CS7641 - Supervised Learning

Student username: plivesey3 GTID: 90336997

Introduction

This assignment aims to examine and compare 5 different machine learning algorithms (including Decision Trees with Pruning, Neural Networks, Boosting, Support Vector Machines and k-nearest neighbours) using a variety of different metrics including learning and validation curves.

Two interesting data-sets have been chosen for the comparisons described, including:

Dataset 1: The *UCI Phishing Websites Data Set* which predicts whether a website is one that will attempt to steal a user data by comparing various attributes such as the URL and whether a URL shortening service is employed. The results are binary classification; It is either a Phishing website or it is not.

Dataset 2: For the *UCI Red Wine Dataset* various attributes of a bottle of wine are used to attempt to predict a quality grading for the wine. The results are a multi-class classification, which is the numeric grading of the wine with a higher value representing higher quality.

The project will make use of the Scikit-Learn data analysis and mining library for Python for its machine learning algorithms and metrics amongst other useful functions.

Datasets

Phishing is a constant battle between the groups trying to send the attacks and the people trying to block them. Being able to detect and stop website phishing attacks is very important for businesses as well as individuals. Using machine learning techniques it is possible to inform users when they are about to visit a site that shows a potential to be a phishing site. This is what we will attempt to replicate here using the UCI Phishing Websites Data Set.

Another useful way in which data can be used is to give an item a possible quality grading. In this case we will attempt to give red wine a numeric grading from attributes such as amount of alcohol, chlorides or citric acid.

The Phishing dataset is a binary classification problem whereas the Red Wine dataset is a multi-class classification problem. . There are two main reasons for comparing these to different datasets. The first is the size. The Red Wine dataset consists of only 1599 samples. It was suspected that many of the algorithms would have problems producing accurate descriptions compared with the 11055 samples in the Phishing Dataset. The second reason these datasets were selected is to compare the different ways a multi-class classification problem compares with a binary classification problem.

When examining data sets, precision is the number of correctly classified positive classes over the predicted positive classes. For over Phishing data set, we do not want data that is false positive to appear (i.e. it is a most important that we don't get much spam), so we want the precision to be high.

Recall is the number of correctly identified positive classes over the total number of positive classes. For the Red Wine dataset, it isn't too important if we give a false alarm. It is more important that we find the true positives correctly. For this reason we should be looking for high recall for the red wine dataset. Unfortunately the recall could not be used with GridSearchCV for Red Wine as it does not work with this metric for multi-class data, so the f1 score was used instead.

Methods

The datasets have been split into 70/30 proportions, with 70% being used for the training set and 30% for the test set. 5 different supervised learning algorithms have been applied to the datasets, including: Decision Tree, Multi-Layer Perceptron , K-Nearest Neighbour, AdaBoost and Support Vector Machine algorithms. The Scikit-Learn library was used to run the

algorithms, as well as for analysis of the results. The hyper-parameters for each of the algorithms were tuned via cross-validation. A learning curve and validation curve is given for each algorithm as well as a detailed analysis of the results and the algorithms performance. Other charts have been included where necessary.

The Phishing data set uses binary classification to specify whether the result is whether a phishing website (Result = 1) or not (Result = -1). A Phishing website is such when it attempts to steal your password or other confidential information. This is generally achieved by replicating legitimate websites.

It can be seen from the count-plot in figure 1 that most of the samples are found to have a grading of 5 or 6, which will have important consequences for the performance of the different algorithms.

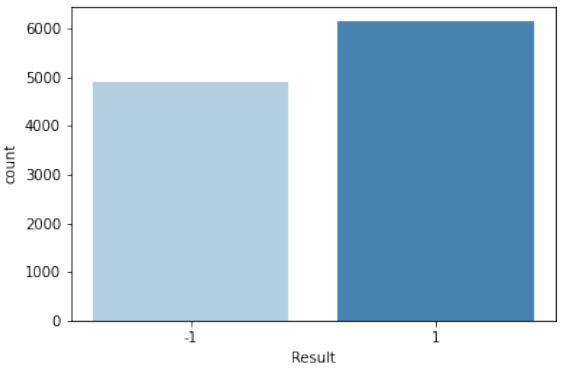

Figure 1: Phishing Dataset Count Plot

The three highest correlating attributes (Persons r measure) include *SSLfinal_State at 0.72%*, *URL_of_Anchor at 0.69%* which tests whether an anchor contains the same domain name as the site as well as whether there are anchors that link to no site at all, and *Prefix_Suffix at 0.35%* which includes websites the use prefix and suffixes to domain names (e.g. 'hotrockpaypal.com'). The attribute with the lowest correlation is popUpWidnow (sic) at 0.000086 (see figure 2) which counts websites that use pop-up windows.

