MATH42815: Machine Learning Report

Anonymous Marking Code: Z0182576 Date: February 2023

1 Introduction

1.1 Motivation

Spam email is defined as a category of unwanted, potentially harmful email, which generally includes advertisements, money and cryptocurrency scams, sexually explicit content, and emails that are packaged with malware. Yet, in 2021, spam emails made up about 45% of total email traffic worldwide [Pet]. Naturally, there is a problem concerning how to effectively reduce the amount of spam email the user sees, to improve the user experience and to protect the user from potential scams. However, while most people will agree that emails in the aforementioned range of spam topics will be spam, the concept of whether an email is actually spam or not is dependent on the individual receiving it. For example, an email containing a 10% off coupon from a store an individual visited only once may be seen a nuisance to them and then marked as spam. But another individual may see the same email with the coupon differently, and instead want to keep the email. Clearly, a spam filter that works the same for the entire population would be too broad and imprecise. Instead, we will likely need to tailor spam filters to each person's preferences, so that emails that they would like to see are not hidden in the spam folder. That is, we would like to reduce the false positive rate for each person. Similarly, we would like less spam mail to appear in the user's inbox (a low false negative rate), but this is not nearly as detrimental as the false positive rate. This is because spam email in the inbox can easily be ignored, but desired or important email misplaced in the spam folder can easily be forgotten, as the spam box is not checked nearly as often. We aim to create a classification program that maintains a high accuracy rate while keeping the false positive rate as low as possible.

1.2 Data Exploration

This report aims to create a personalised spam filter using marked spam and non-spam email data provided by George Forman at [HRFS]. Certain properties of the emails that may indicate either a spam or non-spam email were created and the emails were scraped to obtain the data for each criterion. These criteria, along with the classification of 'spam' (as "1") or 'non-spam' (as "0"), form the variables of the dataset. In total, we have 58 variables in the dataset, 57 of which are predictors. 4601 emails are provided, 1813 of which are classed as spam (39.4%). We detail the type of variables and their properties in Table 1 below.

Variable name	Description of variable	Quantity	Data type
word_freq_WORD	Percentage of words that match the word WORD	48	Continuous
char_freq_CHAR	Percentage of characters that match the character CHAR	6	Continuous
capital_run_length_average	Average length of uninterrupted capital letter string	1	Continuous
capital_run_length_longest	Length of longest uninterrupted capital letter string	1	Continuous
capital_run_length_total	Total number of capital letters	1	Continuous
spam	Email classification: Spam (1) or Non-spam (0)	1	Nominal

Table 1: Types of variables, quantity of each type, and data type

The relevant words and characters are tabled below.

make	address	all	3d	our	over	remove	internet
order	mail	receive	will	people	report	addresses	free
business	email	you	credit	your	font	000	money
hp	hpl	george	650	lab	labs	telnet	857
data	415	85	technology	1999	parts	pm	direct
cs	meeting	original	project	re	edu	table	conference
;	(	[	!	$	#		

Table 2: Words and characters considered

The strings "george" and "650" are included in the emails classed as non-spam, since these are work and personal emails taken from the email donor George Forman at [HRFS] who works in area code '650' in the USA. If we wanted to make personalised emails for anyone, such data (name and location) would need to be taken into account before making a classification program, so that emails containing name and location are less likely to be flagged as spam. Looking at the other words, we may expect that spam emails will have a greater percentage of words that match "money" or "free", and a greater percentage of characters that match "!" or "$". We will be able to check the influence of certain words or characters on whether an email is classed as spam or not. This may be verified by looking at the summary statistics of models trained, such as a random forest model.

We begin by building a few models, and then analysing their accuracy, false positive rate, and any other notable statistics, before then suggesting the best model to use in a personalised spam filter.

2 Modelling and Analysis

2.1 Generalised Additive Model

The first model is known as a 'Generalised Additive Model'. It considers a combination of different models to predict the output. Since all the predictor variables are continuous, we consider splines of each of the 57 predictor variables, and then we train the model on the training data, which is an 80% split of all the emails. Then, we predict the class of the remaining 20% of the emails, which have had their class hidden. By looking at the number of times an email from one class (spam or non-spam) is predicted to be in another in a "confusion matrix", we will have a measure of the accuracy and the false positive rate for the algorithm. The confusion matrix resulting from training, and then testing the GAM is presented below.

