Warren Brown (student)

Spam Classification report

Contents

[Introduction 2](#_Toc4674096)

[Development of the classification neural network 3](#_Toc4674097)

[Reading in the dataset 3](#_Toc4674098)

[Pre-processing the dataset 3](#_Toc4674099)

[Training 3](#_Toc4674100)

[Evaluation of the algorithm 5](#_Toc4674101)

[Cross-fold validation 5](#_Toc4674102)

[Advance techniques 6](#_Toc4674103)

[Hyperparameter optimisation 6](#_Toc4674104)

[Conclusion 7](#_Toc4674105)

[Reflection 7](#_Toc4674106)

# Introduction

The goal of this report is to explain the method I used to create a machine learning classification algorithm to decide if an email is spam or not. Within this report I am going to explain what I have done to make the algorithm more efficient and accurate as well as any data manipulation that I have used to produce more accurate results.

# Development of the classification neural network

## Reading in the dataset

To read in the dataset file I made sure that the python file and the dataset were in the same folder to simplify the process as much as possible. I then imported and used the ‘pandas’ library to open the dataset.csv file, as seen in Figure 1 it required very little code.



[Figure 1: using pandas to read in csv dataset]

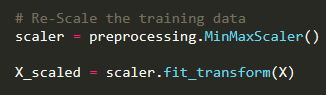
## Pre-processing the dataset

The data now needs to be pre-processed before I can apply any machine learning algorithm onto it. The first thing to do is to sperate the training data from the classification label, this is done as shown in Figure 2 where the data is separated into two different variables.



[Figure 2: Separating the data]

The next thing that I wanted to do as to normalize the values within the data between 0 and 1. This is because I have chosen to use a neural network to classify the emails into spam or not, and it works better when the input numbers are between 0 and 1. Also, a good general rule of thumb is to scale or normalize the input values because there is sometimes drastic differences between some input values. This was incredibly simple using scikit-learn (sklearn for short), as seen in Figure 3 it only required 2 lines of code.



[Figure 3: Scaling the data]

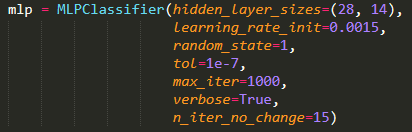
Figure 4 shows how I split the dataset into different lists, X\_train and Y\_train to hold training data, and then X\_test and Y\_test as their respective expected results/classification. As



[Figure 4: Splitting the data]

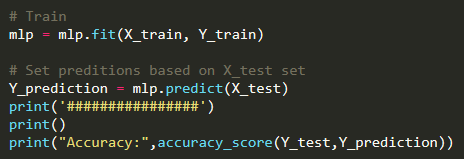
## Training

To train the MLP (Multi-Layer Perceptron) classifier I had to create a ‘MLPClassifier’ object and pass it in params. From the sklearn documentation I was able to research the different param options that can be passed into the ‘MLPClassifier) constructer as shown in Figure 5.d



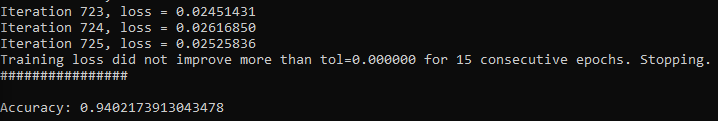
[Figure 5: MLP Classifier parameters]

The next step is to train the ‘mlp’ using the params that I passed in. this is done as shown in Figure 6, by using ‘mlp.fit()’ and passing in the two training data lists. It uses the past in params to contanlty adjust the weights of the neural network by using back propagation. How this is done is on the training set of data the ‘mlp’ knows what the expected results should be, the ‘mlp’ runs through once and then adjusts the weights depending on the accuracy of the answer that was output by the hidden layers. This continuously happens until it reaches the constraints that I set during the creation of the ‘mlp’.



[Figure 6: Training and testing code]

Below in Figure 7 is the output of the ‘mlp’. As you can see it ran through 725 iterations before it satisfied all the conditions. The produced accuracy was 94% with the params that I originally used. Although this is not completely accurate as the ‘mlp’ when ran again will produce a different accuracy result as I will explain in the next section.