Examining the Red Wine data set, it can be

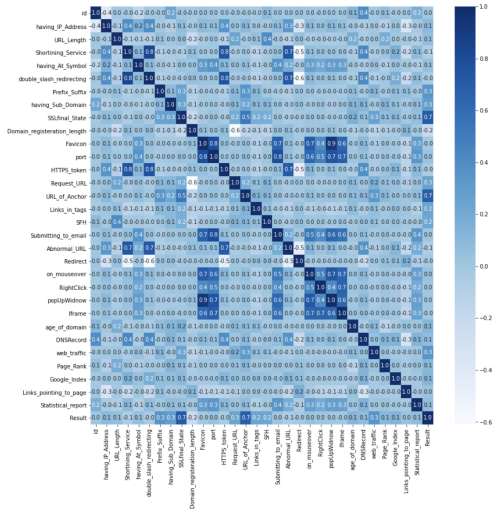

Figure 2: Phishing Attribute Heat Map

seen that most of the data gathers around the quality values of 5 and 6 (See figure 3).

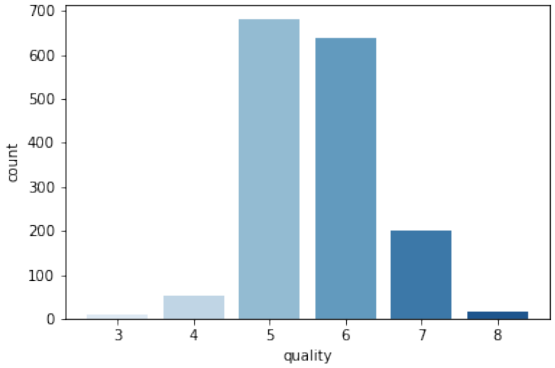

Figure 3: Red Wine Data Set Count Plot

The red wine dataset is very limited in the attribute that correlate (see diagram). The three highest correlating attributes (Persons r measure) include alcohol at 0.48%, volatile acidity at 0.39% and citric acid at 0.23%. The attribute with the lowest correlation is residual sugar at 0.01%. The red wine data set is very limited in the attribute that correlate (see figure 4).

The Scikit-Learn Gridsearch algorithm will be used throughout the experiments to help determine the best hyper-parameters. These will be altered when they are found to be less than optimal.

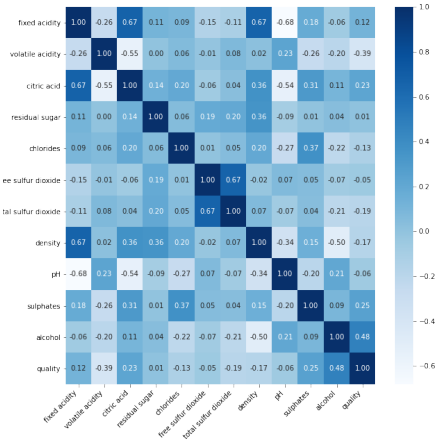

Figure 4: Red Wine Attribute Heat Map

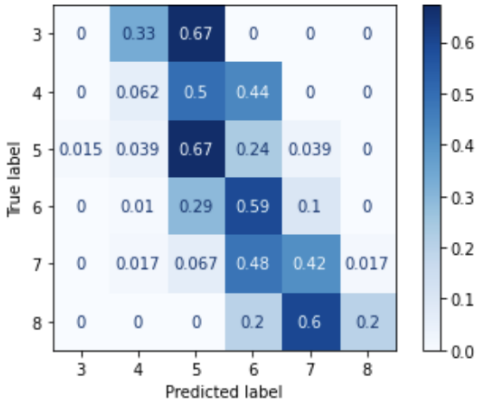

Figure 5: Phishing Data Set Confusion Matrix

Decision Trees - Phishing Dataset

The Decision Tree algorithm builds up tree of questions about a samples attributes that allow the sample to be classified. Running the Phishing data through the Scikit Learn Decision Tree Gridsearch with the Criterion being either Gini or Entropy and the maximum tree depth ranging from 1 to 59, the attributes for the latest run came out as Criterion: **Entropy**, Maximum Depth: **8** and Class Weight: : **balanced**. Criterion being Entropy was expected as this generally gives higher results.

The cross-validation used is 'Kstratified' which works better with classification models compared with 'KFold', which is more useful with regression models [1]. This is repeated for each of the following algorithms.

	precision	recall	f1-score
-1	0.94	0.96	0.88
1	0.96	0.95	0.91
weighted average	0.95	0.95	0.95
accuracy			0.96

For the Phishing dataset, probably the most important numbers are: **False Positives** - These show the number of websites that get through the filter, with a relatively low probability of 0.049. **False Negatives** - These are the wanted websites that are caught in the filter, with a commendable probability of 0.95. **Accuracy** - This shows the ratio of the correct predictions against the total number of predictions, which is the number of websites we predicted to be true correctly, here giving a probability of 0.96.

Precision (weighted average - We want to reduce False positives (when real websites are labeled as phishing websites), so this is important. Here we have a decent probability of 0.95.

The validation curve for the Phishing dataset compares accuracy against the maximum depth shows results as would be expected. The training score achieves a near perfect score upon reaching a depth of 2, although obviously massively over-fitted at that point. The maximum depth for the cross-validation score tops out around a depth of 15, which shows that this would be a reasonable balance between bias and variance, although slightly higher than that given by GridSearch. When a depth of 15 was tested individually, the precision, recall and accuracy were slightly lower, although the f1 scores where 7% and 5% highest respectively.

