Fake News Detection - Project Report

Ajay Rajput (c0871742), Rahul Rawal (c0871230), Dhru Prajapati (c0867085)

Instructor: Victoria Shtern

Class: CBD 3335

Abstract — In today's society, where information is easily accessible and available, the rise of fake news has become a serious worry. In the digital age, stopping the spread of fake news has become a critical challenge. In this project, we looked closely at the many approaches and tools used to spot fake news. As part of our investigation, we used representative datasets for training and testing, and data mining algorithms to evaluate news stories. We investigated various evaluation measures for gauging these algorithms' and approaches' efficacy. Our findings indicate that a combination of strategies, including the application of machine learning techniques and human knowledge, is necessary to accurately detect and prevent fake news.

Keywords **—** Fake News, Machine Learning, NLP, Text Classification, Data Preprocessing, Multinomial Naïve Bayes, Text Vectorization.

I. INTRODUCTION

Our project focuses on applying machine learning and natural language processing methods to identify bogus news. In recent years, the transmission of misleading information, political propaganda, and social unrest have all been related to the issue of fake news, which has grown in importance. By creating a model that can reliably distinguish between authentic and fraudulent news pieces, our initial attempts to solve this issue.

In order to do this, we loaded two datasets with both legitimate and bogus news stories and ran exploratory data analysis to learn more about the data. Then we labeled the datasets, concatenated them into a single dataset, and cleaned the data by removing null and duplicate values. In order to prepare the text for analysis, we also did feature engineering, which involved trimming the text of punctuation and stopwords and lemmatizing it. Overall, our experiment shows how machine learning and natural language processing methods can effectively identify fake news. The outcomes of our study can be used in real-world situations to spot fake news in the future and stop it from spreading.

II. EXPLORATORY DATA ANALYSIS AND DESCRIPTION

We used two datasets called "True.csv" and "Fake.csv" that contained data on the title, text, subject, and date of news stories to identify fake news. Each dataset had the shape (23481, 4), and upon closer examination, we discovered that no null values were present. However, we noticed that the "subject" column in the fake news dataset had six categories, whereas the actual news dataset only listed two. 'News', 'politics', 'Government News', 'left-news', 'US_News', and 'Middle-east' were the subjects in the fake news dataset, whereas 'politics news' and 'world news' were the subjects in the real news dataset.

We eliminated duplicate values from the datasets to assure data quality, giving the new shapes for true.csv and fake.csv of (21211, 4) and (23478, 4) respectively. After deleting them, we only discovered three duplicate values in the false news dataset. We were able to prepare the data for research and start looking into potential models for fake news detection by cleaning and processing these datasets. We assigned labels to the fake_news and true_news datasets in order to get the datasets ready for false news identification. We assigned a value of 0 to fake news and a value of 1 to truthful news. After all preprocessing, we merged both the datasets (fake_news.csv and True_news.csv) into one using the “concate()” function from pandas. The dataset was made by joining these two datasets together was named "final_news_df". The dataset that was produced when the index was reset had 44898 rows and 5 columns, including a new column called "label" that lets us tell the difference between fake and real

news. We were able to produce a dataset appropriate for analysis and machine learning by carrying out these processes.

III. FEATURE ENGINEERING

In the feature engineering phase, we utilized several libraries including string, tokenize, word_tokenize, nltk, and stopwords from the corpus. First, we performed a punctuation count and removed all punctuations from the text, including '!"#$%&'()*+,-./:;<=>?@[]^_`{|}~'. This process created a new column named "punct_count" that indicated the number of punctuations in each news item. Additionally, we performed a body length count for the texts in each news item, generating a new column that reported the number of characters in the text after removing the punctuation. These feature engineering steps allowed us to prepare the data for further analysis and modeling.

	text	subject	date	label	punct_count	text_body_length
	ething original? The draise off of Donald Trump ...	politics	Dec 23, 2015	0	35	1490
	website of the U.S. y, EPA.gov, is gett...	politicsNews	April 29, 2017	1	58	2158
	tly caused a bit of a ed to deliver his a...	News	August 7, 2016	0	36	1322
	ght being president one love him.The...	News	January 25, 2017	0	50	2572
	thorities in Vietnam ple from low-lying...	worldnews	December 25, 2017	1	56	2518
	nstream media was selves. From part...	Middle-east	November 23, 2016	0	228	7657

Figure 1 - New Columns Added

IV. DATA PREPROCESSING

For Data Preprocessing, several steps were taken to clean and transform the data into a suitable format for machine learning. Firstly, the Vectorizer library was imported, and the Lemmatizer was initialized. Secondly, irrelevant data fields such as the "Date" and "Subject" fields were removed from the dataset. Although no null values were found initially, some records with zero values in the "text_body_length" column were discovered and subsequently dropped. Finally, data cleaning was performed, which involved converting all text to lowercase, removing punctuations, tokenizing the text, removing stopwords, and lemmatizing the text. These steps were taken to ensure that the data was in a consistent and useful format for building the machine-learning model. Lastly, the tokenized list of text bodies was added to the datasets under the name "tokenized_text". That figure is depicted below:

IV. DATA VISUALIZATION

We were also able to gather some key factors from both datasets through histograms and Wordcloud. To compare the length of the text bodies in the two datasets, we first generated a histogram. The libraries required for charting and contrasting the text body lengths of the two datasets were first imported into the code. We were able to draw the conclusion from the below figure that fake news has a longer news text body than true news.

