it contains numbers in it, because it conveys crucial meaning for our classification and should not be removed.

8. **Stop-word Filtering:** A stop word is a frequently used word in the natural language such as a, an, the, do etc. Removing stop words would leave us with fewer and only meaningful tokens, thus decreasing the dataset size, performing faster and improving the overall performance.

9. **Lemmatization:** This process consists in finding the lemma of a word (the base form based on context). We group together the different forms of a word so that they can be analysed as a single feature.

10. **Short words removal:** In this step all the remaining words that are less than 3 characters are removed. These might be words resulting from punctuation removal that do not convey any meaning.

The full pre-processing pipeline is summarized in Figure 3.

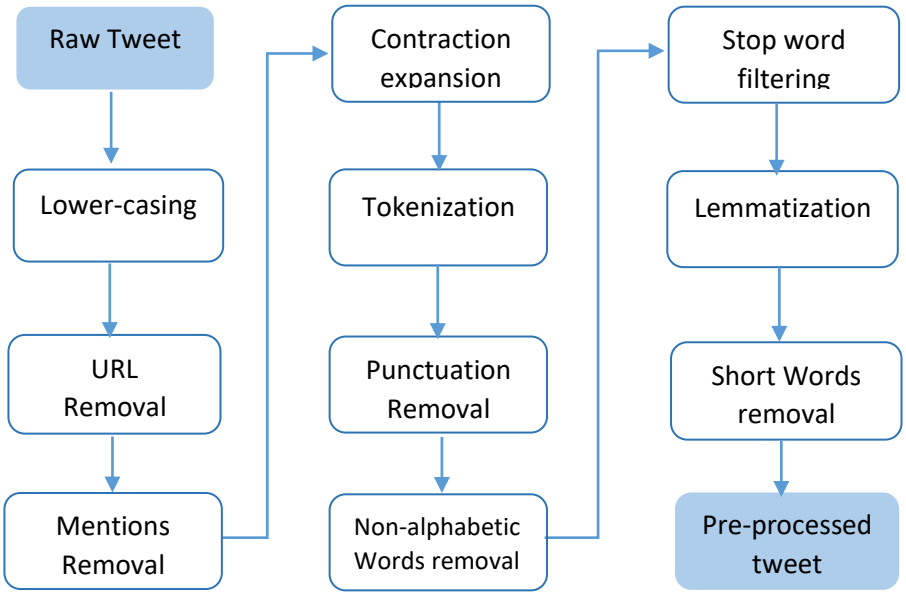

Figure 3 *Tweet pre-processing pipeline*

3.1.2 Feature extraction

In order to perform machine learning on text data, we need some feature extraction techniques to convert each tweet content into a numerical feature vector that the model can understand. The probabilistic model of naïve Bayes classifiers is based on Bayes' theorem, and the adjective naïve comes from the assumption that the features in a dataset are mutually independent. How can we extract mutually exclusive features from text data and calculate their probabilities to use in the Naïve bayes classifier? There are two approaches to this problem that will be consider:

3.1.2.1 Bag of words (BOW) representation

The Bag Of Words(BOW) model is a simplifying representation used in natural language processing and information retrieval (IR). Using the BOW model, a text is represented as an unordered collection of its words, disregarding grammar and even word order [14]. Each distinct word from the whole training dataset is assigned a weight according to its occurrence in each sample. Each occurrence is used as a feature for training the classifier. In our case, a fixed integer Id is assigned to each distinct word occurring in any tweet [15]. Then for each tweet, the number of occurrences of the word 'w' in that tweet is calculated and stored in position *X[i][w]* where *X[i]* is the feature vector representation of tweet '*i*'.

3.1.2.2 Term Frequency times Inverse Document Frequency (TF-IDF)

TF-IDF is a numerical statistic that is intended to reflect how important a given word is to a particular document in a collection or corpus [16]. It works by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus [17]. Computing TF-IDF representation of our BOW model, is composed of 2 steps:

1. **Term Frequency:** To perform Term Frequency, we divide the number of occurrences of each word in a tweet by the total number of words in that tweet. This way we consider as features, the frequencies and not the occurrences. This normalization is necessary in the case of long tweets that may have higher count values than shorter tweets, even though they may talk about the same topics.

$$TF = \frac{Number\ of\ times\ word\ \boldsymbol{w}\ appeared\ in\ tweet\ \boldsymbol{i}}{Total\ number\ of\ words\ in\ tweet\ \boldsymbol{i}} \quad (1)$$