Figure 8 shows the learning curves of the Phishing data set for contrasting maximum depths, which generally confirm the findings from the validation curve. Here we can see the training set accuracies increasing as the maximum depth goes up. Initially, there is quite a difference in cross-validation accuracy when the maximum depth is very low, but for any training set sizes above 200 we get a slightly better accuracy for depths of close to 10. After 10, the

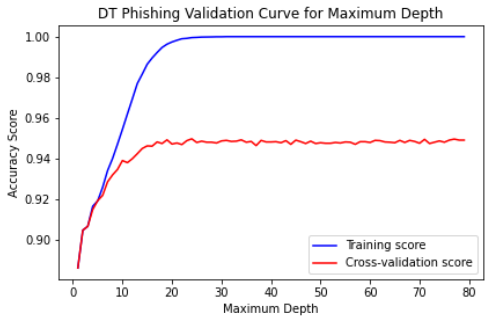

Figure 6: Decision Tree Phishing Validation Curve

testing accuracy can be seen to be slightly lower. So, low depths are generally more accurate for this data set when the sample size is small, but after 200 it is more accurate when the optimal depth of 8 is used.

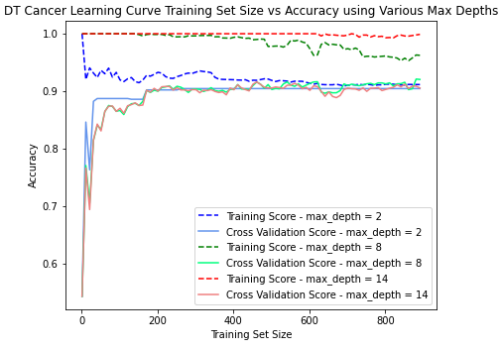

Figure 7: Phishing Learning Curves

Decision Trees - Red Wine Dataset

Running the Red Wine data through the Scikit Learn Decision Tree Gridsearch with the same parameters as above gave the highest f1 score with the following values: Class Weight - **balanced**, the same as the Phishing dataset, A Criterion of **Entropy**, again the same as Phishing and is unsurprising as it tends to give a slightly higher average compared with Gini, and a maximum depth of 17. The learning curves later on show that this is very overfit and a maximum depth of 9 seems to strike the right balance and give the highest results

	precision	recall	f1-score
3	0.00	0.00	0.00
4	0.17	0.06	0.09
5	0.62	0.73	0.67
6	0.58	0.52	0.55
7	0.52	0.55	0.54
8	0.00	0.00	0.00
accuracy			0.59
weighted average	0.57	0.59	0.58

For the Red Wine dataset, probably the most important numbers are: **Accuracy** - This shows the ratio of the correct predictions against the total number of predictions, which is the number of wines we predicted to be true correctly, here giving a very probability of 0.59. This can be blamed on the relatively small dataset. Only grade of 5 and 6 have a large number of samples, whereas the others are very low.

Recall (weighted average) - We want the False Positives (the correct predictions) to be as high as possible. This is again low with an value of 0.59

The validation curve for the Red Wine data sets accuracy against the maximum depth shows results as would be expected. The training score achieves a near perfect score upon reaching a depth of 17, but the training line rises rapidly to overfitting. A balance of between 8 and 12 was experimented with, with a final result of 9 achieving the best results overall.

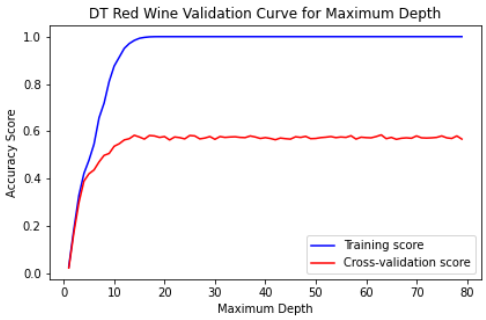

Figure 8: Decision Tree Red Wine Validation Curve

Figure 11 shows the learning curves of the Red Wine data set for contrasting maximum depths and it is a little different to the Phishing

chart in that the cross-validation is not stabilizing, but slowly increasing, which gives us hope that more data will make the tree more accurate. It is also lower, which confirms everything we have seen above. The different depths produce similar results above 10, but once again, anything over 10 is massively overfitting the data.

