Assignment 3

Report summary

Three machine learning approaches (support vector machines, neural networks, and boosting) were tried to see which approach could predict the quality of the wines in the data most successfully.

The approaches were similar in their predictive power on this particular data set, however the support vector machine with a radial boundary slightly outperformed the other approaches with the best model having a success rate of about 79.2%, which is about 0.5% better than the boosting model, and about 2.2% better than the neural network model.

Exploratory Data Analysis

The data was first explored to develop an understanding about the relationship between the quality of the wines, and their composition. This was done mainly to help guide the model building.

A linear discriminant analysis was performed to see whether the data was linearly separable in terms of quality. This is a particularly useful question to answer for determining the best type of decision boundary the support vector machine will use.

Figure 1 Shows an LDA of the data where the response variables are the colour and quality of the wine. The wines are almost perfectly linearly separable by colour, however the separation between the wines in terms of quality is harder to distinguish. A linear boundary or simple non-linear boundary may work reasonably well as a classifier for quality. The rightmost observation in the plot is an extreme outlier and was removed after determining this was a result of a data input error (see Appendix 1 for details).

In all the following machine learning approaches, the data set was normalised before being used to train the models. This was to keep all the variables within approximately the same range in order to stop large valued variables dominating over the rest and producing biased models.

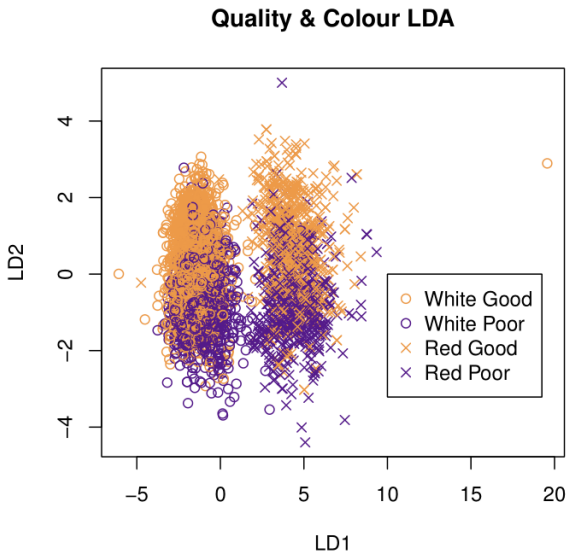

Figure 1: Shows an LDA of the wine data where the colour and quality of the wines are the response variables, and the independent variables make up the composition of the wines. The separation of the wines for colour is very apparent, and less apparent for the quality.

Support Vector Machine Model Tuning

To find the best SVM model for the data, three boundary shapes, `linear`, `radial`, and `polynomial` were tested, with varying values of two regularization parameters, `gamma` and `cost`. Since the parameters `gamma` and `cost` are scale parameters, the chosen values to test followed a logarithmic scale. Also, to prevent over-fitting the data, a coarse grid was used so that the parameters were not finely tuned to the data.

For the `linear` boundary, the set of values tested were `cost` $\in \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$. The `linear` boundary does not require the `gamma` parameter. The best parameter found was `cost` $= 2$, and the corresponding cross validation accuracy was 74.7% (3 s.f). This is okay and is about what is expected, given the results of the LDA.

For the `radial` boundary, the set of values tested

were `cost` $\in \{1, 2, 4, 8, 16, 32\}$ and `gamma` $\in \{.0.0001, 0.001, 0.01, 0.1, 1\}$. The best parameter pair was `cost` $= 2$ and `gamma` $= 1$, and the corresponding cross validation accuracy was 79.2% (3 s.f). This is a significant increase in accuracy over the `linear` boundary.

Lastly, for the `polynomial` boundary. To reduce the demand of computational power needed to perform a grid search with a polynomial boundary, a random grid search was performed to find a good starting selection of parameters to fine tune. The set of values of which the random selection were taken from were `cost` $\in \{1, 2, 4, 8, 16\}$, `degree` $\in \{2, 3\}$, and `gamma` $\in \{0.0001, 0.001, 0.01, 0.1, 1\}$. The best parameter triplet was `cost` $= 4$, `degree` $= 3$, and `gamma` $= 0.1$. Since the best `cost` and `gamma` values are not the bounds of the sets, there was no need to search further for these. Since `degree` $= 3$ was the best, it was compared with `degree` $= 4$ using the best `gamma` and `cost` parameters; `degree` $= 3$ was found to outperform `degree` $= 4$ slightly and gave an accuracy of 75.1% (3 s.f).

