Email Classification HAM or SPAM

Kris Ghimire, Rajesh Satluri, Suchismita Moharana

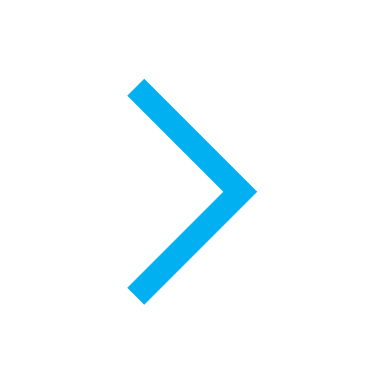
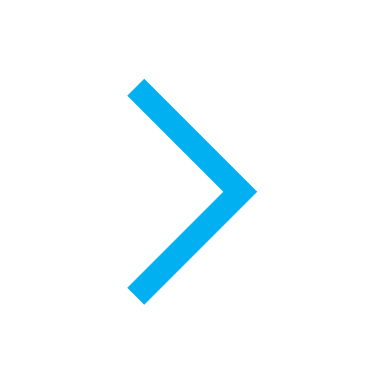
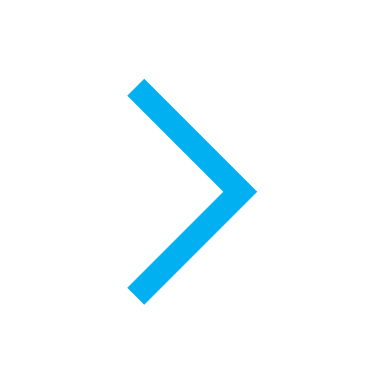
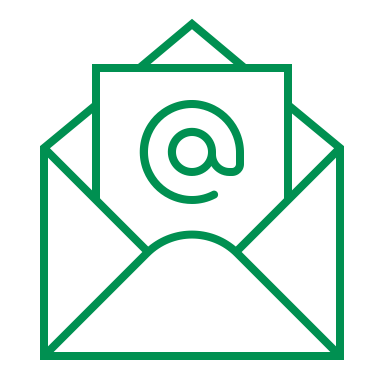
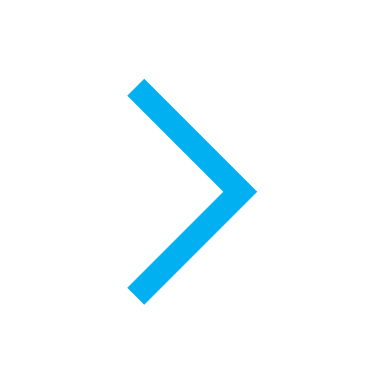
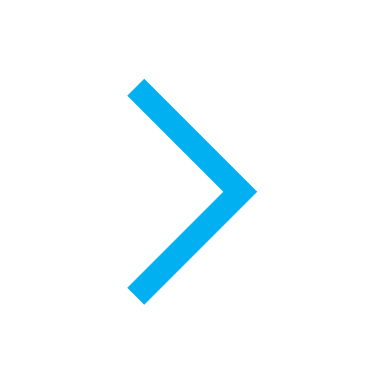
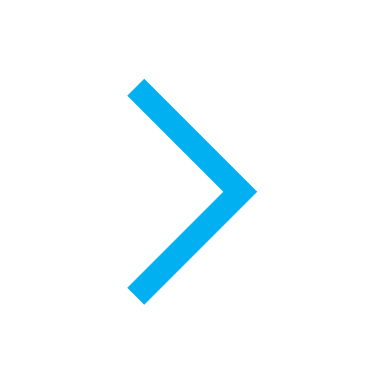
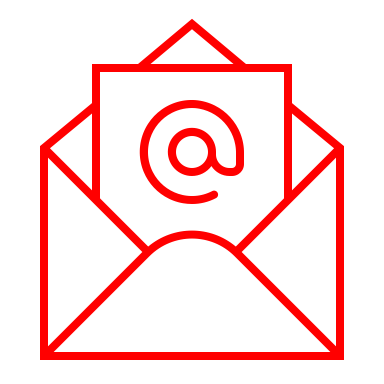
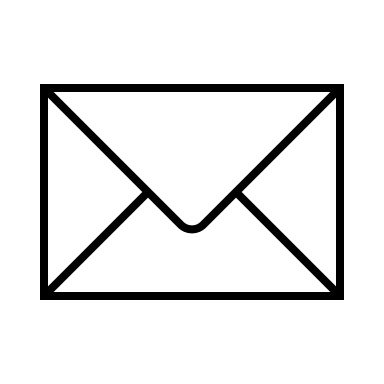
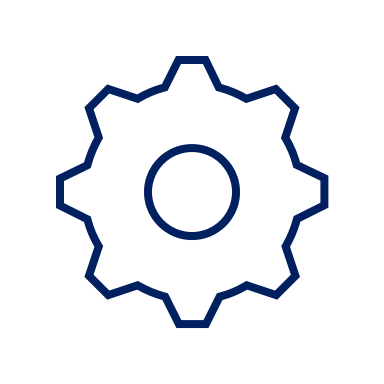
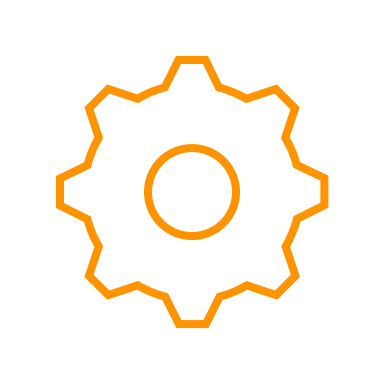
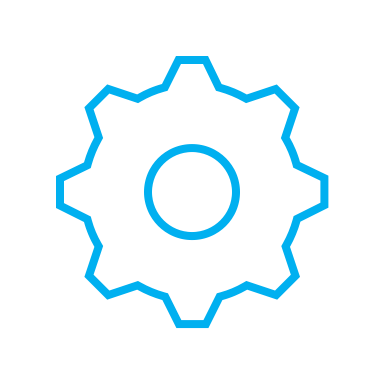
June 6, 2021