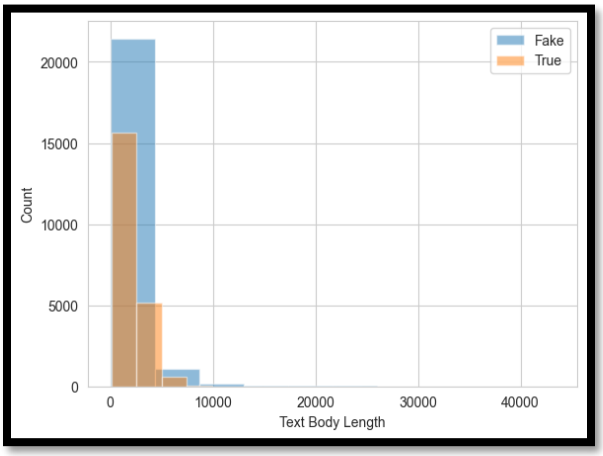

Figure 2 - Comparing text body lengths of Fake and True News

It would have been simpler to compare characteristics with various scales if we had normalized the data to scale it to a common range, often between 0 and 1. Normalization, though, might not be required or even acceptable for text data. This occurred because text data, like the body length of a news story, is frequently expressed as a count or a frequency. The underlying distribution of this data may have been affected by normalizing it, which would have made the data more difficult to analyze and less interesting. In our scenario, as long as the histogram accurately depicts the distribution of the data, a right-skewed histogram was not necessarily an issue. Right-skewed distributions are actually extremely typical for many different types of data.

Apart from this, we also plotted the Wordcloud of both datasets to quickly identify the most important words in this large text corpus. The larger the word size in the word cloud, the more frequently that word appears in news articles. Figures 3 and 4 represent the Wordcloud of both datasets.

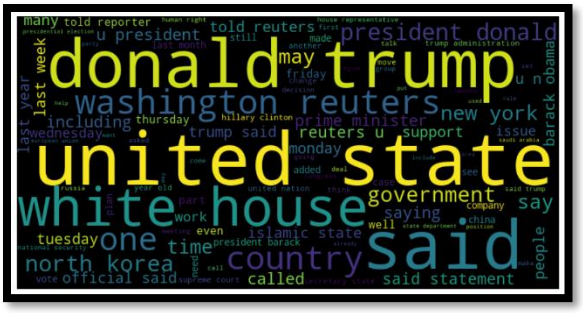

Figure 3 - Fake News Wordcloud

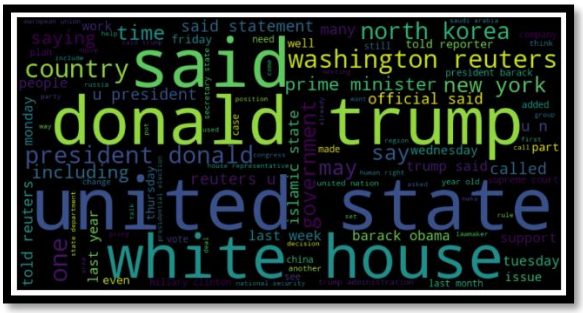

Figure 4 - True News Wordcloud

V. SPLITTING DATA (TRAIN & TEST)

We imported the train_test_split() function from sklearn.model_selection module to divide the data into train and test data.

The "final_news_df" data frame's "text" column was then given to the variable "X," and the "label" was given to the variable "Y." The data is divided into training and testing sets using the train_test_split() function. The 'X_train' (training text data), 'X_test' (testing text data), 'y_train' (training label data), and 'y_test' (testing label data) are the four sections that make up the 'X' and 'Y' text and label data. 80% of the data is utilized for training and 20% is used for testing in the split, which is done in a ratio of 80:20.

To guarantee that exactly the same random samples are created each time the code is executed, we set the random_state option to 42. The shape of the 'X' data's training and testing sets, i.e., 'X_train' and 'X_test', was subsequently outputted by the code. It was (35413,) (8854,).

VI. TEXT VECTORIZATION

In the next step of our project, we performed text vectorization on our data using the TF-IDF vectorizer. We imported the TfidfVectorizer class from the sklearn.feature_extraction.text module and initialized an object named "vectorizer". We set the maximum number of features to 50,000, turned off lowercase conversion since we already performed lowercase conversion during data cleaning, and set ngram_range as (1,2) to consider both unigrams and bigrams.