It is unlikely that the `radial` model with parameters `cost` $= 2$ and `gamma` $= 1$ could be improved on without some work and fine tuning, so it was chosen as the best SVM model for classifying the wines by quality with a approximate 79.2% prediction success rate.

Neural Network Model Tuning

First, a simple neural network consisting of 12 input nodes, 2 output nodes, no hidden layer, and a `sigmoid` activation function was tested first to obtain a baseline accuracy value to improve upon. The `SGD` optimizer was used for the model as the training and validation loss converge slower and smoother than the `rmsprop` and `adam` optimizers; this makes easier to spot under and over-fitting. The validation accuracy for this model was around 73.5% and converged after around 40 epochs. A batch size of 32 was used for a balance between convergence and performance. The training and validation loss were close in value which suggests the model is under-fitting the data (see Appendix 2).

A nine node hidden layer with a `relu` activation function was added to the model to combat the under-fitting problem. After 100 epochs the validation accuracy improved to approximately 75.2%. The validation and training loss were still close, and the variation in the losses was very small which indicates an insensitivity to noise, and so the model could still be under-fitting the data.

Next, a shallow version of the network with one large hidden layer consisting of 60 nodes was compared against a deeper version of the network with 3 hidden layers, each with 6 nodes. All the hidden layers mentioned again used a `relu` activation function.

After 100 epochs the shallow model had a validation accuracy of around 76% which is a slight improvement; the training and validation loss diverge slightly and are no longer close in value to each other which suggests the model is likely well fit to the data and should not be added to. The deep model only achieved a validation accuracy of around 74.5% and the validation loss converged completely after around 50 epochs.

Two copies of the shallow model were compared, one with 30 nodes in the hidden layer, and the other had 120 nodes in the hidden layer. Both of the models showed no improvement in accuracy over the 60 node model.

Lastly, a small hidden layer with 6 nodes and `relu` function was added to the 60 node shallow model to compare. The extra layer did yield an improvement in accuracy, and the new model had a validation accuracy of around 77% after 100 epochs. This model was taken as the best model.

Figure 2 & 3 show the loss graph for the new model, and the accuracy of the model respectively. There is a slight divergence in the training and test loss which could be an indication that the model is over-fitting slightly, however the training and validation accuracy do not diverge and are close together which suggests the model is not over-fitting. Overall the model looks to be a good fit to the data.

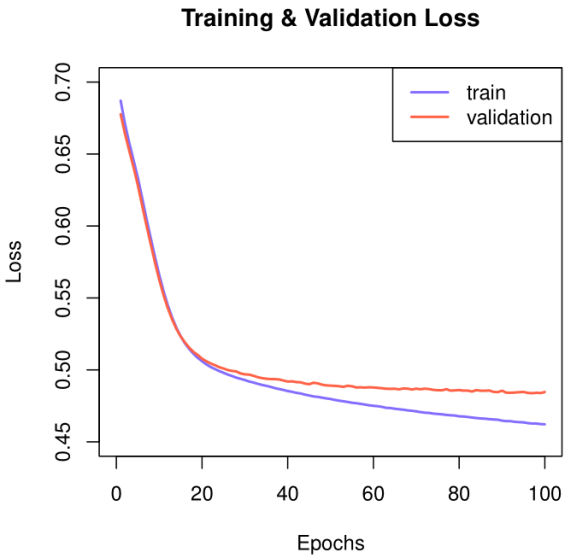

Figure 2: Shows the training and validation loss for the final network. The losses diverge slightly indicating a possibility of over-fitting. The validation loss after 100 epochs is about 4.9%.