		Actual	
		Spam	Non-Spam
Predicted	Spam	476	99
	Non-Spam	165	181

Table 3: GAM Confusion Matrix

In the table, we see that the GAM has an overall accuracy of

$$\frac{476+181}{476+181+99+165} = \frac{657}{921} \approx 71.3\%,$$

and a false positive rate of

$$\frac{99}{99+181} \approx 35.4\%.$$

This is quite undesirable and we may need a different type of model to deal with this problem. GAMs do not consider non-linear interactions between variables which may be important to our issue. For example, the chance of an email being spam may be much greater than a linear model may suggest if both the words "free" and "money" occur together with high frequency. In this case, we turn to two types of non-linear models: Classification Trees and Neural Networks.

2.2 Classification Trees

We start by considering a basic classification tree, which predicts the class of an email based on emails that are 'similar' to it. Again, an 80:20 training-testing split of the data is made, and then a tree model is trained using the training data. The tree is pruned to reduce any effects of overfitting, and the resulting tree is reproduced below:

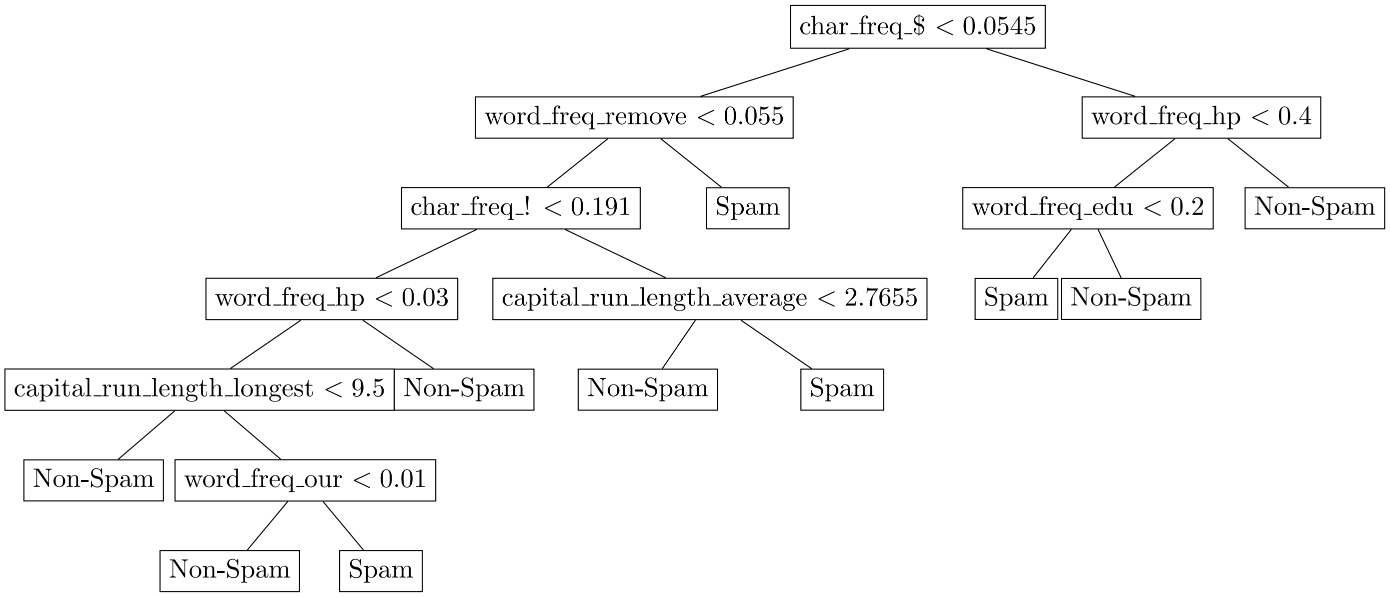

Figure 1: Classification tree to predict class of email

To read the tree, one evaluates the statement at the top of the tree, called the "root node". If the statement evaluates to true, proceed down the left branch, but if it evaluates to false, proceed down the right branch. After evaluating the necessary statements one arrives at a terminal node labelled "Spam" or "Not Spam", which serves as a prediction for the class of the email. The predicted classes of the test sample emails are compared to the actual classes and recorded in the confusion matrix below.

