Prediction of Fake Covid News with Classification Algorithms

Marina Amorim
Drury University
Springfield, MO
mamorim@drury.edu

Scott Sigman
Drury University
Springfield, MO
ssigman@drury.edu

Shannon McMurtrey
Drury University
Springfield, MO
smcmurtrey@drury.edu

***Abstract*— Covid-19 has impacted each of us in different ways. We have now been living in this new reality for over two years. It now comes to our attention how everything we have done as a society affected the development of this outcome. Constantly, news articles and posts about the pandemic were spread all over the Internet. Since the beginning of the Covid-19 pandemic, it was noticeable that a significant amount of this news was fake. This creates a necessity to ensure the validity of such articles and posts, so the reading users do not lose trust in the news platform. By analyzing not only how to predict the veracity of a piece of news, but the importance of pre-processing we can build an algorithm that predicts with high accuracy the veracity of a given Tweet.**
Keywords— Covid-19, Fake News, Preprocessing, Classification

I. INTRODUCTION

In today's world, we are connected through social media all the time. We have access to any kind of information if we have an internet connection and a device. Consequently, we are surrounded constantly by information. The veracity of the messages we can access impact sour lives not only as individuals but also as a society and come from all kinds of users such as influencers, news accounts, and politicians. These unreliable pieces of information are known as fake news, which is defined by the Cambridge dictionary as "false stories that appear to be news, spread on the internet or using other media, usually created to influence political views or as a joke" [3].

During major events such as the 2016 elections or the Covid-19 pandemic, it became clear the power these kinds of news have in changing the course of events. According to an article in Statista 80% of Americans report they had seen fake news regarding the Covid-19 pandemic [1].

Covid-19 is a disease caused by the SARS-CoV-2 virus. It was characterized as a pandemic by the World Health Organization (WHO) in March of 2020 [2] and since then has impacted our lives. Articles that go from the use of certain kinds of medication without the proper research to false assumptions over the use of masks were widely spread during this period. Since information about the virus was being researched as we faced the pandemic, it was hard to keep up with the discoveries and consequently hard to recognize false information. The dimension of the spread of these messages had unprecedented consequences. At some point, not even big media companies were able to determine what was reliable information. We saw several cases of well-known sources doing retaliations after sharing misinformation.

Being able to recognize what kind of information is true and what is not is an important tool for the population in general. Machine learning algorithms are a widely used tool for recognizing patterns. These algorithms can be applied to news to detect information validity. When it comes to detecting fake news related to the Covid-19 pandemic, it is important to notice the nuances of these specific kinds of news. In other words, we must carefully consider syntax composition, use of acronyms, capital letters, and others since these factors can significantly impact the meaning of a message.

In this paper, I will be using known classification machine learning algorithms such as logistic regression, and KNN among others to detect possible fake news related to Covid-19 on tweets to detect fake news related to Covid-19 on tweets. Furthermore, I will compare the accuracy of different models using different kinds of preprocessing on data. I will then see which kind of preprocessing works better for each model I have used.

II. BACKGROUND

1) Fake news on Twitter

Thousands of new messages are posted every second on social media platforms. This amount of constant data makes it extremely hard to filter and validate the information. Social media platforms have been struggling with the spreading of false information in their platforms. Users are allowed to post anything they want under the guidelines of the platform. The problem is these guidelines do not guarantee that the information users are uploading is trustworthy. Influencers, media accounts, actors, and political figures have an immeasurable power over their audience and can reach millions of people around the world. If we look at the most followed

account on tweeter, Barak Obama has 131.4 million people following him. It is not surprising that a post by one of these figures can be worth millions of dollars and influence a large group of people. Furthermore, a tweet from a powerful account like this one can have the same impact, if not more, than a newspaper. If we take into consideration certified news accounts, the messages can be even more impactful.

For this paper, all data were obtained from Twitter. This platform considers the spread of misleading information as a violation of one of its policies [17]. Fact-checking tools are being developed and implemented, but the issue with false information is still far from being resolved. Since fake news tends to have a sensational and high emotional impact, they also reach big audiences faster than other types of tweets. For this reason, there is a necessity of improving the checking tools to be able to detect fake news with higher accuracy.

The issue with fake news particularly related to Covid-19 on Twitter is so extensive that Twitter has created a policy just for this kind of news. Global phenomena usually provoke a higher chance of fake news. For example, due to the high amount of fake news related to Covid-19, Twitter created a new policy dedicated to this type of tweet [17]. The policy explains patterns and behaviors not accepted for this specific type of news.

2) Text preprocessing

Text data contains different kinds of noise such as punctuation, text in a different case, and stop words. In some scenarios, noise can negatively affect the accuracy of the models. The process of removing the noise and making the data more friendly for the models is called preprocessing. To build better machine learning algorithms it is important to preprocess our data before categorizing it [12].

When it comes to data from tweets how noise impacts the message is even bigger. Social media communication relies a lot on full uppercase words to express emotions. For example, the word COVID and Covid have the same meaning, but the word typed in all capital letters has more impact. It also includes lots of hashtags, abbreviations, and slang. These features might be easy for humans to read and understand, but computers do not work the same way. It is necessary to refactor our data to make it more computer friendly.