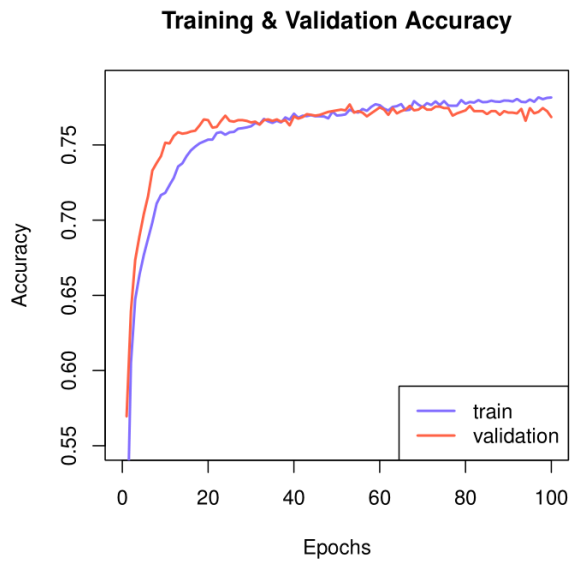

Figure 3: Shows the training and test accuracy of the final model. The model under-fits the data below 65 epochs and is a good fit at 100 epochs. The accuracy at 100 epochs is about 77%.

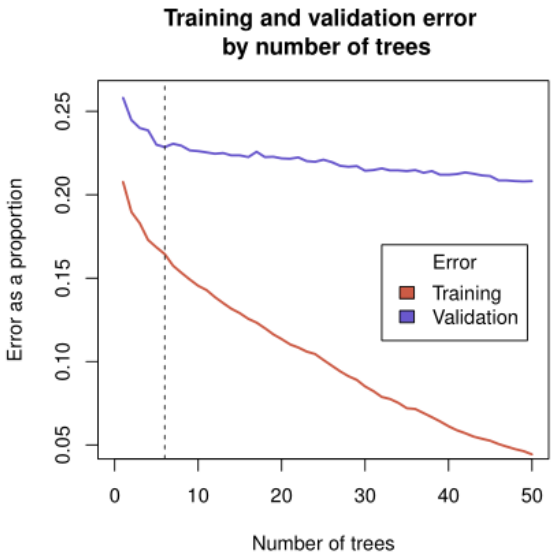

Figure 4: Shows the divergence of the training and validation errors change as more trees are added. The black dotted line represents a model with six trees. After six trees the divergence increases substantially.

XGBoost Model Tuning

The default `xgboost` model was cross validated using the default parameters to see what the approximate baseline accuracy is before parameter tuning. Figure 4 shows the approximate training and validation error of the model as the number of trees added to the model increases. After about six trees the errors begin to diverge (marked by the dashed black line); this is an indication that adding more than six trees may be over-fitting the data as the training accuracy is not generalizing to out of sample data. The optimum number of trees for the default model for the data is therefore about six. The convergence is extremely fast, timed at approximately 0.215 seconds, and the accuracy is about 76.7%. This already performs as well as the NN model.

The learning rate for the model was chosen before parameter tuning. The learning rate is a trade-off between performance and accuracy. The more trees that can be included in the model, the more accurate the model predictions will be, but the performance will also be lower with the addition of more trees. Also, if the model learns too quickly for the number of trees the model contains, then over-fitting of the training data will occur and the model accuracy will not generalise. Therefore the learning rate needs to be as low as possible so that more trees can be included without over-fitting the data, all while maintaining good performance.

Divergence of the training and validation errors in cross validation is a signal of over-fitting. The goal then is to choose a learning rate that enables many trees to be used while keeping the divergence of the errors as low as possible.

Figure 5 shows the difference between the training and validation errors for 5 different learning rates as more trees are added to the model. As the learning rate decreases, so does the divergence between the errors, and below 0.01 there is diminishing returns of reduced divergence for model performance. A learning rate of 0.01 looks like a good trade-off between performance and model robustness to over-fitting; however it must be said that increasing the `maximum depth` parameter for the trees will increase the divergence of the errors for all learning rates over the same range of trees. This is because if trees are more complex, then it requires less of them to fit the data. Therefore over-fitting occurs earlier. This problem can be dealt with somewhat with the regularization parameters.