		Actual	
		Spam	Non-Spam
Predicted	Spam	540	52
	Non-Spam	45	284

Table 4: Classification Tree Confusion Matrix

We have that the overall accuracy of the tree is

$$\frac{540 + 284}{540 + 284 + 52 + 45} \approx 0.895 = 89.5\%,$$

and the false positive rate is

$$\frac{52}{52 + 284} \approx 0.155 = 15.5\%,$$

which by comparison to the accuracy and false positive rates achieved by the GAM (71.3% and 35.3% respectively), indicates that using classification trees as a spam filter will prove to be far more effective and useful. However, 15.3% of non-spam getting sent to a user's spam folder is again undesirable, and we wish to bring the false positive rate as close to 0 as possible. To do this, we aim to consider multiple classification trees and then 'average' the models, which should help reduce the variance of the model, and the effects of overfitting. This process is called "Bagging".

2.2.1 Bagging and Random Forests

We perform bagging choosing to average 500 classification trees. The class label of the email will be the label which the majority of trees predict. Since we are averaging multiple trees, we cannot represent the final model as a tree. We display the results in a confusion matrix below.

		Actual	
		Spam	Non-Spam
Predicted	Spam	571	27
	Non-Spam	14	309

Table 5: Bagging Confusion Matrix

The overall accuracy is

$$\frac{880}{921} \approx 95.5\%,$$

and the false positive rate is

$$\frac{9}{112} \approx 8\%.$$

This indicates that bagging results in a better model for spam filtering than a single classification tree! Moreover, the false positive rate is quite low, at only 8%. Still, we consider a variety of other methods to see if we can improve these statistics.

One such method is by using a random forest. In a random forest, whenever a split is made in a tree, only a subset of the predictors are considered for making the best split. This helps reduce the variance of the model, as using all predictors for all 500 trees in bagging leads to using correlated models, and so the resulting model would be more sensitive to deviations found in new data. The number of predictors to be considered in a split, m, is usually taken to be roughly $\sqrt{p}$ where p is the full number of predictors. Therefore, we start by considering $\sqrt{57} \approx 8$.

		Actual	
		Spam	Non-Spam
Predicted	Spam	573	29
	Non-Spam	12	307

Table 6: Random Forest Confusion Matrix, $m = 8$

It appears that $m = 8$ provides us with a higher false positive rate than we obtained from bagging. However, we can run the random forest procedure many times using different values of m in an acceptable range, and plot the results to see which value of m provides us with the lowest false positive rate. This is shown below.

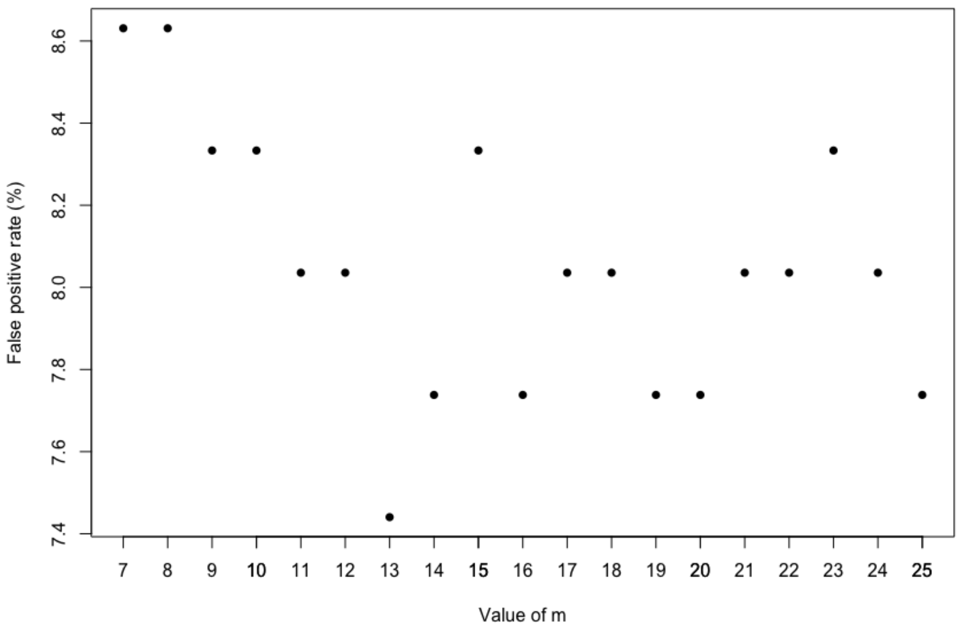

Figure 2: How the false positive rate varies with m.

The graph shows that the false positive rate is lowest when we choose $m = 13$. Below is the corresponding confusion matrix.