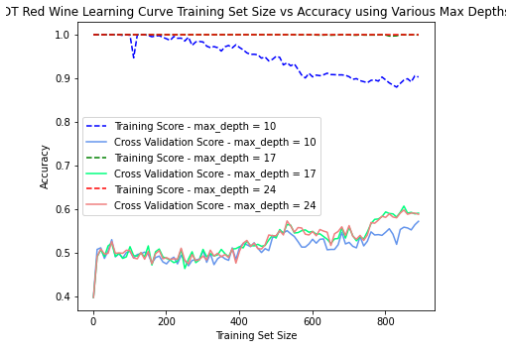

Figure 9: Decision Tree Red Wine Learning Curves

Bias and Variance - Decision Trees

It is easy to overfit a Decision Tree as the depth of the tree can allow for the data to be fit too closely to data. With the Scikit-Learn library, one way to overcome this is to try to find a balance between the maximum depth and the metrics we are most concerned about. It can be seen from the learning curves for both dataset, maximum depths over 10 give training accuracies that are almost perfect and therefore overfit, so a value lower than this that achieves a descent accuracy should be chosen.

K-Nearest Neighbors - Phishing Dataset

The K-Nearest Neighbours algorithm attempts to classify by matching a sample with other data that is relatively close in value distance for each of its different attributes. This is a relatively simple algorithm. In some ways this is because a lot of the design decisions are up to the coder. The algorithm choices made can have a significant difference to the results. Here three of the more important will be examined, including the distance metric, the number of nearest neighbours, the distance weighting and the algorithm itself.

Running the Red Wine data through the Scikit Learn K-Nearest Neighbours (KNN) Gridsearch with the same parameters as above gave the highest f1-score with the following values: the number of neighbours came out at **2**, the algorithm was **KDTree**, the metric was **Manhattan** and weights was **Uniform**.

	precision	recall	f1-score
-1	0.93	0.97	0.95
1	0.98	0.94	0.96
weighted average	0.95	0.95	0.95
accuracy			0.96

False Positives - The number of websites that would get through the filter are very low relatively at 0.029 **False Negatives** - These are the wanted websites that are caught in the filter, with an excellent probability of 0.97. **Accuracy** - Again, a very good result at 0.96

Precision (weighted average - An excellent probability of 0.95.

The validation curve for the Phishing data sets accuracy against the number of nearest neighbours drops consistently for both cross-validation and testing. This explains why the Gridsearch gave the lowest value of 2 for the number of nearest neighbours.

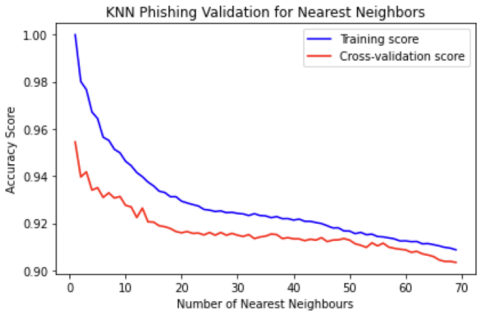

Figure 10: KNN Phishing Validation Curve

Could not get the learning curve working on time.

KNN - Red Wine Dataset

Running the Red Wine data through the Scikit Learn K-Nearest Neighbours (KNN) Gridsearch

with the same parameters as above gave the highest f1-score with the following values: the number of neighbours came out at **13**, the algorithm was **KDTree**, the metric was **Manhattan** and weights was **Distance**. The algorithm and metric were the same choices as with the Phishing dataset Gridsearch.

	precision	recall	f1-score
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.72	0.76	0.74
6	0.64	0.69	0.67
7	0.74	0.65	0.69
8	1.00	0.20	0.33
accuracy			0.69
weighted average	0.66	0.69	0.67

Accuracy - This is the highest of all the algorithms at 0.69.

Recall (weighted average) - Again this is an excellent result (for this data) with 0.69

The validation curve for the Red Wine data sets accuracy against the number of nearest neighbours and the accuracy increases until 12 and then seems to stabilize. The testing stays constantly at 1.0.

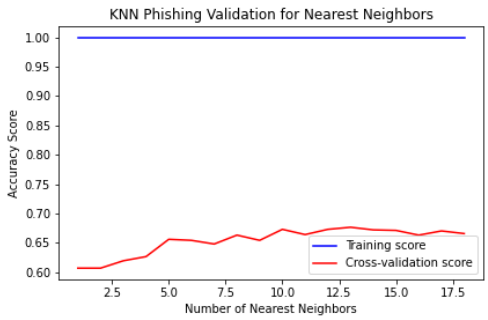

Figure 11: KNN Red Wine Validation Curve

Could not get the learning curve working on time.

Bias and Variance - K-Nearest Neighbours

For the Phishing dataset there never seems to be a problem with overfitting as the training value drops as the nearest neighbours increases and so the smallest number of neighbours at 2 is best, although this does not allow for much balancing between bias and variance. For the Red Wine dataset, the training is constantly at 1.0 which seems to be overfit for every value. More information is needed (the missing learning curve may have helped here) to determine was is happening here.

Multi-Layer Perceptron - Phishing Dataset

The MLP uses connected nodes named artificial neurons which are loosely based on the neurons in a biological brain.

Running the Phishing data through the Scikit Learn Multi-Layered Perceptron (MLP) Gridsearch with the same parameters as above gave the highest precision with the following values: an activation function of **Tanh**, the number of hidden layers came out as **19**, the alpha was **1e-06**, the maximum iterations was **1000**, and a **Constant** learning rate was deemed best.

