**Introduction**

Spam. This is a word that many, if not all people in modern day know, it has been a pressing issue for a long time and now more than ever with the advancements of technology as well as the widespread use of email. Spam is defined as “Irrelevant or unsolicited messages sent over the Internet, typically to a large number of users, for the purposes of advertising, phishing, spreading malware, etc.” (Lexico) and this is a growing problem for email providers as this threatens the integrity and security of their customers. One click on a link from one of these spam emails could set loose a virus onto the victims device which can have terrible consequences at worst, or it could waste someone’s time at best.

45% of all emails sent per day globally are spam and some research companies claim that this is in reality 73% instead (Spamlaws, 2020). In July of 2019 there were reportedly roughly 17.3 million malicious attachments in emails (Maria Vergelis, 2019) and this is why many email providers use spam filters to protect their users from malicious intent as well as to save companies time and money, as it was estimated that spam costs companies an estimate of $20.5 billion annually in decreased productivity (Spamlaws, 2020).

Spam doesn’t only cause problems with the spam emails themselves, because people use spam filters to filter out theses emails not all of these filters have a high precision. This means that a lot of the time these filters will quarantine what’s called a false positive where it flags a non-spam email as spam resulting in many people not getting possible crucial emails. This was a problem for a South African automotive tyre and industrial rubber production company named Dunlop where it was said at one point their IT department were spending up to an hour and a half each day sifting through some 12,000 emails to find false positives that had been caught in a so called ‘spam trap’ (Mimecast, 2003-2020). This Company enlisted the help of Mimecast which is an email protection service which uses spam filters that use a cloud based Multi-Layer detection algorithms to filter through millions of emails on a daily basis. The company boasts an accuracy 99 percent for stopping spam emails with 0.0001% of these emails being false positives, which then in turn improves employee productivity (Mimecast, 2003-2020).

From the above points I have made you can clearly tell that spam is a big problem in the corporate world as well as personal life and therefore something needs to be done about it. I have already alluded to a way of dealing with spam by using spam filters such as the one offered at Mimecast. I will be planning on doing something similar by using algorithms and an ensemble method to sort through a dataset and filter out spam and non-spam emails.

**Proposed Method**

Spam filters use algorithms to try and detect whether the email that you have received is a real safe email or a spam email, these algorithms check every email that is received and compares them to previous emails received both real and spam and identifies similarities. However this process doesn’t always give you 100 percent accuracy if you are only using one algorithm and this means that you will still sometimes get spam come into your inbox, this is why I propose to use multiple algorithms together as this gives you a higher chance to correctly identify spam.

To detect spam in a higher degree of accuracy I will be combining 3 algorithms using a simple stacking classifier, what this does is runs each algorithm individually and then it runs all of the algorithms together comparing them to find the best possible iteration.

The 3 algorithms that I will be using is Decision Tree, KNN and Stochastic Gradient Descent (SGD) algorithms and the previously mentioned Simple Stacking CV Classification ensemble method. I got the algorithms from Scikit learn and I used the Stacking Classifier from mlxtend.

To properly test and run these algorithms I need a dataset, which was provided for me. This dataset contains 4600 entries with statistical with some being classified as spam and others being classified as not spam. Using the algorithms I have chosen I will be aiming to get the most accurate result of classifying whether or not an email is spam or not spam as well as to minimise the amount of false positives and false negatives. False positives are when the algorithm flags a not spam email as a spam email, and false negatives are where spam emails are flagged as not spam, so ideally I want my algorithm to have no false positives so that the users will get their emails and I want no false negatives as I don’t want any spam emails to get through. However I cannot guarantee that there will be none of either one; I can only hope that my algorithm has as few false negatives and false positives as possible.

I have chosen to use Stochastic Gradient Descent as one of my algorithms as it has been used for machine learning for a long time, it is reliable whilst also being simple and effective and has been used in large scale machine learning. Stochastic Gradient Descent also has a lot of opportunities to tune the code in the event of problems as well as being very efficient. And for my ensemble method I will be using a Simple Stacking Classifier from mlxtend, I chose this method as it is a relatively simple one which is well suited for the 3 algorithms that I chose.

Because the ensemble method combines all of the algorithms I believe that it will be more accurate than any of the algorithms on their own however the single algorithms will be more consistent than the ensemble due to the fact that a small change in any of the algorithms used could have a major impact on the ensemble.

**Pseudo code of my algorithm program**

Imports

Set X and Y values in relation to dataset

Decision Tree = Decision Tree algorithm  
Prediction = decision tree.train ( data for training)  
Print (“accuracy of Decision tree” + scores( testing data compared with prediction)  
Crossval = shufflesplit( no. of splits = 10, size of the test = 0.1 / 10%)  
Cross validation (Decision Tree, Data for training, Data for testing, Crossval)

KNN = KNN algorithm  
Prediction = KNN.train ( data for training)  
Print ( “accuracy of KNN” + scores( testing data compared with prediction )  
Crossval = shufflesplit( no. of splits = 10, size of the test = 0.1 / 10%)  
Cross validation ( KNN, Data for training, Data for testing, Crossval)

Stochastic = Stochastic Gradient Descent algorithm  
Prediction = Stochastic.train ( data for training)  
Print ( “accuracy of Stochastic” + scores( testing data compared with prediction)  
Crossval = shufflesplit( no. of splits = 10, size of the test = 0.1 / 10%)  
Cross validation ( Stochastic, Data for training, Data for testing, Crossval)