The next parameters to be tuned control the maximum depth and the splitting behaviour of the parent nodes, i.e the tree structure. This was done by trying 3 wide-ranging values for the maximum depth, and 3 wide-ranging values for the minimum number of observations needed in the child nodes before the parent node can split. Once the best pair was found, the range of values for each parameter was narrowed around these values and tested.

Figure 6 shows the validation error for 10 samples of each pair of parameters. The best structure is a tree with a maximum depth of 10 splits, and a minimum

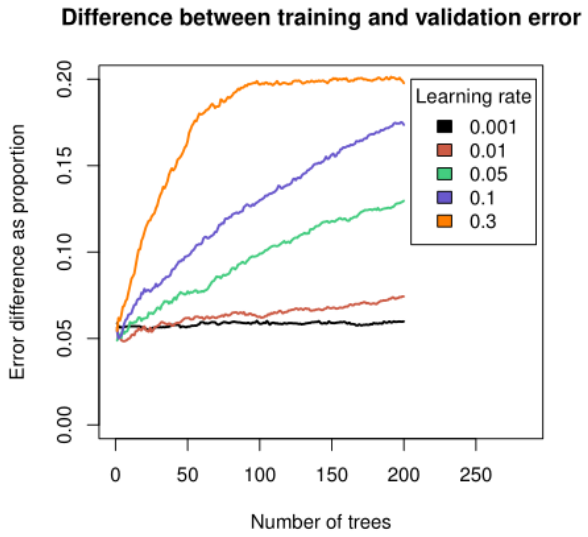

Figure 5: Shows the difference between the training and validation errors as more trees are added to the default model for 5 different learning rates. As the learning rate decreases, so does the divergence of the errors.

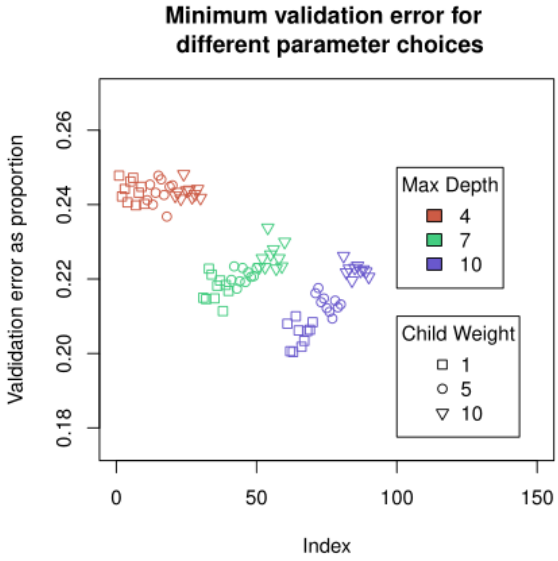

Figure 6: Shows the minimum validation error for 9 combinations of depth and child weight parameters and 10 samples for each combination. A maximum tree depth of 10 and minimum child weight of 1 generally gives the lowest validation accuracy.

of 1 observation needed in the child nodes before the parent can split. Figure 11 shows 10 samples for the narrowed range for each parameter; a maximum depth of 11 seems to performs slightly better than 10, and a child weight of 1 remains the best.

Figure 12 compares 11, 12 and 13 for the depth; there is no real performance increase by increasing the depth past 11, so 11 was taken as the best parameter for the maximum depth.

The next group of parameters control the robustness of the model to over-fitting. `Sub sample` controls the proportion of data used to construct each tree, `column sample` controls the amount of variables used for each tree, and `gamma` controls the penalty applied to over-fitted models. Using the same method as the previous section to tune these parameters, 27 triplets covering a wide range of the parameters were tested, and then the ranges were narrowed until the best triplet was found.

Figure 7 shows 7 samples for each triplet and the minimum validation error for them. The best performing triplet is `gamma` = 0.5, `sub sample` = 14, and `column_sample` = 0.6. Figure 13 shows the narrowed ranges for these parameters and shows no obvious difference between the new set of triplets. Since `gamma` helps prevent over-fitting, a value of 0.5 was chosen as the best `gamma` value; then a `column sample` of 0.7 was chosen as it seems to perform slightly better that the other values; and lastly a `sub sample` value of 0.9 was chosen as the best value as smaller values can also help over-fitting.