Running the Phishing Data through the Scikit Learn MLP Tree Gridsearch with the achieves the following results, which have the highest precision and recall of all of the algorithms:

	precision	recall	f1-score
-1	0.97	0.95	0.95
1	0.96	0.98	0.96
weighted average	0.96	0.96	0.96
accuracy			0.96

False Positives - These show the number of websites that get through the filter, with a quite a low (although not the lowest) of 0.061. **False Negatives** - These are the wanted websites that are caught in the filter, with an excellent probability of 0.97. **Accuracy** - This shows the ratio of the correct predictions against the total number of predictions, which is the number of websites predicted to be true correctly, here giving a probability of 0.96. **Precision (weighted average** - A reasonable probability of 0.96.

The validation curve for the Phishing data sets accuracy against the number of hidden layers. The curve show is that the training and cross-validation curves are closest when the number

of hidden layers is at a relatively low 2. After this they slowly move apart, giving more of a chance of over fitting.

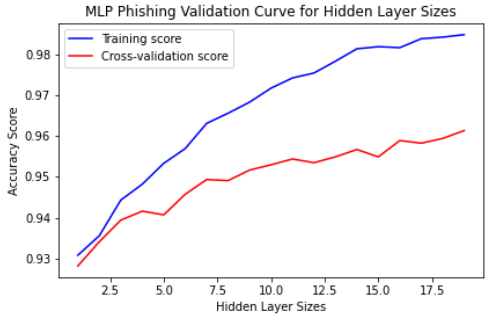

Figure 12: MLP Phishing Validation Curve

Figure 18 shows the learning curves of the Phishing data set for contrasting numbers of hidden layers. This shows that for a small number of samples, the algorithm is very inaccurate for cross-validation, but when the number reaches around 200, it stabilizes around an accuracy of 0.9. Interestingly, the training accuracy drops with more samples. The different numbers of hidden layers has seems to have little effect on the cross-validation accuracy, so a small number would be better as it is more efficient.

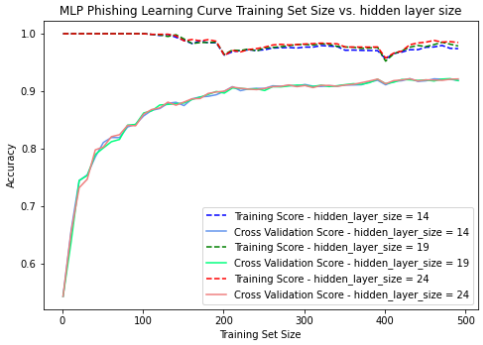

Figure 13: MLP Phishing Learning Curves

Multi-Layered Perceptron - Red Wine Dataset

Running the Red Wine data through the Scikit Learn Multi-Layered Perceptron (MLP) Gridsearch with the same parameters as above gave the highest f1 score with the following values: an activation function of **Relu**, the number of hidden layers came out as **15**, the alpha was **0.001**, the maximum iterations was **1500** and it chose an **Adaptive** learning rate. The number of hidden layers was later changed (see validation curve below) to 7 with better results.

The following table and confusion matrix in figure show the overall statistics for the MLP:

	precision	recall	f1-score
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.64	0.72	0.68
6	0.52	0.56	0.54
7	0.52	0.40	0.45
8	0.00	0.00	0.00
accuracy			0.58
weighted average	0.55	0.58	0.56

Accuracy - This was a relatively low at 0.58.

Recall (weighted average) - Another low at 0.58.

Originally, the GridSearch gave 15 as the best value for achieving good results for the number of hidden layers. Upon seeing the validation curve, it shows that higher values could be achieved with a value of 7. When this was tested, better results were achieved and so all of the charts were recreated with this value.

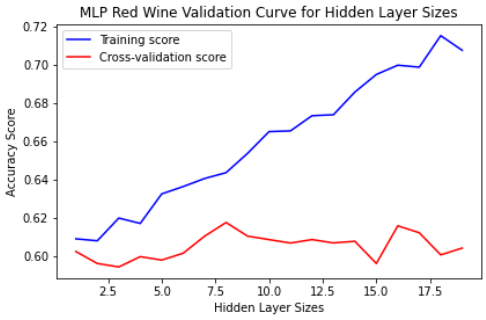

Figure 14: MLP Red Wine Validation Curve

The learning curves for the Red Wine data sets accuracy against increasing Training Set sizes for 3 different hidden layers sizes. The cross-validation curves show that increasing amounts of data allow more accurate results and the number of layers are all quite similar. The

training curves drop in accuracy as the number of samples increases until somewhere between 300 and 400, when the accuracies seem to stabilize. Increasing the number of hidden layers increases the accuracy of the training.

Figure 15: MLP Red Wine Learning Curves

Bias and Variance - Multi-Layer Perceptron

The validation curve for both of the datasets show similar patterns in that both the training and the cross-validation curves keep rising and so a little experimentation is was needed to find the correct balance between bias and variance. The Gridsearch did not output the best results and so the experimentation was performed manually. Having too many layers would mean that the algorithm would overfit the data, but too few layers would result in a model that is too simple.