2. **Inverse Document Frequency:** Another refinement applied on top of TF is to weight down the words that occur in many tweets and scale up the rare ones. From information theory, the more common a word is among the tweets, the less informative it is in comparison to those occurring only in a small portion of the corpus. The IDF representation of tweets is found applying the following formula:

$$IDF = \log\left(\frac{Number\ of\ tweets}{Number\ of\ tweets\ with\ word\ \boldsymbol{w}\ in\ it}\right) \quad (2)$$

3.2 Software Components

This project is implemented in Python Programming Language using Jupyter Notebook. The implementation depends on the usage of python libraries such as: *pandas* [18] for dataset loading and data analysis, *seaborn* and *matplotlib* [19] for data visualisation, *nltk* [20] for text pre-processing and *sklearn* [15] for feature extraction, Naïve Bayes implementation and classification metrics.

4 Methods

4.1 Naïve Bayes Algorithm

The supervised learning method used in this paper for fake news text classification is the Multinomial Naïve Bayes algorithm. Naïve Bayes is actually a family of classifiers that are based on the popular Bayes' probability theorem. The Bayes' Theorem, finds the probability of an event occurring given the probability of another event that has already occurred [21]. It is represented by the following formula:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3}$$

P(A|B) is the posterior probability needed to be calculated, P(B|A) is the conditional probability or the likelihood, P(A) is the prior probability and P(B) represents the evidence. Naïve Bayes is a probabilistic classifier, meaning that for a document *d*, out of all classes $c \in C$, the classifier returns the class which has the maximum posterior probability given the document [22]. Mathematically this is represented as:

$$\hat{c} = \underset{c \in C}{argmax}\ P(c|d) \tag{4}$$

4.1.1 Multinomial Naïve Bayes Classifier for tweet classification

This sub-section explains how Multinomial Naïve Bayes algorithm is applied to our specific problem. To apply Multinomial Naïve Bayes for text classification, we need to have a set of features and calculate the probabilities for their respective values. Multinomial Naïve Bayes considers a feature vector where the given term represents the number of times a word appears in a document [23] (tweet in our case), or more often the frequency. Performing the feature selection pipeline as presented in section 3, each tweet is represented as a feature vector using BOW model and is refined using Term Frequency – Inverse Document Frequency representation. We will use the frequencies in the data to learn the likelihood of feature f_i given class c: $P(f_i \mid c)$.

Supposing we want to classify a given tweet as real or fake. We need to calculate both the probabilities that the class is fake given the words that are in the tweet, and that the class is real given the words that are present in the tweet. These two probabilities need to be compared so that the model can predict the class of the given tweet.

It is assumed that a feature f_i is just the existence of a word w_i in the tweet's bag of words. So, we are interested in the finding probability that the news is fake when the tweet contains words like w_1 w_2 … w_n. This can be represented as $P(\text{fake} \mid w_1, w_2, \ldots w_n)$. To break down the problem further, a naïve assumption is made: the words (features) w_1, w_2, … w_n are considered to be independent of each other. Thus, based on Bayes theorem, the probability that the news is fake can be calculated as:

$$P(\text{fake} \mid w_1, w_2, \ldots w_n) = \frac{P(\text{fake}) \cdot [\, P(w_1|\text{fake}) \cdot P(w_2|\text{fake}) \cdot \ldots \cdot P(w_n|\text{fake})\,]}{P(w_1, w_2, \ldots w_n)} \quad (5)$$

Firstly, the prior probability P(fake) needs to be known. It is calculated as the number of tweets labelled as fake over the total number of tweets:

$$P(\text{fake}) = \frac{N_{fake\ tweets}}{N_{tweets}} \quad (6)$$

To find the likelihood for a feature, it is assumed that $P(f_i \mid \text{fake}) = P(w_i \mid \text{fake})$, which we compute as the fraction of times the word w_i appears among all words in all tweets labelled as fake [22]. This likelihood is calculated for every word (feature) present in the new tweet.

$$P(w_i \mid \text{fake}) = \frac{Count\ of\ w_i\ in\ fake\ tweets}{Total\ number\ of\ features\ in\ fake\ tweets} \quad (7)$$

Similarly, the probability that the tweet is real is calculated as:

$$P(\text{real} \mid w_1, w_2, \ldots w_n) = \frac{P(\text{real}) \cdot [\, P(w_1|\text{real}) \cdot P(w_2|\text{real}) \cdot \ldots \cdot P(w_n|\text{real})\,]}{P(w_1, w_2, \ldots w_n)} \quad (8)$$

The prior probability P(real) as:

$$P(\text{real}) = \frac{N_{real\ tweets}}{N_{tweets}} \quad (9)$$

The likelihood is computed as the fraction of times the word w_i appears among all words in all tweets labelled as real.

$$P(w_i \mid \text{real}) = \frac{Count\ of\ w_i\ in\ real\ tweets}{Total\ number\ of\ features\ in\ real\ tweets} \quad (10)$$

The posterior probabilities for both classes will be compared, so the evidence $P(w_1, w_2, \ldots w_n)$ which is the same for both classes can be ignored. The class with the maximum posterior probability determines the best predicted class for the given tweet.