Although preprocessing can help our results, we must be careful to not lose meaningful information during the process. Sometimes if we clean our data too much, we might also end up cleaning parts of the data we did not want to. For example, the use of abbreviations or exclamations in tweets can be an indicator of fake news. By removing them the algorithm may turn less accurate. For this reason, it is important to test how our models perform with different kinds of preprocessing.

III. METHODOLOGY

In this section, I will talk about the methodology used for my experiments. I will explain each of the steps performed to build the different classification algorithms used.

1) The dataset

The chosen dataset contains three tables in the comma separated value format (csv): a labeled training file, a labeled test file, and an unlabeled test file. The first one has two columns one with tweets and one with the tweet's respective labels that can be real of fake. This table contains around 6400 different tweets in English. The tweets are strings with up to 140 characters. The second and third files are the test files. These files will be used to check the accuracy of the models built. The labeled test file has the same structure as the labeled train file but contains different tweets. The unlabeled test file only has the tweets.

2) Libraries

To perform our machine learning algorithms, we need to import some libraries first. A library is a collection of modules to perform specific operations. The most important libraries I will be using are Pandas, Skcit learn, and numpy to perform operations such as data analysis, classification, and data manipulation.

3) Experimental enviroment

The first step on my project was to explore the type of data with which I was dealing with. The environment used to analyze my data was Google Colab. First, I made sure that the data in my training dataset was well distributed between fake and real news. To do that I grouped the data by the labels' size. As a result we have that the table contains 3060 fake news and 3360 real ones, as shown in the bar graph bellow. That means our data is nicely partitioned.

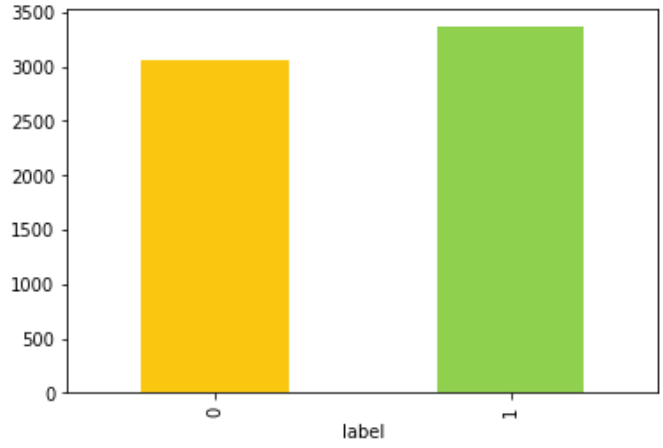


Fig. 1. Bar graph of fake real distribution

4) Preprocessing

4.1) One-hot-encoding the labels

The first step on preparing the data is to one-hot-encode the labels. One-hot-encoding means replacing categorical variables of fake and real with 0's and 1's. Categorical values are variables that contain labels instead of numerical values. Some machine learning algorithms work with categorical values such as decision tree for examples, but many others such as logistic regression will not. Hence, to be able to use build a variety of models, the best practice is to convert the categorical variables to numeric values.

4.2) Bag-of-words algorithm

The same way we should hot-encode the labels in the data set, we cannot simply leave the tweets as text. Algorithms work better with structured and organized data, opposing as how the tweets and overall text data is: unstructured and messy. Converting our text data to a more structured "Find word" is necessary. The Bag-of-words model (BoW) is a technique of text modeling. It turns our unorganized text into a vector with fixed length. For that reason, this model can also be called vectorizing.

The two steps in vectorizing are to create the vocabulary and then to count. The first step consists in defining a set with the words in the tweets. The second consists of counting the appearances of each word. These steps will be performed after splitting the data into train and test.

4.3) Stop words

Stop words are words such as "there", "they", and "a". there are words that appear in a high frequency in out texts but have no meaning to our classification. Using the NLTK library we can remove around 180 different stop words from the data. If we take the following tweet: "Masks can help prevent the spread of #COVID19 when they are widely used in public" and remove the stop words we are left with "Masks help prevent spread # COVID19 widely used public". As we can see, words such as can, the, and of were removed from our message.

4.4) Lower casing

Lower casing is the process of turning the text to the same case. It is important to notice that for text that involve emotions, lowercasing might not always be a good choice since writing in uppercase can be a sign enthusiasm or disappointment for example. When talking about COVID, we might have it written all uppercase or capitalized (Covid) or even all lowercase (covid). If we don't apply lowercasing in our algorithm, these three ways of writing it will be considered different occurrences.

4.5) Removing Punctuatuon

The data is full of punctuation and that usually does not provide a lot of meaning to our classification. Text from tweets is full of hashtags and exclamation points for example. If we take #covid! and covid for example are considered different without the punctuation removal.

4.6) Stemming

Stemming is the process of standardizing words. During this part of the preprocessing, we are going to replace the words for its base form. The word books for example would be replaced by book. I will be using one of the most common stemming algorithms, the Porter stemming algorithm. This algorithm removes commons morphological and inflexional endings from words [Source]. The message "Italy has surrendered to the coronavirus pandemic as all the measures to control COVID-19 have been exhausted." after applying the Porter Stemmer algorithm will look like "itali ha surrend to the coronaviru pandem as all the measur to control covid-19 have been exhausted."