Boosting - Phishing Dataset

The AdaBoost algorithm uses a number of 'weak learners' together to produce a results that often gives the highest accuracy of all of the others.

The parameters that will be ranged include the learning rate and the arguably most important, the number of estimators. Here the base estimator used will be the Decision Tree Classifier, whose parameters will be the same as those found above. The number of estimators tested here are spaced values ranging from 1 to 100. The learning rates tested are from

$$x^2/100 \quad (1)$$

with x ranging from 1 to 7. The best values were found to be **0.16** for the Learning Rate and **30** for the number of estimators.

	precision	recall	f1-score
-1	0.95	0.94	0.95
1	0.96	0.96	0.96
weighted average	0.95	0.95	0.95
accuracy			0.95

False Positives - The number of websites that get through the filter was quite low (although not the lowest) relatively of 0.056. **False Negatives** - These are the wanted websites that are caught in the filter, with an OK probability of 0.94. **Accuracy** - This shows the number of websites we predicted to be true correctly with a high probability of 0.96. **Precision (weighted average** - An reasonable probability of 0.96.

The validation curve for the Phishing data sets accuracy against the maximum depth. The cross-validation and training curve accuracies rise very quickly until just less than 20 estimators and then slowly move apart. The Gridsearch found the most accurate value for the number of estimators at 30, which from the chart seem to find a good balance between bias and variance.

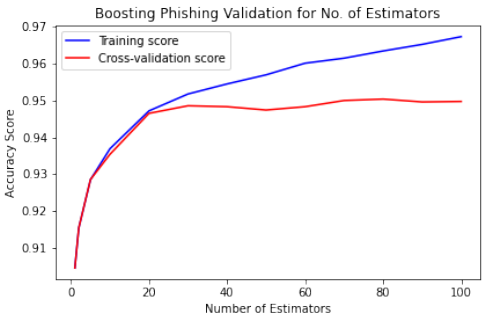

Figure 16: AdaBoost Phishing Validation Curve

Figure 24 shows the learning curves of the Phishing data set for contrasting numbers of estimators. It shows that the algorithm gets its best accuracy with only 100 samples. It also confirms the findings from the validation curve in that values around 20 are OK, but much higher than this and overfitting begins to occur.

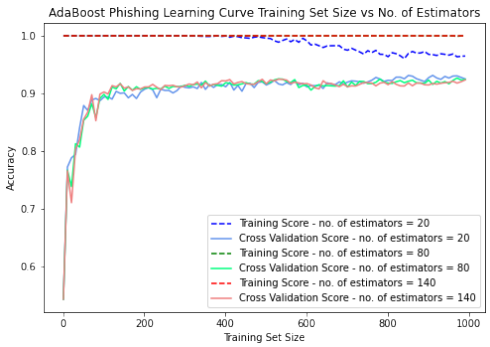

Figure 17: AdaBoost Phishing Learning Curves

AdaBoost - Red Wine Dataset

Running the Red Wine data through the Scikit Learn Boosting Gridsearch with the same parameters as above gave the highest f1 score with the following values: Learning rate: **0.16**. As explained below, the number of estimators was the same as with the Phishing dataset at **80**, but a value of **30** was eventually discovered to achieve higher results.

	precision	recall	f1-score
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.60	0.69	0.64
6	0.50	0.52	0.51
7	0.55	0.37	0.44
8	0.00	0.00	0.00
accuracy			0.55
weighted average	0.52	0.55	0.53

For the Red Wine dataset, probably the most important numbers are: **Accuracy** - This was a very low 0.55.

Recall (weighted average) - 0.55. These values are the lowest achieved for any of the algorithms

Originally, the GridSearch gave 80 as the best value for achieving good results for the number of estimators. Upon seeing the validation curve, it shows that higher values could be achieved with values around 30 to 40. When this was tested, it was to be the case and so all of the charts were recreated with this value. After this value the cross-validation accuracy drops slowly.

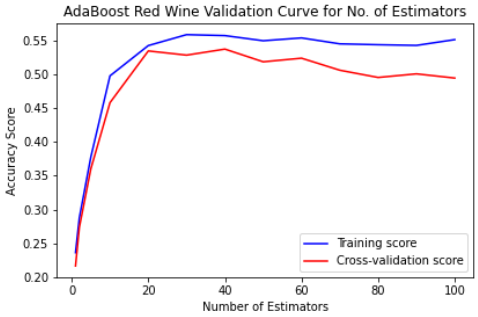

Figure 18: AdaBoost Red Wine Validation Curve

The learning curves for the Red Wine data sets accuracy against increasing Training Set sizes. This shows a similar pattern to the Phishing dataset, so no further explanation is needed.

