## DS7333 - Quantifying the World -Case Study 3

## By: Scott Frazier

# Introduction

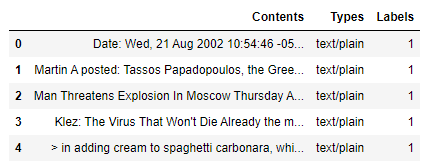
Classification and clustering are both very important machine learning methods that are used at an astounding rate in modern day life to the point where we as members of society do not even realize we are using technologies such as these. An example of one of these technologies could be the use of clustering techniques to identify when spam emails are being sent to your e-mail address, and then separately categorizing those emails as “spam.” Given how frequently user’s data is being sold between companies, it is no wonder how the world of automation has led us to where we are today in terms of how much spam is being blocked by our email providers. The purpose of this report is to create a spam classifier using naive Bayes and clustering. This report will also include the process of taking raw email inputs, and converting them to a useable format in which we can perform data analysis, and utilize our machine learning methods.

# Data Preparation

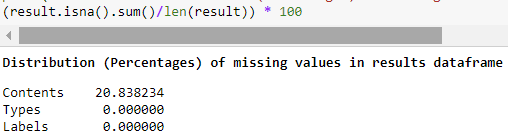
Starting off, we were given five folders which consisted of three folders with regular email strings, and the remaining two contained spam messages (which is what we are building a classifier to identify; if a message is considered spam or not). Each of these folders contained text files which we would need to access through Python libraries such as the os, email, path, and pandas.

It is worth mentioning that upon first glance of the data, it was clear we would need to create our own dataset and there were a few elements of the text files that would become our main variables. Using the libraries dealing with file directories, we were able to grab the files we needed for the analysis, and loop through the thousands of text files and organize them into different categories of file type (this is touched on more in the EDA section).

Once this was completed, we were then able to move on to the remaining values in our data sources. The remaining variables we sought to create included the contents of each email, and the label associated with each email. Fortunately for us, the emails were already split up into folders which classified them as spam or ham (not spam). Each variable was ultimately added to its own data frame and then all three data frames were joined together resulting in a final dataset, which consisted of a pandas data frame with the following columns: Contents, Types, and Labels. The Labels column is what we are using to identify whether an email’s contents are consistent with that of spam (denoted by a 0) or not spam (denoted by a 1). The values of our Labels column (0, or 1) were created by mapping the value to the data frame based on the folder’s spam or ham (not spam) value.



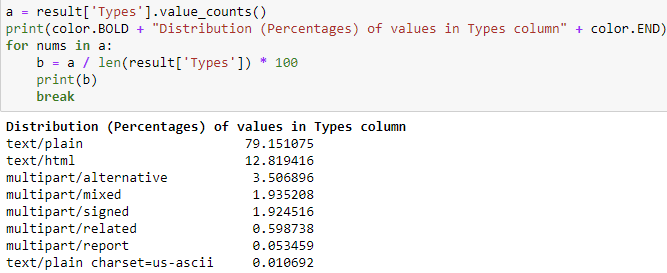
Later, while completing the project, it was discovered that many values (approximately 21%) were not populated in the Contents section.



It is unclear if this is because the files were originally missing data, or if it is a result of the method chosen to parse the emails. To deal with these missing values, we first exported the ***results*** data frame (which was the most recent version of the updated data frame with all columns joined together) to Excel and it was decided that we would randomly select a content value that was not missing, which contained the same Type, and Label as the one which was missing. We chose to do this because the emails identified as spam or not spam seemed fairly identical to one another respectively, and the main goal of this is to get an accurate reading of whether or not we can predict the correct label. Therefore, by filling in these values with the method described above instead of simply deleting them, we are able to work with more data. This analysis and imputation, as well as converting the contents column to all lowercase letters for the sake of model performance were conducted in Microsoft Excel.

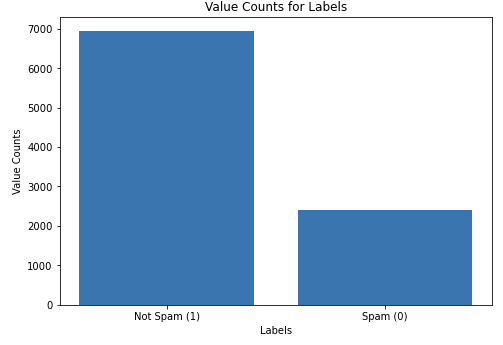
# Exploratory Data Analysis (EDA)

We are now able to move on to the EDA portion of the project. To begin, we want to get a better idea of the distribution of file types in the data set. Using a for loop, the distributions (in the form of percentages for each file type) can be seen below:



Next, we want to see a breakdown of the classes in the Labels column we created. To do this, we will utilize a similar for loop to the one above, but modify it for our label values; the results can be seen below. Looking at the results, we can see a fairly large difference in the balance of our data, seeming as though the emails which are not spam (roughly 75% of the values) are heavily outweighing the emails which are spam (roughly 25% of the values).





These imbalances are obviously not ideal, but they could be worse. Class balancing methods are not implemented with an exact science involved, but this imbalance seems *ok* for now so we will leave it as is and revisit this issue later if our outcomes suffer as a result.