		Actual	
		Spam	Non-Spam
Predicted	Spam	570	25
	Non-Spam	15	311

Table 7: Random Forest Confusion Matrix, $m = 13$

Indeed, we achieve a false positive rate of $\approx 7.44\%$ and an overall accuracy rate of $\approx 95.7\%$, which are the best results so far. As previously mentioned, we can have a look into which variables in the model appear to be most influential in determining the classes. Below is a table of the five most and least influential variables for this random forest model. Each variable is matched with its "MeanDecreaseGini" score, the size of which represents how important the variable is (the greater the score, the more important the variable).

Variable name	Mean Decrease Gini	Variable name	Mean Decrease Gini
char_freq_$	231.6420250	word_freq_conference	1.3995758
char_freq_!	223.4203626	word_freq_cs	1.3086460
word_freq_remove	165.4655435	word_freq_415	1.1534387
capital_run_length_average	110.509762859758	word_freq_parts	0.5763720
word_freq_free	103.0633836	word_freq_table	0.2656188

Table 8: Most and least important predictors

From this table we see that the most influential variables correspond to the frequency of the dollar sign "$" and the exclamation mark "!", which is unsurprising considering of a lot of spam mail serve to scam people, often using a sense of urgency as a motivator. This is just as we predicted in our initial exploratory data analysis. The least influential variables correspond to the frequency of the words "parts" and "table", which again is not surprising as these words are not expected to be significantly more common in spam emails than in non-spam emails.

2.2.2 Boosting

We would like to test another method for training a model, known as boosting. Boosting is an ensemble method concerned with converting weaker models into more powerful ones by trying to minimise the error of previous models. By adjusting tuning parameters, we might be able to produce a model with a better false positive and accuracy rate than we could using random forests. We start by choosing the number of trees n to be 1000, and we obtain the results presented below.

		Actual	
		Spam	Non-Spam
Predicted	Spam	707	29
	Non-Spam	24	391

Table 9: Boosting Confusion Matrix, $n = 1000$

Interestingly, the false positive rate is $\approx 6.9\%$, lower than the estimate given by the chosen random forest model (with $m = 13$) as shown in Table 7. The accuracy rate is $\approx 95.4\%$, only slightly lower than the accuracy rate of the chosen random forest. We test values of $n = 1500, 2000, 2500, 3000$ to see if increasing n bestows a meaningful benefit in accuracy, and then we present the results in a table. Fortunately, we can choose n to be very large, since although overfitting may occur at large values, the growth of the effect of overfitting in boosting is minimal.

n	False Positive (%)	Accuracy Rate (%)
1000	6.90	95.4
1500	7.14	95.3
2000	7.14	95.4
2500	7.62	95.0
3000	7.38	95.0

Table 10: Boosting Confusion Matrix, $n = 1000$

We see that there appears to be no benefit to neither the false positive rate or overall accuracy when increasing the value of n past 1000. While we conclude that boosting cannot provide more predictive power than using a random forest in this problem, it is only slightly weaker. This drop in accuracy is offset by the fact that the boosted model does offer the lowest false positive rate of all the methods, which we are placing greater importance on compared to the accuracy, so we will favour the boosted model over the random forest.

2.3 Deep Learning Neural Networks

In efforts to see if we can improve upon the boosted model, we train and test another non-linear classification model. This model uses neural networks to make a function approximation for the model by attempting to minimise the loss function, which for a binary classification problem, is the "binary cross-entropy". The model will have 57 input neurons corresponding to each of the predictor variables, two hidden layers with 16 neurons in the first and second layers each (following advice from Heaton at [Hea]), all using the 'ReLU' activation function, and a single output layer neuron which uses the sigmoid activation function. The output of the neural network (which corresponds with the range of the sigmoid function $[0, 1]$) is then compared with 0.5: if the output is greater than 0.5 then the email is classed as spam, and non-spam otherwise.

Using the Adam optimiser for advanced gradient descent and a learning rate of 0.01, we train the model considering the binary cross entropy loss function over 100 epochs. How the accuracy of the model changes over epochs is illustrated below.