**Abstract**

In this case study, researchers are going to build a classification model to classify emails into ham and spam

**1. Introduction:**

If you login to your Gmail or any mailbox you will quickly notice that more than half of your emails are unwanted and often unsolicited junk that is filling up storage. Often time these emails are phishing mail intended to steal information. These annoying and useless emails are called spam. Filtering out only the email we want and discarding unnecessary spam is not an easy task. It is excessively time consuming and almost impossible to manually identify only the ham out of thousands of emails we receive. To overcome this challenge and automate the process of identifying ham and spam, in this research paper we will use NLP pipeline to build a machine learning classifier to classify ham and spam. The figure below (figure 1) shows the general pipeline of how the classification process works.



ML classifier

Incoming mail

Ham

Spam

## Figure 1

## 2. Methods

**Dataset:**

Table

Description automatically generated with medium confidence

Table 1

The dataset provided for this case study is slightly different than the usual format. It would be much easier if we always get dataset in csv format. However, in reality, data comes in different forms. One of the biggest challenge of Data Scientist is to get the data together. The data were provided in five different folders easy\_ham, easy\_ham2, hard\_ham, spam2 and spam, all of which contained multiple files of raw text and html format emails. The first step was to get only the text content from all the given files. We used BeautifulSoup and HTMLparser and email package to extract only the contents from the file. All text contents from each file were then transformed into one dataframe (Table 1). Table below shows the size of the dataframe, training set, test set and the features that were used to build classifier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | Training set | Test set | Features | Target |
| 9043 | 123123123 | 1231 | 12312 | 2 |

Out of all email files, there are 17 of them which we couldn’t read due to encoding issue. It is very small volume as compared to 9000+ and we are ignoring these 17 emails. There are some duplicates in the dataset, we decided to build two models, one including duplicates and another without duplicates. We didn’t see much changes in either of them in term of model performance metrics.