4.7) Lemmatization

Lemmatization, the same way as stemming, reduces words by replacing them with a common base form. The big difference between both is that lemmatization uses lexical knowledge to do so instead of simply cutting off part of the words. Taking the following tweet "Multiple posts shared thousands of times on Facebook Twitter and YouTube claim that salt is an effective remedy against the novel coronavirus." and running the word lemmatizer algorithm we will get "Multiple post shared thousand of time on Facebook Twitter and YouTube claim that salt is an effective remedy against the novel coronavirus."

4.8) Expanding Contractions

In the English language we can contract some expressions. We can say I am or I'm for example. If we simply leave our text with contractions, these two ways of writing the same thing will be considered as different by our model. That is where expanding contractions comes to place. Creating a dictionary with common contractions and its expansions, we can expand the contractions in the text and make them equal. We will then check in our text for the expressions in the dictionary and replace them with the expanded versions. If we take the tweet "We're debunking misinformation about coronavirus" and expand the contractions, we get: "we are debunking misinformation about coronavirus"

5) Classification Algorithms used

In machine learning classification algorithms are supervised algorithms. In classification algorithms we predict the category of a new piece of data based on a model created using training data. In other words, we input training data to create a model and we look at how the model makes predictions in the testing data. Then, we compare the predictions in the testing data with its true labels. In the training data I have split the data into training and testing with the testing size representing 20% of the data.

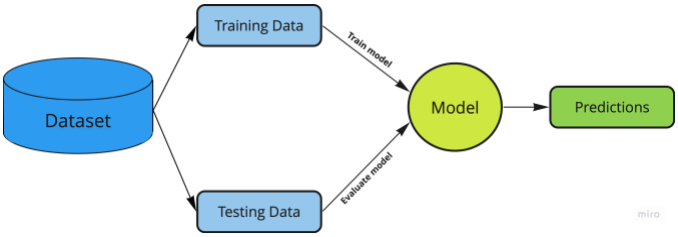


Fig. 2. Diagram of a classification algorithm

To do so I used the train test split and I have split my data into 4: X_train, X_test, y_train, y-test. The X_train represents the tweets we will be using to train the model and y_train the labels used to train the model. The X_test will be used to make a prediction using the models to be compared with the values in y_test. The next thing that's required is to vectorize the text data.

To vectorize my tweets, I have used the Scikit learn Count Vectorizer. In other words, I converted the text into a matrix of token counts "Table 1". For the training data (X_train) have used fit transform to learn the vocabulary dictionary and return a documented-term matrix (DTM) limiting our unique words to 5000. The DTM represents the number of occurrences of each term in each one of the tweets. For the testing data (X_test) I used the transform to transform the tweets into DTM.

TABLE I.

	Tweet 1	**Tweet 2**	**Tweet 3**	**…**	**Tweet n**
Term 1	0	1	1	0	0
Term 2	1	0	3	0	0
Term 3	0	1	1	1	1
…	1	0	0	0	0
Term m	1	1	0	1	1

I will be comparing the results of my models with and without preprocessing the data to find the best way of building the model.

5.1) Multinomial Naive Bayes

The Naïve Bayes classifier is a probabilistic machine learning classifier [2] algorithm uses the Bayes theorem (2), where y and X are events and P(X) is not 0, to calculate the probability that a certain item is part of a category.

$$P(y \mid X) = \frac{P(X \mid y)P(y)}{P(X)} \quad (2)$$

The Naïve in the name comes from the fact that we pretend that each probability is conditionally independent to simplify the calculations. There are different kinds of naïve bayes algorithms that are suited for various applications. In the case of a classification with discrete features such as the word counts in a text the multinomial naïve bayes classifier algorithm is a good option.

For that reason, in my code I have used the skit-learn multinomial naïve bayes classifier algorithm with the parameter *alpha=0.2*. I have tested different values for *alpha*, but 0.2 was the one that game me more accuracy.

5.2) Decision Tree

In the decision tree algorithm, we use a tree structure to split the learning tasks performed by the model. The decision tree algorithm works from the top to the bottom using the design and conquer approach. The algorithm starts with a root node and gives the results mad by the leaves(maybe fix?).

I have used the skit-learn decision tree classifier algorithm.

5.3) Logistic Regression

Logistic Regression works good with binary outcomes, such as our fake and real labels. We can think of Logistic Regression as Linear Regression for classification problems [source]. In Logistic regression we use the Logistic Function (1). In the equation x is the input value and in the case of our dataset the tweets. Mathematically, we are dividing the probability of the event occurring by the probability of it not occurring.

$$\frac{1}{1+e^{-x}} \quad (1)$$

The probability curve for (1) is sigmoid-shaped and constrained between 0 and 1 'Fig. 6'.