After initializing the vectorizer, we fit and transformed the training data X_train using the fit_transform() method of the vectorizer object and then converted it to an array using the toarray() method. Similarly, we transformed the test data X_test using the transform() method and then converted it to an array using the toarray() method. This resulted in two new arrays named X_train_vect and X_test_vect respectively. The dimensions of the transformed training data X_train_vect are (35413, 50000) and the dimensions of the transformed test data X_test_vect are (8854, 50000). After creating the vectorized data we created a new dataframe using pandas to store those vectorized data named "training_data" and "testing_data".

VII. TRAINING AND SAVING MODEL

Then our last process was that we initialized a Multinomial Naive Bayes model from the sklearn.naive_bayes library and importing the accuracy_score function from sklearn.metrics.

Next, we fitted our model on the training data using the fit function. The training_data here refers to the vectorized form of the text data that we obtained after applying TF-IDF vectorization. We also passed the y_train data which contains the labels for the training data.

Then, we made predictions on the testing data using the trained model and stored the predicted labels in y_pred. After that, we evaluated the accuracy of the model using the accuracy_score function and stored it in testdata_accuracy. We print out the test set accuracy to see how well the model performs on unseen data. Next, we made predictions on the training data using the same model and stored the predicted labels in y_train_pred. We then evaluated the accuracy of the model using the accuracy_score function and stored it in traindata_accuracy. After that we printed out the training set accuracy to see how well the model performs on the data it was trained on.

Finally, we used the score function of the model to get the accuracy score of the model on both the training and testing data and print out those scores as well. The accuracy of the test set was “0.954” and the accuracy of the training set was “0.958”. This gave us an idea of how well the model is generalizing to unseen data.

```
# Make predictions on the Testing data using the trained model
y_pred = model.predict(testing_data)
# Evaluate the accuracy of the model
testdata_accuracy = accuracy_score(y_test, y_pred)

print("Test set Accuracy:", testdata_accuracy)
```

```
Test set Accuracy: 0.9541450192003614
```

```
# Make predictions on the Training data using the trained model
y_train_pred = model.predict(training_data)
# Evaluate the accuracy of the model
traindata_accuracy = accuracy_score(y_train, y_train_pred)

print("Training set Accuracy:", traindata_accuracy)
```

```
Training set Accuracy: 0.9584051054697428
```

Figure 5 - Accuracy of Test and Training Set

The last few steps of this process were the following:

- We saved the model using the joblib.dump() function from the joblib library. The saved model is stored in the file named "model.pkl".
- Next, we created a new input data point using the data_clean() function and assigned it to the new_data variable.
- Then we loaded our saved model from the file model.pkl using the joblib.load() function and assigned it to the model variable.
- After that, we transformed our new input data using the vectorizer.transform() function and passed it to the model.predict() function to make predictions on the new input data. The predicted labels are stored in the predicted_labels variable.
- Finally, we checked the predicted label and printed the output message accordingly, which tells us whether the input news is fake or true. If the predicted label is 0, it is classified as fake news, otherwise, it is classified as true news.

Below given figure is a demo of our fake news detector.

```
new_data = data_clean(str("Dr. Manmohan singh is dead now."))
```

```
model = joblib.load('model.pkl')
new_vect_data = vectorizer.transform([new_data]).toarray()
# df = pd.DataFrame(new_vect_data, vectorizer.get_feature_names_out)
predicted_labels = model.predict(new_vect_data)
print(predicted_labels)
if predicted_labels[0] == 0:
    print("This is a Fake News")
else:
    print("This is a True News")
```

```
[0]
This is a Fake News
```

Figure 6 - Prediction Demo

VIII. SUMMARIZATION

As we previously mentioned, our objective was to develop a classification model that could be used to scan news features like headlines and phrases for fake news. We performed the preprocessing procedures, feature selection, and training of a Multinomial Naive Bayes model while gradually but progressively embedding the model in Jupyter - Notebook.

In conclusion, a training and testing dataset was available. Both underwent preprocessing, transformation, and feature selection. The model was built, trained, tested, and verified after the noise removal. Finally, we were able to verify the news that had been questioned and was able to determine which news articles were true and which were fake.

IX. CONCLUSION

To conclude, we used a variety of methods, including preprocessing, feature selection, and Multinomial Naive Bayes model training. In order to enhance the model's performance, we also gradually integrated it into Jupyter-Notebook.

We were able to determine the veracity of the disputed news after careful analysis and correctly label news stories as either true or false. This research shows how machine learning models may be used to identify fake news and emphasizes the value of feature selection and preprocessing in enhancing model accuracy.

Overall, the proposed model offers a useful tool for preventing the spread of false information and fake news, supporting the accuracy of news reporting, and encouraging the use of informed judgment.