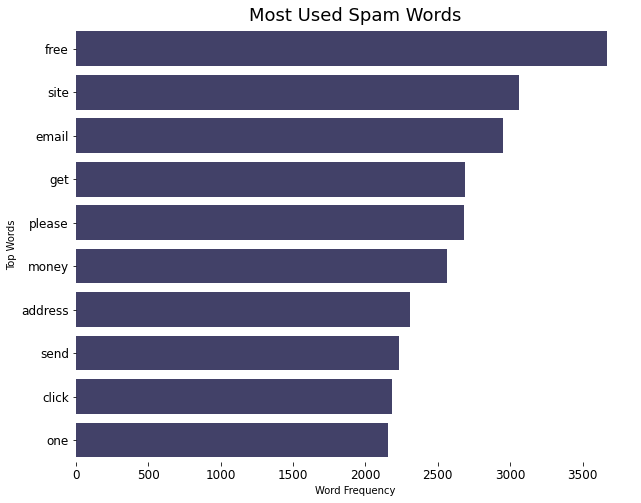
Upon visualizing target features, (figure 2) we noticed unequal distribution of classes, ham and spam which is quite common and always a challenge in classification problems. As with most machine learning algorithms, uneven distribution of class ratio in Naïve Bayes classifier could leads to an inaccurate estimate of class prior which could then potentially decrease the predictive performance.

A picture containing logo

Description automatically generated

Figure 2

**Data Exploration:** Text cleaning also known as text normalizing is crucial before exploration the data in NLP pipeline. We applied majority of text normalization process to extract meaningful information out of text. Some of which includes, case conversions, removing stops words, removing characters that are not alpha numeric, lemmatization, tokenization etc. We wanted to explore the words that are most frequently used in spam email. As expected with all spam email text, we can notice in the figure 3 below that a word ‘FREE’ is the most used word followed by other common spam words such as please, money, get, click, site, one etc. as top ten frequent used spam words.



*Figure 3*

Visualizing not spam (ham) will not provide any insight to us as we are mostly focusing on understanding about spam email and how can eliminate receiving spam junk. However, just for the exploratory purpose we made word cloud of ham text as shown in figure 4 below.

A picture containing text

Description automatically generated

*Figure 4*

**Independent Assumption:** One of the key assumptions of Naïve Bayes classifier which is what we are focusing on using for email classification is that it naively makes an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

**Bag of words feature extraction**: Machine learning algorithms need numeric data to perform calculation. Text data will mean nothing to a Machine. There are various methods of feature extraction. A most popular and simple to understand is a method of feature extraction using bag-of-words (BOW). A BOW is a representation of text data that describes the occurrence of words within word corpus. It is called bag because there is no order of occurrence of words. We will use sklearn CountVectorizer which implements both tokenization and occurrence counting in a single class.

**Train Test Split:** While dataset is split into training and test set (80/20 split) to keep test data separate and to perform modeling only on training set.

**NLP pipelines:** We will follow NLP pipeline to build the Navie Bayes classification. Figure 5.

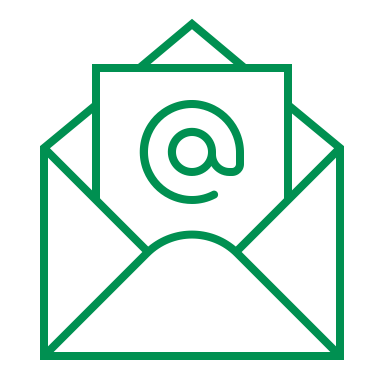
**Training**

CountVectorization

Text pre-process/normalization:

Simple tokenization

Lowercase, stemming and stop words removal



Exploratory Analysis: Most common words, word clouds

Incoming email

Cross-validation

Metric Evaluation

-Accuracy

-Precision

-Recall

Text pre-process/normalization

Stop words removal

Test Set Prediction

Features

Feature extraction

Countvectorization

Learned Classifier Model

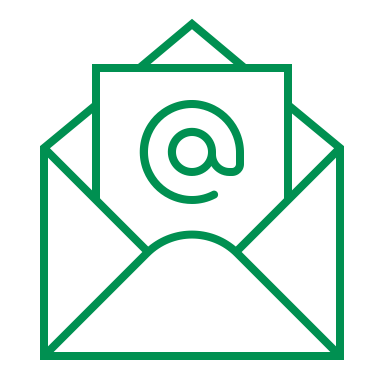
Machine Learning classifiers (NB)

Feature extraction

Features

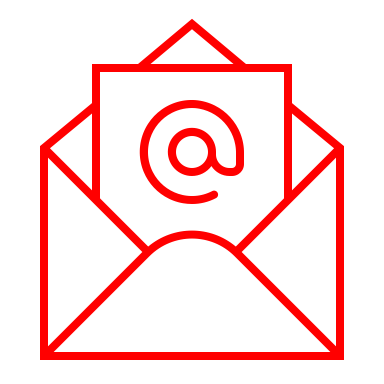
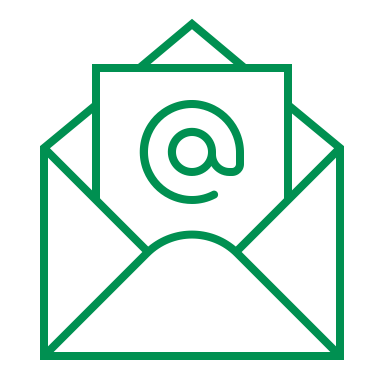
**Prediction**

Incoming emails



Recommended Label

*Figure 5*

Spam Not Spam

**Imbalance target features:** Dataset had imbalanced target features. Modeling on imbalanced target features would give evaluation metrics that are not accurate. We will implement SMOTE technique to balance the dataset. We will build both models, with unbalanced data as well as balanced data using SMOTE.

**Cross-Validation**: We will use StratifiedKFold cross-validation with k=5 because with stratifiedKFold, the class distribution in the dataset is preserved in the training and test splits. Shuffle is set to true so that the splitting will be random.

**Naïve Bayes (NB) Classification:** Naïve Bayes is slightly different from other Machine Learning (ML) algorithms. With NB algorithms we apply statistical distributions to Bayes rule (Figure 6) and use its mathematics to produce results.

Likelihood or evidence

Class Prior Probability or initial believes

P(A|B) =

Predictor Prior Probability or Normalizing factor

Posterior or updated believes

Figure 6: Bayes Rule

We would read above NB formula as the probability that A occurs give B is the probability that B occurs given A times the probability of occurring A divided by the probability of occurring B.

To explain NB in simple term, what we have is, we have our prior or old believes (probability distribution), when the new evidence occurs, we update the believes. Basically, we take that new evidence and multiply it by our prior believes to get our posterior or updated beliefs.

**3. Result:**

Models were build using several approaches. First, we built model 1 on the data as it is without balancing. The 5- fold cross-validation gave mean accuracy of about 95.9 % and standard deviation of 0.0076. Model did a good job on both majority as well as minority class as verified by the Roc curve as well as precision and recall score (Figure 7, 8, 9).

A picture containing logo

Description automatically generatedChart, line chart

Description automatically generated

*Figure 8. ROC Curve*

*Figure 7 Confusion metrics*

Table

Description automatically generated

*Figure 9. Classification Report*

We built second model, model -2 using ComplementNB algorithm. ComplementNB, according to sklearn documentation is particularly suited for imbalanced data sets. We didn’t see much difference in model evaluation metrics. Figure 5,6

A picture containing logo

Description automatically generatedChart, line chart

Description automatically generated

*Figure 11. ROC Curve*

*Figure 10 Confusion metrics*

**4. Conclusion:**