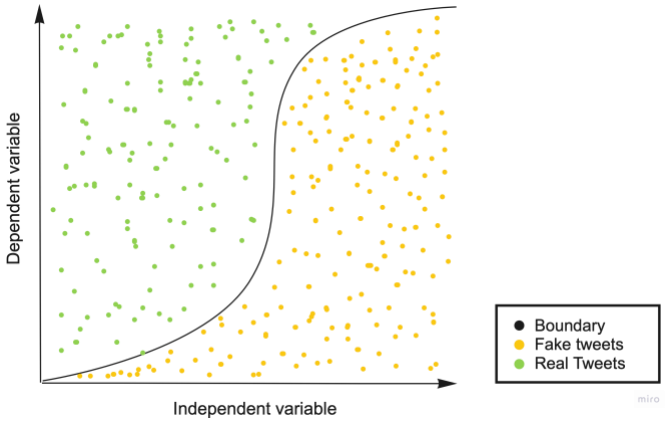


Fig. 3. Logistic Regression

I have used the skit learn Logistic Regression algorithm. [10]

5.4) KNN Classifier

The K-nearest neighbors classifier algorithm in a simplified way stores all the training data and classifies the testing data based on its similarity to the stored data. It is considered a lazy-learner algorithm because it only stores the data during the training phase and does the classification in the testing phase.

I have used the skit lean KNN algorithm with a parameter of 3 neighbors.

5.5) Support Vector Classifier

Support vector classifiers works by spiriting the values using a line called a hyperplane. The idea behind it is to find the line that separates the values with the widest possible distance. To do so, it uses support vectors, and that's where its name comes from. These types of classifiers can perform multiclass classification. In my model I have used two kinds of support vector classifiers: the skit-learn Linear Support vector classifier (LVC) and the (SVC).

6) Predictions

Finally, I will use the created models to predict the results from my testing portion of the data. To do so, I will run my models in the y_train and compare the predictions to the actual values of y_test. To do so I used sklearn metrics accuracy score with the y_train and y_test as my inputs.

After that I imported the test file. I had to one-hot encode the labels to 0's and 1's again. After that, I had to it and perform the same steps I performed in the test part of the training data. Followed by that, I used the accuracy score again but now with the whole table using the tweets as my first parameter and the labels as the second.

7) Evaluating models

To evaluate the predictions made by my models, I have used confusion matrices and accuracy score as tools to evaluate the different models. Using these tools allowed me to compare how the different models and the different preprocessing techniques perform against each other.

7.1) Confusion Matrix

A confusion Matrix is a table that portrays the performance of the models. I'll be using a 2x2 confusion Matrix as in 'Fig 1' to later evaluate my models. A 2x2 matrix consists of 4 instances: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). For the evaluation of the models a TP means we classified a real tweet as real, while a TN means a fake news was classified as fake. The same thing happens with the false cases, where a FP means we incorrectly classified a real tweet as false and a FN that we classified a fake tweet as real. In the case of the data used, we want to mostly avoid labeling a fake message as a real one. That means we want to mostly avoid the FN results. We can use confusion matrices to evaluate the quality of our models.

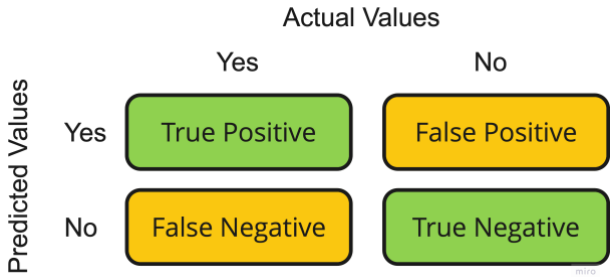


Fig. 4. Confusion Matrix

7.2) Accuracy

The accuracy of the model is defined by the number of correct predictions dived by the total number of predictions. We want to know the accuracy of our models not only to determine which ones better predict the veracity of the messages, but also to be able to analyze how different preprocessing tools influence in the quality of the model. The accuracy can also be represented in terms of the confusion matrix instances as in 'Fig 2'.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

I will compare the accuracy of not only the different models, but also compare how the different kinds of preprocessing impact the accuracy "Fig 2". Using the accuracy values obtained from the different models, we can determine the most efficient ones and get insights to make the changes needed in each one of the models to improve them.

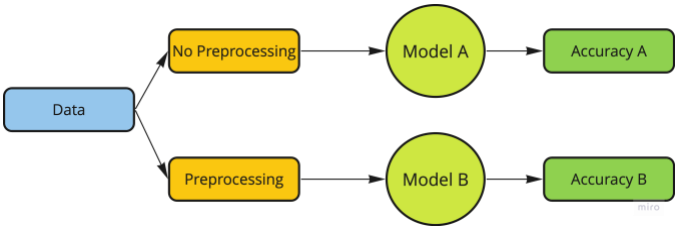


Fig. 5. Accuracy for different models

IV. Results

The models performed different depending on if the data was preprocessed or not. Whether using all kinds of preprocessing at the same time or just some influences in the accuracy scores. I will be using accuracy scores and confusion matrices to evaluate my models.

Depending on how we split the data, we can have a training group with different characteristics that can slightly modify the accuracy of the models. Those changes are more noticeable when we look at the preprocessed data. The reason for that is that when we preprocess, we are removing punctuation, stemming words, and lowercasing for example. In some pieces of the data these alterations can have more impact.