Figure 8 shows the training and validation error for the tuned model. The validation error converges after adding about 20 trees and the difference between the training and test errors is large which indicates the model is over-fitting the data quite a lot. In an attepmt to reduce the over-fitting, the maximum depth parameter was reduced from 11 to 8. This decreased the accuracy of the model by around 1% to about 78.7%, but also reduced the difference between the errors by about 5%.

The fully tuned `xgboost` model therefore classifies the wines with approximately a 78.7% success rate. This model with parameters `gamma` = 0.5, `max_depth` = 8, `column sample` = 0.7, `sub_sample` = 0.9, and `child_weight` = 1 is therefore chosen as the best model.

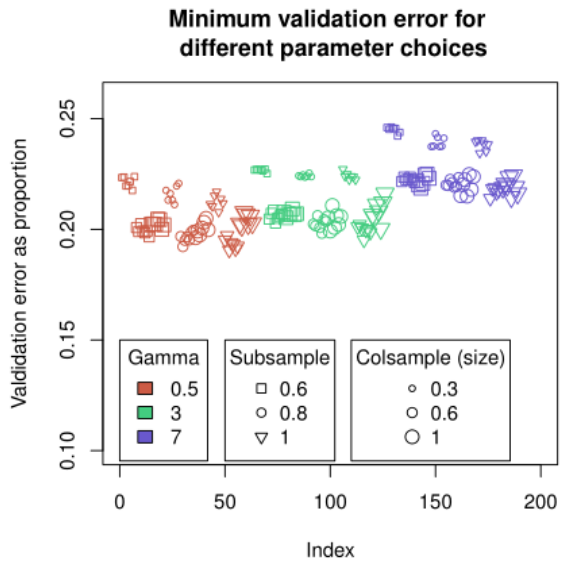

Figure 7: Shows the minimum validation error for 18 triplets of parameters and 7 samples for each triplet. There is no obvious difference between the triplets.

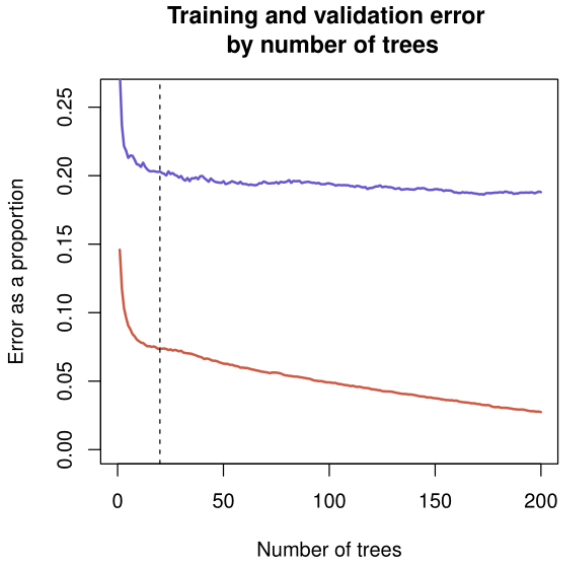

Figure 8: Shows the minimum training and validation error for the tuned model. The validation error has converged after adding about 20 trees.

Appendix 1

Figure 9 shows the data for the outlying observation seen in Figure 1. The mean of the residual sugar values is 5.368, and the 3rd quartile is 8.000. The residual sugar value for the observation is 65.8. This looks like it could be a error in the input, where the 6 key was accidentally hit when the person inputting was trying to press 5, or vice versa. It is also a possibility that the value could be real as the density of this particular observation is also extreme and correlated to the sugar value. In any case this observation is not representative of the rest of the data and it's influence, if real, is not important for the purposes of the report, and so it was removed from the data.

```
> wine.data[1485,]
      X fixed.acidity volatile.acidity citric.acid
1485 1485          7.8            0.965         0.6
     residual.sugar chlorides free.sulfur.dioxide
1485           65.8     0.074                   8
     total.sulfur.dioxide density   pH sulphates alcohol colour
1485                  160 1.03898 3.39      0.69    11.7      0
     quality
1485       1
```

Figure 9: Shows the data for the outlying observation (rightmost observation) in Figure 1. The residual sugar value is an extreme outlying value for the data.