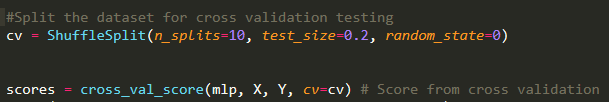
[Figure 7: Training output with accuracy]

# Evaluation of the algorithm

As previously mentioned, testing the algorithm once does not prove that the accuracy is 94% as shown in Figure 7. This is because the weights for the neural network are randomly generated when it starts, and therefore, sometimes better weights are generated or in some case worse weights are generated. This means that sometimes the neural network may randomly be very good and reach 96%, whereas other times it may only achieve 80%.

## Cross-fold validation

Cross-fold validation is a way to get a precise idea of the true accuracy of a neural network. It is very simple, depending on how many slits are passed into the ‘n\_split’ param as seen in Figure 8, the dataset will be split into say 10 random parts. The neural network will then be ran on each of the 10 parts and from this 10 accuracy scores will be generated.



[Figure 8: Cross Validation code]

With the generated accuracy scores for each split of the dataset we can then work out a mean average of the neural network’s accuracy. This gives a truer accuracy as the network has been ran multiple times therefore, removing anomalies where the accuracy is a lot higher or lower than expected. Figure 9 shows the output of the ‘ten-fold- validation that I have used on my neural network. Notice how the mean accuracy is lower than the accuracy I received in Figure 7, even though the exact same neural network was used.



[Figure 9: Cross validation output]

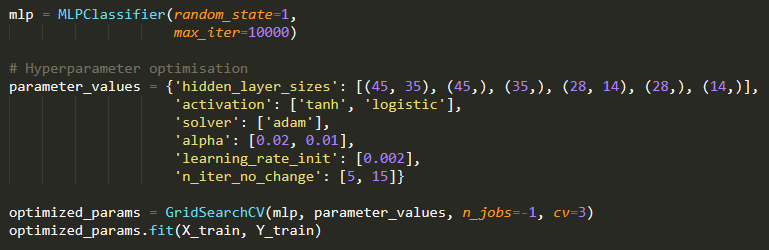
# Advance techniques

A major design issue for neural networks is how many hidden layers the modal should use along with how many neurons should be in each of the layers. A solution to this problem is ‘Hyperparameter Optimisation’.

## Hyperparameter optimisation

Hyperparameter optimisation is essential a method of trying to find the optimal parameters to use for a neural network. Params such as the hidden layer amount and length as well as learning rate, the activation function to use and the ‘alpha’ or penalty given to neural networks with a low fitness/accuracy score.

Using sklearn I was able to implement a ‘GridSearchCV’ that took in a dictionary of parameter values that I declared in the variable, ‘parameter\_values’ in Figure 10. This dictionary is then passed to a GridSearchCV object along with the MLP, amount of jobs to run simultaneously (n\_jobs) and lastly in my example the cv or cross validation amount.



[Figure 10: Hyperparameter params]

After the GridSearchCV object is created it is ran. It then checks all the combinations of the parameter dictionary that was passed into it and produced an output of the best params to use as shown in Figure 11.



[Figure 11: Best params]

Using the values that were returned from the GridSearchCV and passing them into a MLPClassifier and applying a 10-fold cross validation, I received the output shown in Figure 12. As you can see this is similar to what I received from the original ‘mlp’ in Figure 9. Therefore, it is possible that the ‘mlp’ cannot be optimised past what I have already achieved and in order to gain I higher accuracy than 93-95% I must do some more data processing.



[Figure 12: Cross-fold validation output]

# Conclusion

I have achieved everything that I have set out to do from the introduction. I have explained what algorithm I have used as well as the steps I have taken to try and get the most accurate results. I think that a 10-fold cross validation mean accuracy of 93% is good for this dataset, with a higher bound of 95% accuracy. This demonstrates that the neural network is a practical and effective way of detecting spam emails with the provided data set.

# Reflection

In reflection I would have liked to show the differences between the accuracy of the neural network machine learning approach and a KNN or random forest for example, to highlight to increase in accuracy when a neural network is used.

Another thing that I would have liked to do is spend a lot more time on dataset pre-processing. There are a lot a rows and columns in the dataset that are 0 values and I think that if I had combined these columns and mean averaged the values of these columns I may have been able to increase the accuracy of the neural network more.

Overall, I really enjoyed this assignment my knowledge of neural networks as well as other forms of machine learning has greatly improved. I look forward to being able to apply this knowledge in the future.