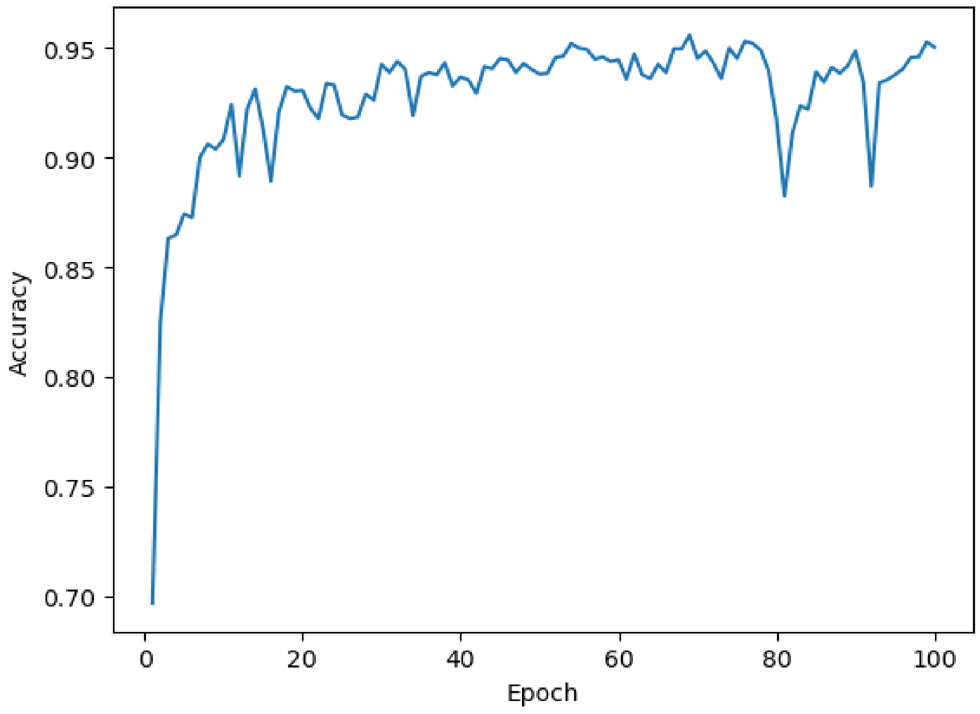

Figure 3: How the accuracy varies with successive epochs.

It appears that training the model for longer will not yield any significant further gain in accuracy. Thus, we elect to train the model for 100 epochs. The resulting classification matrix is presented below.

		Actual	
		Spam	Non-Spam
Predicted	Spam	543	32
	Non-Spam	23	323

Table 11: Neural network classification matrix

The false positive rate is $\approx 9.01\%$, and the overall accuracy is $\approx 94.0\%$. As a result, we still have that the accuracy and false positive rate of our boosted model remains superior. However, we may decide to sacrifice overall accuracy of our neural network model to bring the false positive rate down. We can tune a variety of parameters to achieve this: the class weights in the loss equation, the learning rate of the optimiser, the number of hidden neurons and epochs used in training, or the threshold probability that determines whether we class an email as spam or not. We will adjust the number of hidden neurons and the weight of the 'non-spam' class so that misclassification of non-spam emails as spam emails results in a bigger penalty (the loss function gets bigger). We also alter the threshold probability so that we require the model to be 'more certain' about an email being spam before we classify it as such.

Our model's first hidden layer has 36 neurons and second hidden layer 2 has 32 neurons. We adjust the class weights so that the weight of the non-spam class is ten times that of the spam class, and change the threshold probability from 0.5 to 0.9993 to obtain a model yielding the following results.

		Actual	
		Spam	Non-Spam
Predicted	Spam	575	0
	Non-Spam	234	112

Table 12: Neural network classification matrix to minimise false positive rate

In this case, we achieve the desired 0% false positive rate! While promising, it is only valid for a particular split of the test data, and we cannot guarantee that absolutely no emails will be incorrectly marked as spam (unless we force that every email must be marked as non-spam), since any new samples may be significantly different from the observations in the training set. Furthermore, the false negative rate has increased notably and the accuracy has decreased to $\approx 74.6\%$. Therefore, we expect that about 25.4% of spam messages bypass the email filter. However, this is still more desirable than a model with a high false positive rate, making this neural network model the preferred model, even compared to our boosted tree model.

3 Discussion and Further Work

The best choice of model is down to individual tastes. For the purpose of this report, we have selected the deep neural network model as the best since it can classify emails with the lowest false positive rate. However, one who places a greater emphasis on accuracy might settle for the random forest or boosted models, which are probably more suitable for email addresses not created for work purposes. Ideally, this would be down to the user of the email provider to decide in a settings panel. We also saw that the generalised additive model did not outperform any of the tree-based models or the neural network model in either false positive rate or accuracy, most likely owing to the fact that it does not account for non-linear interactions between the predictor variables.