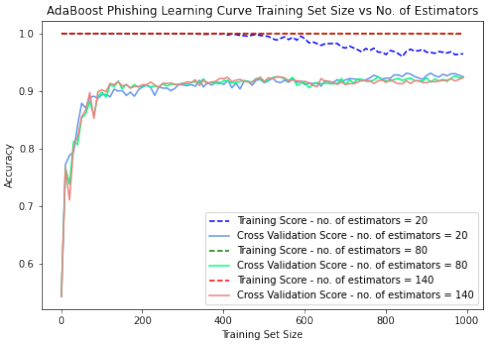

Figure 19: AdaBoost Red Wine Learning Curves

Bias and Variance - AdaBoost

As stated earlier, the Phishing Validations curves both rose and a balance was needed to be found between the bias and variance. For the Red Wine dataset, finding the balance was much easier, even when the Gridsearch returned less than optimal figures as as the curves found a maximum that was not overfit and then slowly declined.

Support Vector Machine - Phishing Dataset

The Support Vector Machine algorithm attempts to create the best dividing line between the different classifications.

The parameters that were ranged include C (the regularization parameter), a variety of kernels (Linear, Poly, RBF and Sigmoid), 3 different tolerances and a range of gammas from 1 to 0.0001. The GridSearch algorithm gave out the Regularization parameter C at **0.501**, the kernel was **Poly**, the tolerance came out at **1e-08**. The Gamma was originally from the GridSearch, but upon examining the validation curve, it was found that a value of 0.1 achieved better results.

	precision	recall	f1-score
-1	0.96	0.94	0.95
1	0.95	0.97	0.96
weighted average	0.95	0.95	0.95
accuracy			0.96

False Positives - The number of websites that get through the filter was quite low (although not the lowest) relatively of 0.059. **False Negatives** - These are the wanted websites that are caught in the filter, with an OK probability of 0.94. **Accuracy** - This shows the number of websites we predicted to be true correctly with a very high probability of 0.96. **Precision (weighted average** - An reasonable probability of 0.95.

The validation curve for the Phishing data sets accuracy against Gamma. The accuracy hits 0.93 with cross validation and 0.94 for training at a Gamma of 0.1 and then the two values slowly start to move apart, giving more and more chance of overfitting. For this reason, 0.1 appears to give the best accuracy for the Gamma.

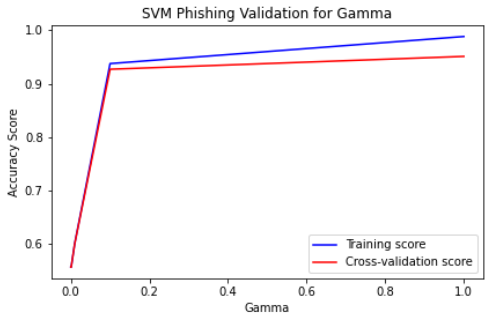

Figure 20: SVM Phishing Validation Curve

Figure shows the learning curves of the Phishing data set for contrasting Gamma, which generally confirm the findings from the validation curve. The accuracies are relatively low for a Gamma of 0.1, stabilizing at around 0.82 for cross-validation and 0.9 for training. Similar accuracies stabilize close to 0.88 for Gammas of both 1 and 10, but there training curves show them to be overfitted.

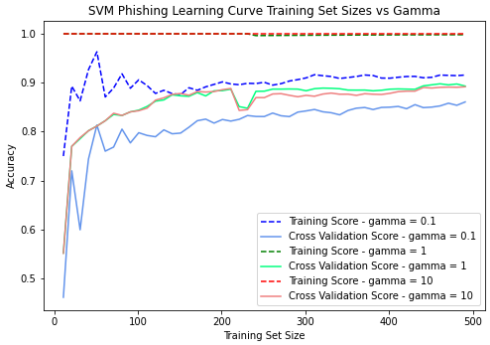

Figure 21: SVM Phishing Learning Curves

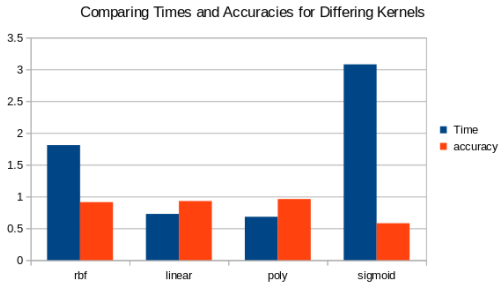

Figure 22: SVM Phishing Kernel Comparison

In figure 22 four of the kernels (RBF, Linear, Poly and Sigmoid) have been compared with respects to the algorithms running times and their accuracy of the Phishing dataset. All of the kernels produce similar accuracies just above 0.9 except for the Sigmoid which performs significantly worse with an accuracy of just 0.58. The Linear and Poly kernels are the fastest, both below 1 second. RBF comes in third position at around 1.7 and once again the Sigmoid performs worse than the others at 3.08 seconds.

