Department of Computer Science
&
Engineering

CSE422 Project Report

Spam Email Detection

ATHAR NOOR MOHAMMAD RAFEE
ID: 20101396
LAB SECTION: 09
GROUP: 08

supervised by
Ms. Afia Fairoose Abedin & Ms. Sumaiya Akter

Contents

1 INTRODUCTION 3

2 LEARNING DATA 3

3 DATA PREPROCESSING 3

4 Experimental Result 5

4.1 Results 5

4.2 Analysis 7

5 Conclusion 8

1 INTRODUCTION

In, today's ever evolving modern and busy world email is a simple yet very important source of communication between various groups of people due to personal, business, corporate and government use-cause reasons. Over the past few years, the usages of email have greatly increased as the internet has revolutionized many parts of the world. Unfortunately, as the number of email users increases, **SPAM**(also known as junk email) email targeting general users also increases. These **SPAM** emails, which are being sent to numerous recipients, are identical in nature and being very annoying, they pose various security vulnerabilities to the client's device if not handled properly. Apart from that, these **SPAM** emails take away a lot of precious time from the user and cause financial loss to corporations and businesses. To prevent that and make efficient use of emails, in this project I used various text mining strategies to extract meaningful information from emails text and used various machine learning algorithms to figure out which one is optimal to use for spam email detection in common day-to-day life scenarios.

2 LEARNING DATA

The dataset that I used for this project from kaggle[1] is called ***spam email detection dataset***, and it's a raw. The initial data set of was **5730** rows and **110** columns, and many of the columns were not necessary for further use cases. The first thing I had to do was preprocess the dataset and took it to a usable state for further text processing and tokenization/word frequency count.

3 DATA PREPROCESSING

As the dataset had a lot of `null` columns and cells, first resolved that issue by dropping the **null** columns. After then moved on the `null/NaN` cell and took care of it as well by dropping the null/NaN rows. Did the necessary encoding(binary label encoding where 0 means non-spam and 1 means spam) for the label and got rid of duplicates values from rows. After performing these series of data-preprocessing steps, I ended up with **5693** rows and **2** columns. However, converting the plain raw email texts into features was yet to be done. To achieve this, I used two slightly different text `vectorizer/tokenizer`*(count the frequency of words in texts ignoring some common stop words that have no significant impact on the meaning of the sentences but appear frequently)*[2] namely `CountVectorizer()` and `TfidfVectorizer()` to convert them into features so that later I can train and test various models on them. After preprocessing the dataset, the dataset contained around **37000**) features or words*(tokens)* found by `CountVectorizer()` and `TfidfVectorizer()`. To get an idea about each set of features, I used `wordcloud` to generate visual representations of the most frequently occurring words shown in the below Figure 1 and Figure 2.

Figure 1: Frequent Words Found Using count vectorizer

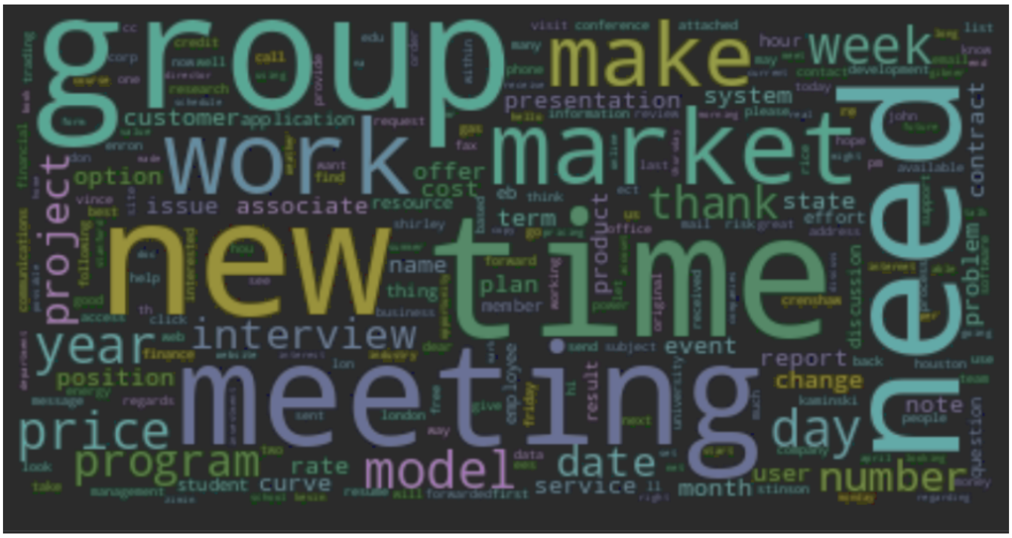

Figure 2: Frequent Words Found Using tfidf vectorizer

4 Experimental Result

I used the training data obtained from both of the vectorizers(`CountVectorizer()` & `TfidfVectorizer()`) to train a variety of models using training data and then used testing data to measure various matrices regarding that particular model, most importantly the **accuracy**. In order to obtain different metrics regarding a particular model, I used the `classification_report()` method imported from `sklearn.metrics`.

4.1 Results

Below, I am attaching all the metrics found by different learning models.