Ensemble = Simple stacking CV Classification ( classifiers (Clfs) = Decision Tree, KNN , Stochastic Gradient Descent, logistic regression)  
 Scores = Cross validation ( Clfs, Data for Training, Data for Testing, Cross validation amount = 10, scoring = “accuracy”)  
Print ( “Accuracy of ensemble” + results)

Ensemble = Simple stacking CV Classification ( classifiers = Decision Tree, KNN , Stochastic Gradient Descent, logistic regression)  
 Scores = Cross validation ( Clfs, Data for Training, Data for Testing, Cross validation amount = 10, scoring = “precision”)  
Print ( “Precision of ensemble” + results)

**Evaluation**

For my evaluation I will be using accuracy and precision as my two metrics. These two metrics tell me how correct my classifications of spam and not spam are and the measure of how many false positive I got respectively. These two metrics together allows me to see exactly how well my algorithms are working. I tested my algorithm 5 times each separately from each other using 10-fold cross validation which can be seen below.

The results I got from the 5 tests I did I plotted into a bar chart, I tried to put them into different charts and graphs to see if they conveyed the data better, but I decided on a bar chart as I felt that it was the easiest to read the data of it.

By looking at my graphs I can clearly see that my ensemble method (Simple Stacking Classifier) has a higher accuracy and higher precision than my Stochastic Gradient Descent (above) as well as my Decision Tree (below). However my Simple Stacking Classifier is not as consistent as my Decision Tree algorithm which maintains an accuracy of 0.87 and a precision of 0.88 for all apart from one of the iterations whereas my ensemble methods’ accuracy and precision were fluctuation between 0.88 to 0.89 and 0.85 to 0.88 respectively. My Stochastic Gradient Descent is the least consistent out of the datasets provided with it having a somewhat of a positive correlation in its accuracy and somewhat of a negative correlation in its precision, it also had the worst scores out of the three datasets provided with the lows of the accuracy and precision being 0.77 and 0.78 respectively and their highs being 0.82 and 0.85 respectively.

Comparing the two algorithms of Decision Tree and Stochastic Gradient Descent I can clearly see that my Decision Tree algorithm is both more accurate and more precise than my Stochastic Gradient Descent algorithm. However none of my algorithms including my ensemble ever went over 0.9 in either accuracy or precision and therefore they aren’t as great of results as I would’ve hoped for.

During my methods I encountered a problem with accuracy in my algorithms, this was because I had trained the algorithm wrong and this meant that my metrics were obscenely low for what I had predicted my metrics would turn out to look like. I fixed this by amending values for training which I had previously been training on the smaller values (b) which resulted in a very low quantity of training data.

**Conclusion**

To conclude I think that my Simple Stacking Classifier and the algorithms I used for my ensemble worked well, although my results weren’t as good as I had hoped they were with them all being less than 0.9 in both accuracy and precision. Therefore if I were to re-do my algorithms I would possibly change out some of my algorithms such as KNN and Stochastic Gradient Descent for others to check if they have a higher yield of accuracy and precision such as Naïve Bayes Guassian algorithm or Multi Layered Perceptron’s algorithm.

My precision rates could definitely be higher from using more algorithms however my algorithm is still relatively simple compared to some of the other corporate algorithms such as the Akismet algorithm which holds a precision rate of 0.999 or like the previously mentioned Mimecast which has a false positive rate of 0.0001 percent meaning that it has a precision of 0.999999. If I would want to reach a precision and accuracy scores similar to that of Akismet I would need to vastly improve upon my own algorithm by possibly using more algorithms as well as changing and using a different ensemble method to combine said algorithms.

If I were to have access to a larger amount of data of emails and spam emails then my precision and accuracy could also be higher as my algorithms would further be able to identify the differences between spam and not spam emails. As I only had access to just over 4000 entries it means that there was less of a chance for my algorithms to spot the difference between not spam and spam, even an increase of 1000 entries can greatly improve my metrics let alone having access to millions if not billions of entries which could possibly push my metrics to be over 0.95 instead of just past 0.9.

As mentioned above if I were to add more algorithms to my ensemble my metrics would improve, this is because the Simple Stacking Classification would have more scores to compare, bouncing the results of each algorithm of each other getting a greater sense of what makes a spam email a spam email. I can however only get so far by using someone else’s algorithm; I do think that if I were to develop my own algorithm I would be able to achieve a higher accuracy as well as a higher precision score. If I were also to add a neural network to my own algorithm which would mean that I could indefinitely develop itself allowing it to reach greater values of accuracy and more importantly of precision.

Therefore if I were to continue to improve and test my algorithm I would try and get my hands on a larger set of data to give it a larger chance to distinguish spam and not spam, I would also improve it by using more algorithms in my ensemble method. These two together would definitely greatly improve my metrics to beyond 0.95 and to get the best scores possible.

# Bibliography

*Cambridge Dictionary*. (n.d.). Retrieved from Cambridge Dictionary: https://dictionary.cambridge.org/dictionary/english/spam

*Lexico*. (n.d.). Retrieved from Lexico powerd by Oxford: https://www.lexico.com/

Maria Vergelis, T. S. (2019, November 26). *Spam and Phishing in q3 2019*. Retrieved from Kapersky: https://securelist.com/spam-report-q3-2019/95177/

Mimecast. (2003-2020). *Dunlop Case Study Mimecast*. Retrieved from Mimecast.com: https://www.mimecast.com/resources/case-studies/dunlop-case-study/

Mimecast. (2003-2020). *Mimecast Email Spam Blocker*. Retrieved from Mimecast.com: https://www.mimecast.com/content/email-spam-blocker/

*Spamlaws*. (2020). Retrieved from Spamlaws.com: https://www.spamlaws.com/spam-stats.html