4.1.2 Smoothing Technique

A key thing that is to be considered when applying the Multinomial Naïve Bayes for this tweet classification, is the possibility that a word in the test tweet, may never be present in class fake (or

class real). In this scenario, the likelihood is zero, and this implies that even the posterior probability will be zero. To prevent from such cases, Laplace Smoothing is used.

Using Laplace smoothing, a parameter alpha(α) = 1 is added to the denominator, and k = |V| is added to the numerator [24]. |V| is the number of words in the whole vocabulary. This technique will ensure that the likelihoods are never zero:

$$P(w_i \mid fake) = \frac{count\ of\ w_i\ in\ fake\ tweets\ +\ 1}{total\ number\ of\ features\ in\ fake\ tweets\ +\ |V|} \quad (11)$$

4.1.3 Pseudocode

Figure 4, represents the Pseudocode for the Multinomial Naïve Bayes Algorithm.

```
TRAINMULTINOMIALNB(ℂ, 𝔻)
 1  V ← EXTRACTVOCABULARY(𝔻)
 2  N ← COUNTDOCS(𝔻)
 3  for each c ∈ ℂ
 4  do N_c ← COUNTDOCSINCLASS(𝔻, c)
 5     prior[c] ← N_c / N
 6     text_c ← CONCATENATETEXTOFALLDOCSINCLASS(𝔻, c)
 7     for each t ∈ V
 8     do T_ct ← COUNTTOKENSOFTERM(text_c, t)
 9     for each t ∈ V
10     do condprob[t][c] ← (T_ct+1) / (Σ_t' (T_ct'+1))
11  return V, prior, condprob

APPLYMULTINOMIALNB(ℂ, V, prior, condprob, d)
1  W ← EXTRACTTOKENSFROMDOC(V, d)
2  for each c ∈ ℂ
3  do score[c] ← log prior[c]
4     for each t ∈ W
5     do score[c] += log condprob[t][c]
6  return arg max_{c∈ℂ} score[c]
```

Figure 4 *Multinomial Naive Bayes with Laplace Smoothing Pseudocode [25]*

4.2 Validation Method

In this work, k-fold cross validation method with k = 5, is used for evaluation. The original training and validation dataset are combined into one. The combined dataset is split randomly into k folds (subsets) of equal sizes. Then the model is trained and tested k times. We provide the accuracy, recall, precision and F1-score for each fold as well as the average one. Accuracy is calculated as the number of correct predictions over the total number of predictions. Precision is calculated as the ratio of the correct positive predictions to the total number of positive predictions. Recall is calculated as the ratio of correct positive predictions to the total number of positive labels and F1-score is calculated as the weighted average of precision and recall. Once the model is trained on the training data, it is then applied to the test set, that was never seen during training. The performance is evaluated again on this test set. This is done to make sure the model is more generalizable and hasn't just learned the training data.

5 Results

This section represents the results obtained from performing the above pipeline on the given dataset. The Multinomial Naïve Bayes model is trained on 8560 tweets (train + valuation datasets) and evaluated using 5-Fold cross validation. Each tweet is fed into the model as a 12780 dimension feature vector obtained from performing BOW + TF-IDF on the pre-processed tweets. Table 1, summarizes the evaluation metrics for each test fold as well as the mean for the entire process.

Table 1 *Validation scores reported from 5-fold cross validation*

Metric	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average
Accuracy (%)	89.95	89.66	90.3	89.36	90.59	89.95
Macro Precision (%)	90.28	90.35	90.8	89.98	90.87	90.45
Macro Recall (%)	89.6	89.37	90.06	89.09	90.41	89.7
Macro F1-score (%)	89.75	89.55	90.22	89.26	90.53	89.9

After training the model on the whole train + validation dataset and evaluating using 5-fold cross validation, we try to predict the tweet labels for the test set, that was never seen during training. This is done to make sure the model is more generalizable and hasn't just learned the training data. Figure 5, shows the confusion matrix for this prediction.