Appendix 2

See Figure 10.

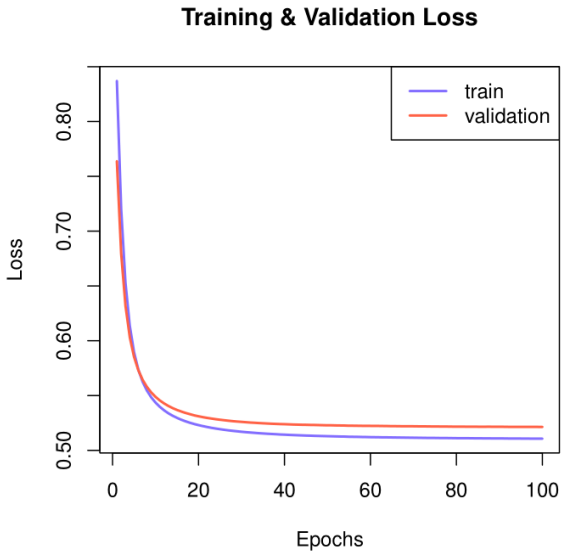

Figure 10: Shows the training and validation loss for the simple neural network. The validation loss converges after around 40 epochs. The training and validation loss are close in value which suggests the model is under-fitting.

Figure 11: Shows the minimum validation error for the narrowed range of depth and child weight parameters and 10 samples for each pair. 11 is a slight improvement over 10.

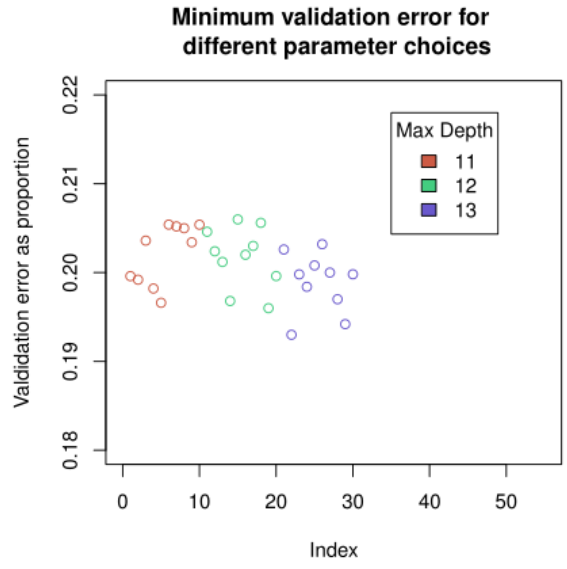

Figure 12: Shows the minimum validation error for maximum tree depths of 11, 12 and 13. A maximum depth of 12 is not an improvement over 11 for the model; 13 shows a possible slight improvement over 12 but this is likely to be noise.

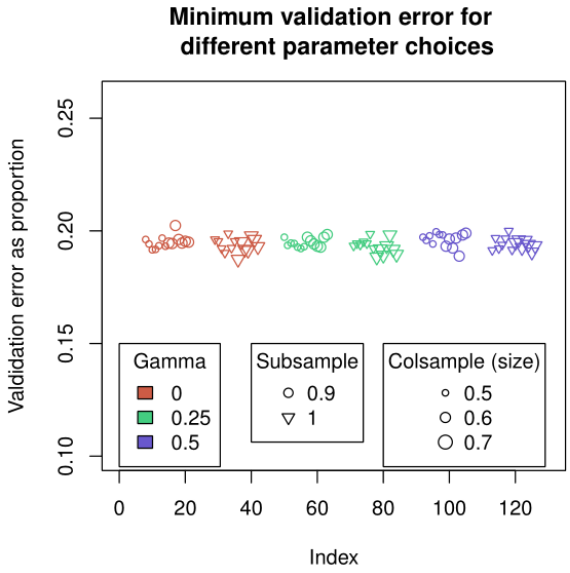

Figure 13: Shows the minimum validation error for 18 triplets of parameters and 7 samples for each triplet. There is no obvious difference between the triplets.