Student number: 18821327

University of Brighton, School of Computing, Engineering and Mathematics

Intelligent Systems 1

Reflective report

Table of Contents

2. The report………………………………………………………………………….

3. Graphs……………………………………………………………………………….

5. References…………………………………………………………………………

What is the problem?

The prediction task is trying to automatically sort an email into spam or not spam based on the information of the email. Without the implication of a spam filter on emails, user’s inboxes would be filled with unimportant/scam emails causing the user to miss important emails. Problems to the training of this algorithm will occur when a user marks a non-spam email as spam, this will impede on the learning process of the algorithm.

The dataset

The spam dataset contains 4601 instances each with 57 attributes, the first 56 are the features which are 48 continuous real [0,100] of type word\_freq\_WORD, 6 continuous real [0, 100] of type char\_freq\_CHAR, 1 continuous real [1,…] of type capital\_run\_length\_average, 1 continuous integer [1,…] of type capital\_run\_length\_longest and 1 continuous integer [1,…] of type capital\_run\_length\_total. Leaving the final attribute to be the target attribute which is 1 nominal {0,1} of type spam. The collection of spam e-mails is from the creator of the dataset’s postmaster (administrator of a mail server) and individuals who had filed spam. The collection of non-spam e-mails came from personal and filed work e-mails.

The algorithm

The algorithm selected is a hard voting classifier, combing a Neural network and Random forest classifier. When deciding on which algorithm to select for the task I ran all of the following algorithms in the using Grid Search cv with 5 folds\*\*, K-Nearest Neighbour; Neural network, Random forest, SVC, Decision tree, Naive bayes. The purpose of this was to find the best hyper parameters for my classifiers whilst also finding the best algorithms to include in the voting classifier. In the end the two highest scoring by were Neural network and Random forest, so I decided to combine them to get a higher accuracy and precision score. The lowest scoring was K-Nearest Neighbour, this is why I have selected it as the benchmark algorithm.

Evaluation

The metrics used to evaluate the classifiers are accuracy and precision. When looking for a benchmark algorithm to compare to my algorithm I opted for K-Nearest Neighbours. This proved useful in evaluating the success of my algorithm. After comparing to a simpler algorithm, I was able to see how much better the voting classifier is. Looking at the graphs below you will be able to see the voting classifier is a much higher performing algorithm than the benchmark algorithm K-Nearest Neighbours.

‘evc’ = Voting classifier, ‘rf’ = Random forest, ‘mlp’ = Neural network, ‘knn’ = K-Nearest Neighbours

Results

Accuracy over 5 tests

Precision over 5 tests

Average accuracy

Average precision

\*\*The grid search cv file is included in the zip, however it does take a long time to complete the process due to the random forest classifier

Reference list

Hopkins, M., Reeber, E., Forman, G. and Supermondt, J. (1999). *UCI Machine Learning Repository: Spambase Data Set*. [online] archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/spambase [Accessed 1 Dec. 2020].

scikit-learn (2010). *sklearn.neural\_network.MLPClassifier — scikit-learn 0.20.3 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html [Accessed 12 Dec. 2020].

scikit-learn (2018). *3.2.4.3.1. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed 15 Dec. 2020].

scikit-learn (2019a). *sklearn.model\_selection.GridSearchCV — scikit-learn 0.22 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html [Accessed 5 Dec. 2020].

scikit-learn (2019b). *sklearn.svm.SVC — scikit-learn 0.22 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html [Accessed 7 Dec. 2020].

scikit-learn (2019c). *sklearn.tree.DecisionTreeClassifier — scikit-learn 0.22.1 documentation*. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html [Accessed 12 Dec. 2020].

scikit-learn (2020a). *sklearn.metrics.accuracy\_score — scikit-learn 0.24.1 documentation*. [online] scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy\_score.html#sklearn.metrics.accuracy\_score [Accessed 5 Dec. 2020].

scikit-learn (2020b). *sklearn.metrics.precision\_score — scikit-learn 0.24.1 documentation*. [online] scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision\_score.html#sklearn.metrics.precision\_score [Accessed 5 Dec. 2020].

scikit-learn (n.d.). *sklearn.naive\_bayes.GaussianNB — scikit-learn 0.22.1 documentation*. [online] scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.naive\_bayes.GaussianNB.html [Accessed 17 Dec. 2020].