For that reason, I have created a matrix with my models and the different preprocessed data. I ran the model for 20 times shuffling the data each time and appending each probability to an array. I then calculated the average and returned an array with the different average precisions for each model. That allows us to look at how different models are build depending on how the data is split.

I will first evaluate my models in the testing portion of my training dataset with both preprocessed and non-preprocessed data. Next, I will import the test dataset and use my models in the new data. I will also look at both non preprocessed and preprocessed data.

1) Non preprocessed data

First, we should look at how different algorithms performed without using the preprocessing tools. The performance of the models in the data without preprocessing is represented in table 1 bellow.

TABLE II.

Model	**Accuracy**
Nayve Bayes	91.1
Decision Tree	84.25
Logistic Regression	91.26
KNN Classifier	72.14
LVC	90.19
SVC	90.73

The algorithm that best performed without the preprocessing was Logistic Regression. This algorithm is used when the dependent variable that we are trying to predict is categorical. In our case we are performing a binary Logistic Regression to determine fake and real news. The algorithm that performed the worse was the KNN classifier. Its accuracy was the lowest between all the other used models.

2) Preprocessed data

The next step was to go back and preprocess the data before training the models. I performed the same previous steps in the preprocessed data. I have combined different preprocessing tools and evaluated the different accuracies.

After using the preprocessing tools in the data, I got to different results as shown in "Table 3"; where 1 is removing stop words, 2 is stemming, 3 is lowercasing, and 4 is lemmatization.

TABLE III.

	Accuracy			
Model	**1,3**	**2,3**	**1,2,3**	**3,4**
Nayve Bayes	90.63	90.90	91.10	90.87
Decision Tree	84.25	84.79	83.70	83.90
Logistic Regression	91.76	91.69	92.08	91.86
KNN Classifier	72.96	72.14	72.30	72.07
LVC	90.50	90.91	90.62	90.56
SVC	91.30	90.98	91.16	90.84

The first important thing to notice is that both expanding contractions and removing punctuation did not improve the accuracy of the model. There are a few reasons that can explain why this happened that are important to point out.

When it comes to removing the punctuation, it is important to notice a few things. First, twitter's language is heavily dependent on hashtags and a hashtag might not have the same meaning as the word in its own. Second, punctuation can add emotions to a word. If we treat "covid" and "covid!" as the same thing we might be removing the emotion that comes with using the exclamation point. Consequently, we end up considering words with slightly different meanings the same thing and that can interfere in the classification of the tweets.

As for expanding contractions, we should notice that using contractions is an informal way of writing. Even if "aren't" and "are not" have the same meaning in the sentence, they have different formality levels. That can influence in a tweet being real or fake. If we remove the contractions from our tweets, we might end removing this trace of informality that can give us important information about the tweet.

The next thing I did was to import the test dataset. It is important not only to test the results using the data we used to train the model, but also to test it another dataset. In the real world we will be putting our models to predict new data each

time. For that reason, only looking at the test portion of the training dataset is not a reliable way of checking how the model really performs.

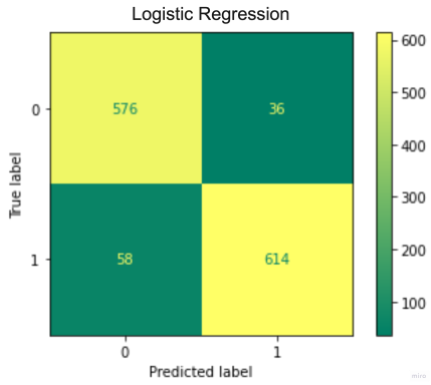
After importing the test dataset, I have performed the steps of preparing the data. Now the data is not split. I will use the previous models to predict the tweets and compare it to the label. The results for the non-preprocessed data are shown in "Table 4"

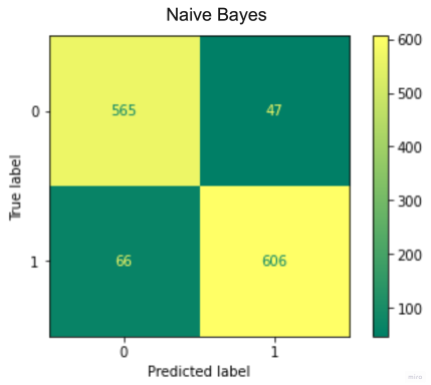
TABLE IV.

Model	**Accuracy**
Nayve Bayes	91.88
Decision Tree	89.37
Logistic Regression	93.83
KNN Classifier	78.04
LVC	93.26
SVC	93.12

We can see that the results were overall better than the ones we had in the test portion of our training data, which is an interesting phenomenon. This tells us the algorithms work well not only with the data used to train it but also in new data. For Logistic regression, for example, the accuracy of the non-preprocessed data in test table was 91.2, while in the test table was 93.26.

Followed by logistic regression, we have LVC, and SVC. Bellow we can see the Confusion Matrices for Logistic Regression, Naïve Bayes, and LVC.