Furthermore, in conducting this analysis we have only used the personal emails provided by the donor, which have been marked as spam or non-spam according to their preferences. As stated in section 1.1, whether an email will be classed as spam or not is down to the individual. Therefore, instead of looking at the classification of emails as a binary classification problem, it may be useful to consider instead three different categories: "non-spam", "spam", and "probably spam". An example as to how this can be implemented in the neural network case is by creating three sections of the range $[0, 1]$ that determine which one of the three categories the email fits into. Then, "probably spam" emails can appear in the user's inbox as usual but with a warning attached, that lets the user know to ignore the message or proceed with caution. Whether the user marks that email as spam can be used as data to update the model or add the sender to a blacklist. Since what may be regarded as spam mail can be subjective, every email account should note what the user considers to be spam, and use that to calibrate the spam filter to their preferences, so that the accuracy can improve, as the model becomes more 'confident' about what the user considers to be spam.

We might think about how we can create a generalised spam filter instead of a personalised one. If we had access to a much larger collection of labelled emails, from a variety of different people, we would be able to distinguish between spam and non-spam emails by looking at the overlap in content between spam emails sent to many different people. For example, the model built using the data from [HRFS] considered the string representing the area code '650', which were signs of non-spam emails, since they were relevant to the receiver. However, other emails using the string '650' may not be relevant to others that live in other areas, and accordingly marked as spam, so its 'non-spam' affiliation may be drowned out, as we would prefer in a generalised spam filter.

4 Executive Summary

This report has tackled the problem of trying to create a personalised spam filter based on the classifications of emails as spam or non-spam by an email donor. We achieved this by writing a program that 'learns' what a typical spam email looks like using the examples provided by the donor. The program learns what a spam email looks like by scanning the email for features that scam mail might have, such as an uninterrupted sequence of capital letters, or a high prevalence of the words "money" or "free", and then classifies a new email as either spam or non-spam depending on whether they possess these features to a significant extent. This is referred to as "training" the model. The model is then tested on emails that we know the classification of, and its accuracy is measured. It is important to ensure that our model is accurate, because we would like to prevent users becoming exposed to unwanted material or victims of scams, which is especially likely for users that may not be comfortable with technology.

This leads into the second problem: how far we can push the accuracy? Ideally, we would have a perfectly accurate program that always classed non-spam mail as non-spam and spam mail as spam. Unfortunately, this is not the case and all of the models that we experimented with did not have a perfect 100% accuracy. One of our models, called the "Random Forest" model, managed to predict the correct class label for the emails about 95.7% of the time, and remains the best model accuracy wise. However, accuracy is not the only metric we used to judge the effectiveness of the classification programs. We also used the 'false positive rate', which refers to how often non-spam emails are classified as spam. This is undesirable as non-spam emails that the user may be expecting or find useful can be hidden in the spam folder, which is inconvenient for the user. In this report, we placed a greater emphasis on reducing the false positive rate rather than increasing the accuracy, as spam emails appearing in one's inbox is often seen as a mild annoyance compared to non-spam emails being hidden in the spam box which can be a hindrance to productivity. A higher false positive rate may drive potential users away if they become frustrated with having to find or preemptively check for missing important emails in their spam folder. Another model we looked at, that we call a "boosted model", offered a slightly lower accuracy than the random forest model, at 95.4%, but in return it lowered the false positive rate from 7.44% to 6.9%. This offers some sort of compromise between the two metrics. Since we were intent on reducing the false positive rate down to zero, we have put forward another model which operates using a "neural network". It offers a roughly 0% false positive rate, but results in a decrease in overall accuracy to 74.6% - we end up having some spam mail bypassing the filter and getting into the inbox, but as previously mentioned, this is preferable to the alternative. Therefore, the neural network model is recommended.

5 Bibliography

References

[Hea] Jeff Heaton. The number of hidden layers. Accessible at: https://web.archive.org/web/20140721050413/http://www.heatonresearch.com/node/707.

[HRFS] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spambase data set. Accessible at: https://archive.ics.uci.edu/ml/datasets/Spambase
Email Donor: George Forman.

[Pet] Ani Petrosyan. Global spam traffic. Accessible at: https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/.