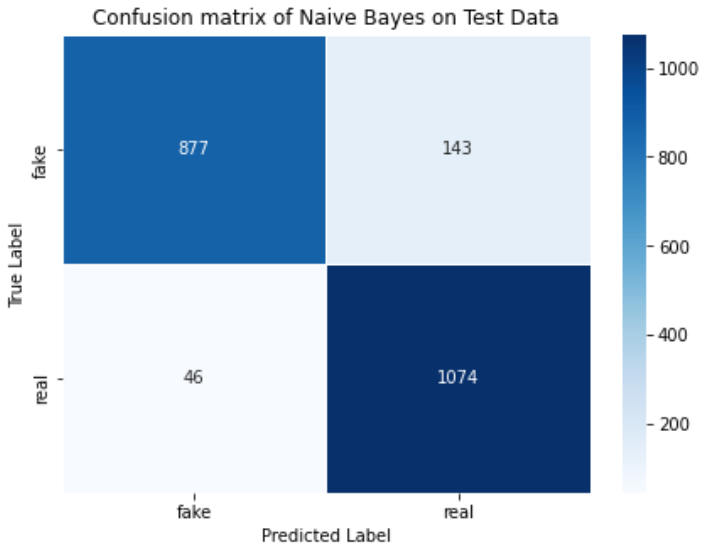

Figure 5 *Confusion Matrix*

As presented in the confusion matrix, with respect to the fake class (fake = 1, real = 0), there are 877 True Positives (tweets that are fake and are predicted as fake), 1074 True Negatives (tweets that are real and are predicted as real), 143 False negatives (tweets that are fake but are predicted as real) and 46 False positives (tweets that are real but are predicted as fake). From these results, the accuracy, precision, recall and F1-scores can also be obtained. Figure 6 represents these scores for both classes as returned from the program.

```
_________________Naive Bayes Classifier_________________

              precision    recall  f1-score   support

        fake     0.9502    0.8598    0.9027      1020
        real     0.8825    0.9589    0.9191      1120

    accuracy                         0.9117      2140
   macro avg     0.9163    0.9094    0.9109      2140
weighted avg     0.9147    0.9117    0.9113      2140
```

Figure 6 *Evaluation Scores for the testing dataset*

This model managed to achieve a 91.17% Accuracy and 91.13% weighted average F1-Score on the testing set. We can see where this Naïve Bayes classifier stands in comparison to the baseline models benchmarked by Patwa et al. in [4]. Since this dataset was released as part of a competition, we can make such comparison. The results by Patwa et al. are obtained from their paper. Table 2, represents the comparison.

Table 2 *Comparison to the baseline results*

Model	**Accuracy %**	**Precision %**	**Recall %**	**F1-score %**	
Naïve Bayes	91.17	91.47	91.17	91.13	This paper
Decision Tree	85.37	85.47	85.37	85.39	Patwa et al.
Linear Regression	**91.96**	**92.01**	**91.96**	**91.96**	
SVM	**93.32**	**93.33**	**93.32**	**93.32**	
Gradient Boost	86.96	87.24	86.96	86.96	

As it can be seen from the table, the Naïve Bayes classifier presented in this paper surpasses the Decision Tree and Gradient Boost Models presented from Patwa et al. Meanwhile the results obtained from their SVM and LR classifiers are higher than the results obtained from the Naïve Bayes model that was presented in this paper. Furthermore, if we were to compare the results with the pre-trained models based on transformers used extensively in [6] [7] [13], it is obvious that those models outperform the simple probabilistic Naïve Bayes presented in this paper. However, the time complexity to train those models is considerably higher, in comparison to our Naïve Bayes model that was trained for a matter of seconds.

6 Conclusion

The aim of this paper was to help in combating the Covid-19 infodemic by constructing a machine learning model for the classification of fake news spreading around social media. A dataset consisting of 10700 fake and real tweets is taken into consideration. A natural language processing pipeline is used to pre-process the tweets and convert them into a clean format. This pipeline consists of: lowercasing, tokenization, URL normalization, username removal, contractions expansion, stop-word filtering and short words removal. As feature extraction techniques BOW and TF-IDF are used to convert each tweet into a V-dimensional feature vector, where V is the vocabulary for the training set. The ML algorithm considered in this paper is the Multinomial Naïve Bayes, which is suitable for the

classification of discreate features (word count for text classification). Taking in consideration that it is one of the simplest Machine Learning Models that doesn't even capture the content of the words, just their probabilities of occurrence, this model achieved a premising result of 91% accuracy and F1-score. It managed to outperform the Decision Tree and Gradient Boost models from the baseline paper [4], however it performs worse than the SVM machine presented in [4]. Running this implementation was done with ease since the Naïve Bayes represents a fast and accurate method of prediction that requires low computational cost.