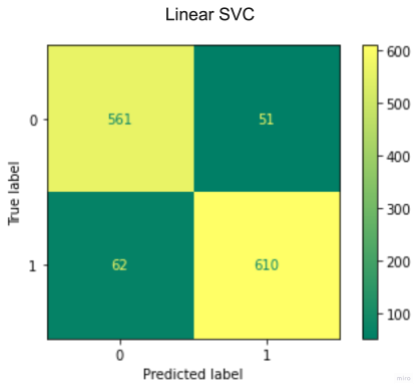


Fig. 6. Confusion Matirces for non preprocessed data where 0 is fake and 1is real

The confusion matrices give us important information about the models. First, I looked at the correct predictions when the predicted label matches the true one. The logistic regression predicts both TP and TN better than the other two algorithms with 576 fake news correct labeled as fake and 614 real ones labeled as real. Comparing Naïve Bayes and LVC the results were very similar; we see that the first one better predicted real news as real (610 vs 606) while the second was better at labeling fake news as fake (565 vs 561).

Next, I will look at the wrong predictions, when the predicted label is not the same as the expected. For real news labeled as fake, logistic regression had a slightly better performance again, as the only model with a value less than 60. As for fake news labeled as fake, the worst case we can have, logistic regression was better by a bigger difference, with only 36 occurrences against the values around 50 for the other algorithms.

The next thing we should do is to look at the performance of the preprocessed algorithms in the testing data. The results are again better than the ones from the testing part of the training data as shown in "Table 5' where 1 is removing stop words, 2 is stemming, 3 is lowercasing, and 4 is lemmatization.

TABLE V.

	Accuracy			
Model	**1,3**	**2,3**	**1,2,3**	**3,4**
Nayve Bayes	91.85	91.94	91.85	91.97
Decision Tree	89.22	88.16	89.04	88.16
Logistic Regression	93.74	93.56	93.52	93.74
KNN Classifier	78.34	77.34	78.19	77.09
LVC	93.14	93.04	92.91	93.28
SVC	93.04	93.01	92.92	93.11

As shown in “Table 5”, we can see that the overall best performing combination of preprocessing tools was to combine lowercasing, removing stop words and stemming. The algorithms that better performed were logistic regression, SVC, and naïve bayes with the confusion matrices shown below “Fig 7”.

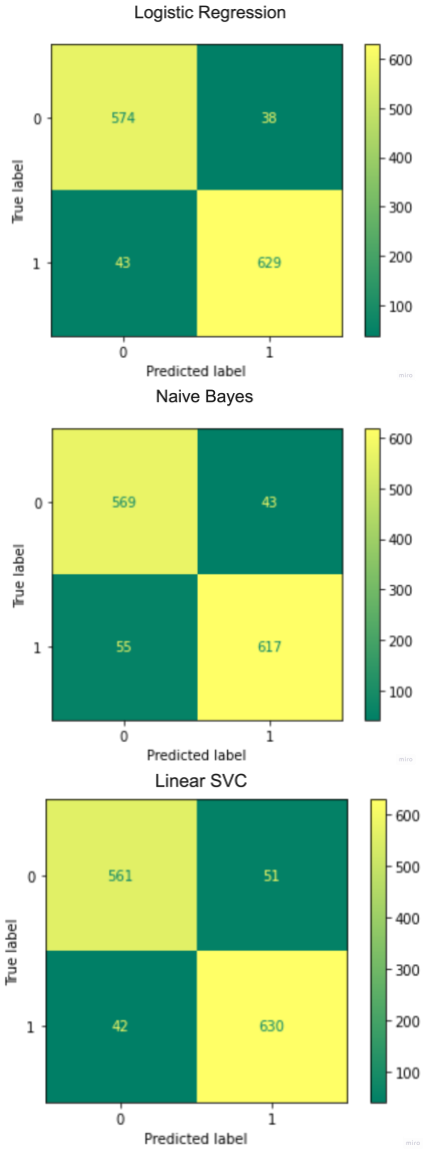


Fig. 7. Confusion Matrices for preprocessed data where 0 is fake and 1is real

For logistic regression we can see that the predictions for real news (TP and FP) were basically the same as the ones in the previous not preprocessed data. The improvement with the preprocessing occurred in the detection of fake news, where there was less cases of FN and more cases of TN. In other words, for logistic regression the use of preprocessing tools improved de prediction of fake news.

When it comes to Naïve Bayes the improvement was more balanced between the prediction of real and fake news. The preprocessed model had less FN and FP and more TN and TP than the non-preprocessed model. That means the preprocessed Naïve Bayes model is better than the non-preprocessed at predicting both fake and real news.

As for the LVC, we had the same phenomena as logistic regression. The preprocessed algorithm works basically the same as the non-preprocessed in the detection of real news but works better when it comes to detecting fake news.

The best performing algorithm was the logistic regression preprocessed with lowercasing and only the removal of stop words or just lowercasing and applying lemmatization.

V. Related Work

[2] The author uses the Naive Bayes classifier to evaluate if the news is real or fake. The model is a simplification on how to recognize news but is very informative and explicative. It first explains how Naive Bayes works, and then proceeds to perform the classification. In this article, some concepts of probability are covered as well as the use of Python. The model can help us better understand how to use Naive Bayes. The hypothesis is that adding extra functions, such as removing stop words for example can increase the accuracy of the model.