Support Vector Machine - Red Wine Dataset

Running the Red Wine data through the Scikit Learn Support Vector Machine (SVM) Gridsearch with the same parameters as above gave the highest f1 score with the following values:

Regularization parameter C at **1.501**, the kernel was **RBF**, the tolerance came out at **1e-08**. The Gamma was **1**.

	precision	recall	f1-score
3	0.00	0.00	0.00
4	0.00	0.00	0.00
5	0.67	0.71	0.69
6	0.58	0.67	0.62
7	0.72	0.52	0.60
8	1.00	0.20	0.33
accuracy			0.63
weighted average	0.62	0.63	0.62

Accuracy - This was a quite low 0.63
Recall (weighted average) - 0.63.

For the SVM validation curve for the red wine dataset, the gamma gives it's highest cross-validation accuracy around 0.1 and then drops slowly drops as it goes higher. The testing also has a sharp incline to 0.1, but then keeps increasing at a lesser incline to 1 and again increases at an even smaller incline to 10. It may seem that a gamma of 0.1 might achieve better results than that given by the Gridsearch algorithm, but when the algorithm is run with this value it achieves lower recall, precision and accuracy than for a gamma of 1, which must be the best bias/variance balance.

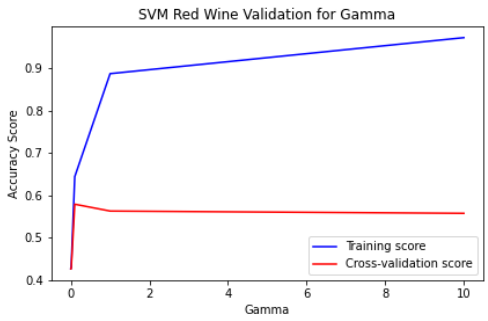

Figure 23: SVM Red Wine Validation Curve

The learning curves for the Red Wine data sets accuracy against increasing Training Set sizes for different values of gamma. There does not appear to be a gamma that does better when the size of the training set is below 300, but after

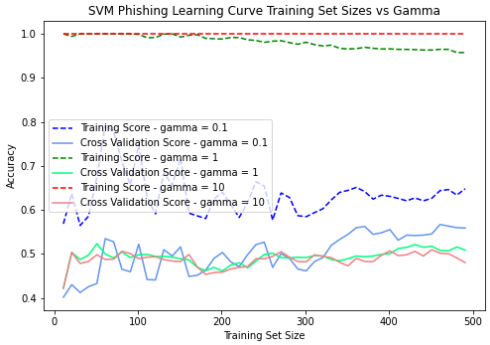

Figure 24: SVM Red Wine Learning Curves

300 it again seems that the best cross-validation accuracies are achieved with a gamma of 0.1.

Bias and Variance - Support Vector Machine

The validation curves for both datasets were similar in that the accuracies rose up quickly to 0.1 and then the training curve kept increasing. In the case of the Phishing dataset the cross-validation also kept rising, albeit to a lesser extent, whereas the Red Wine began to drop slightly. Finding a balance between 0.1 and 1 took a little bit of manual experimentation to check the Gridsearch had found the best bias/variance balance.

Summary

Decision Trees
Decision Trees have very fast running times. Their f1 score and accuracy are good at 0.95 and 0.96 respectively for the large Phishing dataset and achieves middling results for the Red Wine dataset at 0.58 and 0.58. The most successful maximum depths for the trees seem to be close to 9 for both datasets. A big problem with this algorithm is that it is very easy to overfit

K-Nearest Neighbours
The KNN algorithm was quick, easy to understand and achieved the best f1 scores and accuracies overall. the f1 score for the phishing dataset was similar to all of the others at 0.95 and 0.96, but it out-performed on the smaller red wine dataset with 0.67 and 0.69 respectively.

Without further information it is difficult to give conclusive information about the bias and variance.

Multi-layer Perceptron

The MLP algorithm achieved middling scores which is surprising when the algorithm is complex and takes a relatively long time to run. The f1 score for the phishing dataset was similar to all of the others (although marginally higher than all of the others) at 0.96 and 0.96, but it only achieved 0.56 and 0.58 for the Red Wine. Manual balancing of the parameters was required to find the best balance between bias and variance as the Gridsearch did not output the best results.

AdaBoost

The Boosting Algorithm was relatively quick. It was expected that this algorithm would do well, but it received the lowest f1 score and accuracy for the Red Wine data at 0.53 and 0.55. The Phishing numbers came out similar to the others at 0.95 and 0.96 respectively. For AdaBoost, finding the correct bias/variance balance took a little experimentation between the point on the validation curve rose quickly and the point when overfitting occurred.

Support Vector Machine

The SVM algorithm was very slow and complex to understand, but they did achieve some of the highest results. The f1 score for the phishing dataset was similar to all of the others at 0.95 and 0.96, and it achieved 0.62 and 0.63 for the Red Wine. A little experimentation was needed with this algorithm for both datasets to find whether the best bias/variance balance occurred at a Gamma of 0.1 or 1.

References

[1] Yildrim, S. (2020, May 21). How to train_test_split : KFold vs StratifiedKFold - Towards Data Science. Medium. https://towardsdatascience.com/how-to-train-test-split-kfold-vs-stratifiedkfold-281767b93869

[2] sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.24.1 documentation. (n.d.). SKLearn. Retrieved February 22, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html