# Modeling

Now we can begin the modeling portion of our project, specifically through the use of a clustering algorithm (in this case we will be using Support Vector Machines), as well as Multinomial Naïve Bayes to compare. We should also let it be known that we chose to split the data up into four sections which included a training and test set for both our predictor variables (primarily the contents of the email) and the target (our label). It is worth noting that the main differences between SVM and Multinomial Naïve Bayes are that with SVM we are looking at the interactions between our variables, and MultinomialNB naively assumes that each of them are independent of one another.

Next, we utilized the TF-IDF vectorizer to help with the preparation of our data for our model. We utilized this tool on the xtrain split of data, which then allowed us to use the newly transformed data in our models.

We also chose to use a C value of 1000 in the SVM model because we did not see much of a change in our outcomes when testing C values of 1000, 100, 10 or 1. However, we did see a change in the output when the parameter value was changed to .1, albeit not in a direction that we would have wanted (this being because we saw an increase in the number of incorrectly identified predictions).

# Results

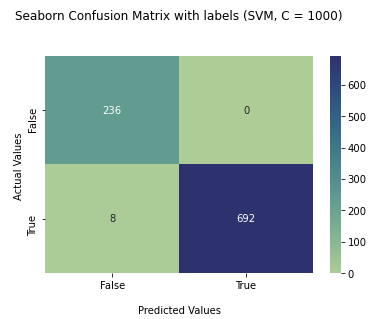
SVM

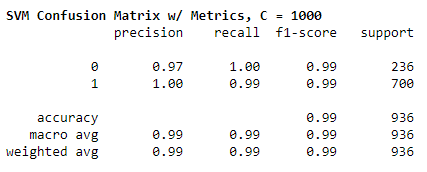
Here is a breakdown of the SVM confusion matrix results:

SVM C parameter value tuning:

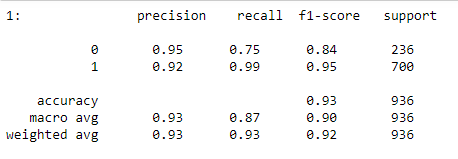
C = 1000, 100 C = 10, 1 C = .1

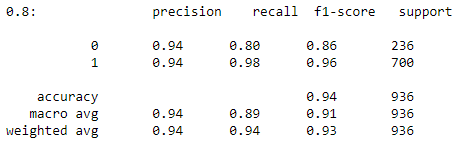
  

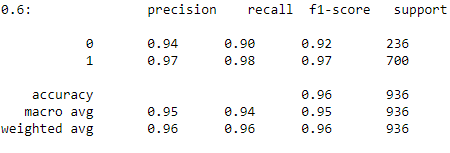


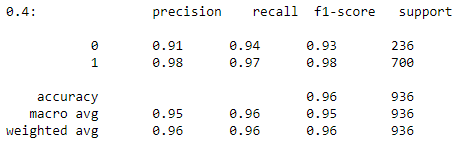


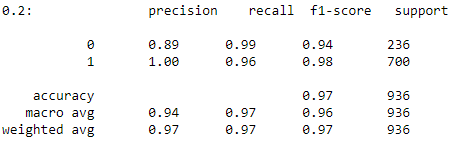
Multinomial Naïve Bayes (with different levels of alpha)











When looking at the results we see above for both our clustering algorithm, and the Naïve Bayes model; we see a variety of results. SVM seemed to do very well for the majority part, with an accuracy score of around 99%, which is very good. It also did very well across the board, with a bit less deviation from its output unlike MultinomialNB.

On the other hand, we have the results for Multinomial Naïve Bayes which are a bit of a mixed bag, even though all of the final scores are not bad by any means. When changing the alpha levels, we see a spectrum of results in terms of precision, recall, f1-score, as well as our overall accuracy for each alpha level. When looking at the accuracy scoring metric, we see that an alpha level of 0.2 gave us the highest value for accuracy with 0.97.

# Conclusion

Overall, both of these models performed very well for the task at hand. Clustering proved to be a very effective tool to produce accurate results when dealing with tasks such as spam detection like we did here. If I were to pick model to use again in the future out of these two methods, I would go with Multinomial Naïve Bayes for this data, as it is a generative model and we do not have a lot of data to work with; therefore, this model seems like the best for this data. This is not to say that SVM is a bad choice though, our results were still rather good all things considered.

# Appendix

Here are the imports which were used in the course of completing this assignment:

**Grabbing files and email parsing**

1. from email.message import EmailMessage
2. import email
3. from pathlib import Path
4. import os
5. import email
6. import pandas as pd

**Graphing**

1. import matplotlib.pyplot as plt
2. import numpy as np
3. import seaborn as sns

**Support Vector Machine model (SVM)**

1. from sklearn.model\_selection import train\_test\_split
2. from sklearn import svm
3. from sklearn.metrics import confusion\_matrix as cm
4. from sklearn.metrics import classification\_report as cr

**Multinomial Naïve Bayes**

1. from sklearn.naive\_bayes import MultinomialNB