[3] The author of this article hypothesizes that models that detect fake news can be very beneficial for our society. The research aims to analyze the numerous factors of the pieces of fake news to detected them and apply a deep learning approach to address these issues. To do that, the authors use a random forest algorithm combined with natural language processing. This combination shows to be very successful, and the built model outperforms reaching an accuracy of 99.88%. The paper also focuses on the importance of fighting and detecting fake news, not only for each one as an individual but also the impact that these pieces of misinformation can have on our government and society.

[7] The author of this tutorial does a great and succinct job of explaining how to use the 4-step modeling pattern of scikit-learn along with the behavior of the logistic regression algorithm. This algorithm handles the prediction of digit labels based on images. To do that, the paper starts with applying Logistic Regression to a toy dataset. The goal of that is to start with a simpler example to make sure the concepts are well understood and clear. The following step is to apply Logistic Regression to a more realistic dataset.

[10] The authors of “Fake News Detection Using Logistic Regression” perform a pre-processing followed using logistic regression techniques to identify fake content in news articles. The preprocessing includes steamming and tokenizing for example, and both could be applied to my dataset before the classification. The article is well explained and provides a good definition of important terms such as what is steamming, tokenizing, and logistic regression classifier. The authors also perform a good job of explaining the performed steps to build the classifier.

[14] Basil Saji presents an introduction to the use of the K-Nearest Neighbour classification in Python, a tone of the simplest and more widely used classification algorithms. He

provides a complete and direct explanation of how the algorithm works and some of its applications. We are presented with a step-by-step process of how the algorithm works. He builds from an overall explanation of how it works to how to an example with code.

[17] The article wants to solve a text classification problem using a traditional machine learning tool, Google Colab, and a special encoder released by Google known as Universal Sentence Encoder as well as BERT(Pre-training of deep bidirectional transformers for language understanding). The process starts with explaining the dataset. Then, the author explains the libraries he uses, followed by some data exploration. He then proceeds to classify the text in two different ways: by BERT fine-tuning and Universal Sentence Encoder. I would highlight the importance of the Universal Sentence encoder because it is what I have used in this project. It allows us to transform a sentence into a vector in a special way.

[24] The authors highlight how the issue of reliability and credibility of information has risen and emerged as a global issue. Their hypothesis relies upon the fact that we can use an existing machine learning algorithm for the classification of fake news combined with extracting lexical and host-based features of urls. They conclude that the combination of exploring both textual and URL features improve the effectiveness of the methods to detect fake news.

[25] The authors perform the classification of fake news data related to the COVID-19 pandemic using deep learning approaches. They compare the use of machine learning and deep learning approaches on classifying the data and conclude that the deep learning approach shows to be more effective. They also provide a set of recommendations on how to select and perform a sentiment analysis model on Twitter data. The authors perform me the classification of fake news data related to the COVID-19 pandemic using deep learning approaches. They compare the use of machine learning and deep learning approaches on classifying the data and conclude that the deep learning approach shows to be more effective. They also provide a set of recommendation on how to select and perform sentiment analysis model on twitter data.

VI. Conclusions

It is important not only to preprocess the data, but also to understand the data and choose the correct preprocessing tools accordingly to the kind of data we are dealing with. In different kind of data, certain features have different importance, and we should evaluate how those features interfere in the meaning of the texts. When it comes to tweets related to covid, lowercasing stemming, lemmatization, and removing stop words improved the accuracy of most models while expanding contractions and removing punctuation had the opposite effect. Using this, I found that the best model was built using logistic regression and applying lowercasing, stemming, and removing stop words.

Before my experiments I expected that preprocessing the data would have a bigger impact in the accuracy scores of the models. Different than what I expected, the difference was small and some of the preprocessing tools even reduced the accuracy.

I also expected that the accuracy scores of the train portion of the data would be better than the one in the test dataset. Instead, most of the algorithms performed better in the new data.

The next important thing to notice is that depending how we split the test and train model, the accuracy of the models slightly varies. That shows the importance of running the code more than once to see how the model's average. That is even more important when we talk about comparing the preprocessed data's results with the non-preprocessed ones.

The challenge run the model more than once and calculate the averages was mostly because we must apply many different preprocessing tools and then create different models with each of the created data frames. I firs tried to run only some of the preprocessing tools in the data and later run my models more than once for each time I changed the data. That approach was not only not efficient, but also not structured. For that reason, I decided to create different data frames and apply different preprocessing tools to each. Next, I was able to use a loop and build models with each of the different possible data frames to further calculate the averages.

In future work, I plan to improve the model that best performed by exploring different preprocessing algorithms and using a bigger training dataset. I believe that studying this specific model and making changes on it, it is possible to improve its accuracy.

Another important step to improve the algorithm is to import datasets that contain more features about the tweets. Using features such as the urls, users, and date of publication for example allow us to do a further analysis on how there other factors have an influence in the tweets being fake or real.