Logistic Regression using `CountVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.99	0.99	0.99	843
SPAM	0.98	0.97	0.98	296
accuracy			0.99	1139

Logistic Regression using `TfidfVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.97	0.99	0.98	843
SPAM	0.96	0.91	0.93	296
accuracy			0.96	1139

MultinomialNB using `CountVectorizer`

	precision	recall	f1-score	support
NON-SPAM	1.00	0.99	0.99	843
SPAM	0.97	0.99	0.98	296
accuracy			0.99	1139

MultinomialNB using `TfidfVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.83	1.00	0.91	843
SPAM	1.00	0.43	0.60	296
accuracy			0.85	1139

For some reason there is a lot of difference in terms of accuracy between `Count & Tfidf` Vectorizer when using Multinomial Naive Bayes Model.

BernoulliNB using `CountVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.98	0.99	0.99	843
SPAM	0.97	0.95	0.96	296
accuracy			0.98	1139

BernoulliNB using `TfidfVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.98	0.99	0.99	843
SPAM	0.97	0.95	0.96	296
accuracy			0.98	1139

GaussianNB using `CountVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.96	0.99	0.97	843
SPAM	0.97	0.87	0.91	296
accuracy			0.96	1139

GaussianNB using `TfidfVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.95	0.99	0.97	843
SPAM	0.98	0.86	0.92	296
accuracy			0.96	1139

DecisionTreeClassifier using `CountVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.96	0.97	0.97	843
SPAM	0.92	0.89	0.91	296
accuracy			0.95	1139

DecisionTreeClassifier using `TfidfVectorizer`

	precision	recall	f1-score	support
NON-SPAM	0.96	0.97	0.97	843
SPAM	0.92	0.89	0.91	296
accuracy			0.95	1139

4.2 Analysis

All of the used models have more or less the same outcome excluding Multinomial Naive Bayes 4.1 model where we see a significant difference between two vectorizers. Below is the accuracy barchart using `CountVectorizer()`3 and for `TfidfVectorizer()`4.

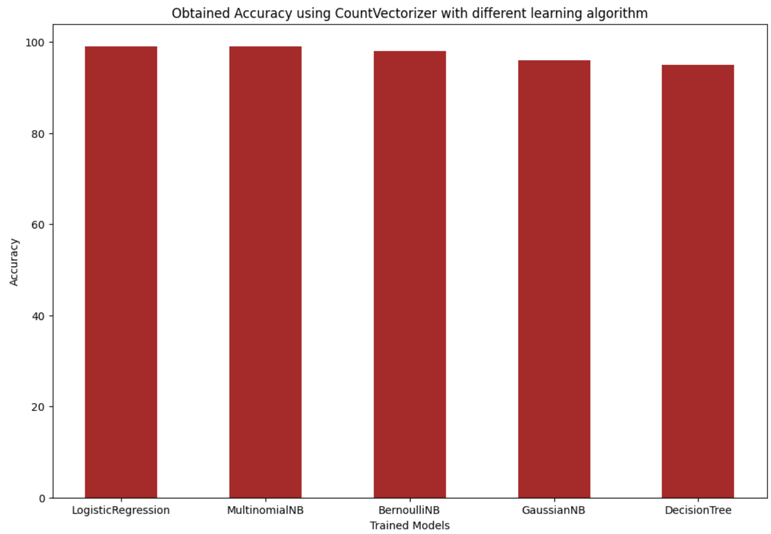

Figure 3: Accuracy using count vectorizer for different Models

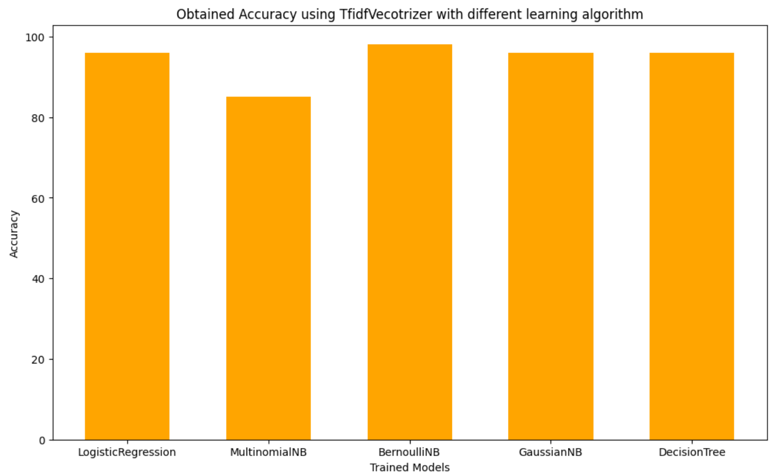

Figure 4: Accuracy using tfidf vectorizer for different Models

From the above graph we can see, except for one model, both tokenizers are equivalent and they are quite accurate in terms of detection spam and non-spam email regardless of the model.

5 Conclusion

In the end, the use of machine learning techniques for spam email detection has proven to be an effective way to automatically filter out unwanted messages. And to do so, we don't need to use advanced methods such as 10-level deep neural networks; instead, a simple Logistic Regression is way more than enough to detect 98% of the time. By training a model on a large dataset of labeled emails, we were able to achieve high levels of accuracy in predicting whether a given email was spam or not. The ability to accurately identify and remove spam emails not only helps to protect individuals from potential scams and phishing attempts, but also helps to reduce the overall amount of spam that is sent and received, making the internet a safer and more efficient place. Overall, this project has demonstrated the power of machine learning in the fight against spam email.

References

[1] S. MART, “spam email detection dataset,” September 2020. [Online]. Available: https://www.kaggle.com/datasets/studymart/spam-email-detection-dataset

[2] Dec 2020. [Online]. Available: https://pianalytix.com/countvectorizer-in-nlp/#:~:text=CountVectorizer%20means%20breaking%20down%20a