References

[1] D. Carrion-Alvarez and P. X. Tijerina-Salina, "Fake news in COVID-19: A perspective," *Health Promot Perspect,* vol. 10(4), pp. 290-291, 2020.

[2] M. Cinelli et al., "The COVID-19 social media infodemic," *Scientific Report,* vol. 10, no. 16598, 2020.

[3] K. Ding, K. Shu, Y. Li, A. Bhattacharjee and H. Liu, "Challenges in Combating COVID-19 Infodemic -- Data, Tools, and Ethics," *arXiv preprint arXiv:2005.13691,* May 2020.

[4] P. Patwa et al., "Fighting an Infodemic: COVID-19 Fake News Dataset," *arXiv:2011.03327 [cs.CL],* Nov 2020.

[5] T. Felber, "Constraint 2021: Machine Learning Models for COVID-19 Fake News Detection Shared Task," *arXiv:2101.03717 [cs.CL],* 2021.

[6] A. Wani et al., "Evaluating Deep Learning Approaches for Covid19 Fake News Detection," *arXiv:2101.04012 [cs.LG],* Jan 2021.

[7] T. Raha et al., "Identifying COVID-19 Fake News in Social Media," *arXiv preprint arXiv: 2101.11954,* 2021.

[8] M. Granik and V. Mesyura, "Fake news detection using naive Bayes classifier," in *IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, Kyiv, UKraine, 2017.

[9] J. Zhang et al., "Fake news detection with deep diffusive network model," *arXiv:1805.08751,* 2018.

[10] H. Karimi, P. Roy, S. Saba-Sadiya and J. Tang, "Multi-source multi-class fake news detection," in *Proceedings of the 27th international conference on computational : pp. 1546–1557*, 2018.

[11] A. Rogers, O. Kovaleva and A. Rumshisky, "A primer in bertology: What we know about how bert works," *Transactions of the Association for Computational Linguistics,* vol. 8, pp. 842-866, 2020.

[12] H. Jwa, D. Oh, K. Park, J. M. Kang and H. Lim, "exBAKE: Automatic Fake News Detection Model Based on Bidirectional Encoder Representations from Transformers (BERT)," *Appl. Sci,* vol. 9 (19), 2019.

[13] S. D. Das, A. Basak and S. Dutta, “A Heuristic-driven Ensemble Framework for COVID-19 Fake News Detection,” *arXiv preprint arXiv:2101.03545,* 2021.

[14] G. K. Soumya and J. Shibily, “Text Classification by Augmenting Bag of Words (BOW) Representation with Co-occurrence Feature,” *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 16, no. I, pp. 34-38, 2014.

[15] Pedregosa et al., “Scikit-learn: Machine Learning in Python,” *JMLR 12,* pp. 2825-2830, 2011.

[16] M. Apra and V. Santosh, “Analysis of TF-IDF Model and its Variant for Document Retrieval,” in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, Jabalpur, India, 2015.

[17] J. Ramos, “Using TF-IDF to Determine Word Relevance in Document Queries,” *Proceedings of the first instructional conference on machine learning,* vol. 242 (1), pp. 29-48, 2003.

[18] McKinney, “Data structures for statistical computing in python,” *Proceedings of the 9th Python in Science Conference,* vol. 445, 2010.

[19] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Computing in Science & Engineering,* vol. 9, no. 3, pp. 90-95, 2007.

[20] S. Bird, L. Edward and K. Ewan, Natural Language Processing with Python, O’Reilly Media Inc., 2009.

[21] S. Raschka, “Naive Bayes and Text Classification I, Introduction and Theory,” *arXiv:1410.5329v4 [cs.LG],* 2014.

[22] J. Daniel &. J. H. Martin., “Naive Bayes and Sentiment Classification,” *Speech and Language Processing,* 2020.

[23] A. M. K. F. P. Holmes, “Multinomial Naive Bayes for Text Categorization Revisited,” *Lecture Notes in Computer Science LNCS,* vol. 3339.

[24] V. Cherian and Bindu M.S, “Heart Disease Prediction Using Naïve Bayes Algorithm and Laplace Smoothing Technique,” *International Journal of Computer Science Trends and Technology (IJCST) ,* vol. 5, no. 2, 2017.

[25] C. D. Manning, P. Raghvan and H. Schulze, Introduction to information retrieval, Cambridge: Cambridge University Press, 2008.

APPENDIX

Code: github.com/aligori/fake-news-detection/blob/main/Fake-News-Detection.ipynb