I also plan to perform a sentiment analysis in the dataset. Since the dataset does not have labels on sentiments, I would be performing an unsupervised machine learning algorithm. After doing a sentiment analysis on the data, I plan to add use it to column into my dataset with either negative, neutral, or positive label obtained. Next, I will use the same algorithms in the dataset. Finally, I will compare the results from the first models and the ones with the new columns. I expect that adding another feature to the dataset will improve the accuracy of the models.

References

[1] A. Watson, "Topic: Fake news in the U.S.," *Statista*. [Online]. Available: https://www.statista.com/topics/3251/fake-news/. [Accessed: 24-Apr-2022].

[2] *Build a naive Bayes classifier with python*. Enlight. (n.d.). Retrieved February 25, 2022, from https://enlight.nyc/projects/build-a-naive-bayes-classifier

[3] Chauhan, T., & Palivela, H. (2021, November 30). *Optimization and improvement of fake news detection using Deep Learning Approaches for societal benefit*. International Journal of Information Management Data

Insights. Retrieved March 3, 2022, from https://www.sciencedirect.com/science/article/pii/S2667096821000446

[4] "Coronavirus," *World Health Organization*. [Online]. Available: https://www.who.int/health-topics/coronavirus#tab=tab_1. [Accessed: 24-Apr-2022].

[5] "Fake news," *FAKE NEWS | Significado, definição em Dicionário Inglês*. [Online]. Available: https://dictionary.cambridge.org/pt/dicionario/ingles/fake-news. [Accessed: 24-Apr-2022].

[6] *Fake news spreads faster than true news on Twitter-thanks to people, not bots*. Science. (n.d.). Retrieved April 12, 2022, from https://www.science.org/content/article/fake-news-spreads-faster-true-news-twitter-thanks-people-not-bots

[7] Galarnyk, M. (2021, November 17). *Logistic regression using python (scikit-learn)*. Medium. Retrieved March 3, 2022, from https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a

[8] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[9] -, G. L. T., By, -, Great Learning TeamGreat Learning's Blog covers the latest developments and innovations in technology that can be leveraged to build rewarding careers. You'll find career guides, Team, G. L., & Great Learning's Blog covers the latest developments and innovations in technology that can be leveraged to build rewarding careers. You'll find career guides. (2022, March 22). An introduction to bag of words (bow): What is bag of words? GreatLearning Blog: Free Resources what Matters to shape your Career! Retrieved April 20, 2022, from https://www.mygreatlearning.com/blog/bag-of-words/

[10] Hannah Kim and Young-Seob Jeong. 2019. Sentiment classification using Convolutional Neural Networks. (June 2019). Retrieved January 30, 2022 from https://www.mdpi.com/2076s-3417/9/11/2347/htm

[11] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[12] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[13] K. Elissa, "Title of paper if known," unpublished.

[14] "K Nearest Neighbor Classification Algorithm: Knn in Python." *Analytics Vidhya*, 22 Jan. 2021, https://www.analyticsvidhya.com/blog/2021/01/a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python/.

[15] *Must known techniques for text preprocessing in NLP*. Analytics Vidhya. (2021, June 14). Retrieved April 12, 2022, from https://www.analyticsvidhya.com/blog/2021/06/must-known-techniques-for-text-preprocessing-in-nlp/

[16] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[17] Paialunga, P. (2022, February 14). *Fake news detection with machine learning, using python*. Medium. Retrieved February 23, 2022, from https://towardsdatascience.com/fake-news-detection-with-machine-learning-using-python-3347d9899ad1

[18] R. Miller, "Data preprocessing: What is it and why is important," *CEOWORLD magazine*, 13-Dec-2019. [Online]. Available: https://ceoworld.biz/2019/12/13/data-preprocessing-what-is-it-and-why-is-important/. [Accessed: 24-Apr-2022].

[19] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[20] *Text preprocessing NLP: Text preprocessing in NLP with python codes*. Analytics Vidhya. (2021, June 29). Retrieved April 12, 2022, from https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/

[21] Thainá do Nascimento de Barcelos, Luíza Nepomuceno Muniz, Deborah Marinho Dantas, Dorival Fagundes Cotrim Junior, João Roberto Cavalcante, and Eduardo Faerstein. 2021. Análise de fake news Veiculadas Durante a pandemia de covid-19 no Brasil. (May 2021). Retrieved January 25, 2022 from https://iris.paho.org/handle/10665.2/53907 J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[22] Transfer learning NLP: Fine tune bert for text classification. Analytics Vidhya. (2020, July 26). Retrieved February 23, 2022, from https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/

[23] Twitter. (n.d.). *Our synthetic and manipulated media policy | twitter help*. Twitter. Retrieved April 12, 2022, from https://help.twitter.com/en/rules-and-policies/manipulated-media

[24] Valeria Mazzeo, Andrea Rapisarda, and Giovanni Giuffrida. 1AD. Detection of fake news on covid-19 on web search engines. (January 1AD). Retrieved January 25, 2022 from https://www.frontiersin.org/articles/10.3389/fphy.2021.685730/full

[25] Waqas Haider Bangyal et al. 2021. Detection of fake news text classification on covid-19 using Deep Learning Approaches. (November 2021). Retrieved January 29, 2022 from https://www.hindawi.com/journals/cmmm/2021/5514220/

[26] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].