Project 1 – Supervised Learning Report

Selected Datasets: The 2 data sets analysed are:

1. **Abalone**[1]: This dataset was obtained from the UCI machine-learning repository. The objective is to predict the age of abalones (a kind of sea snail), based on physical measurements. The dataset has 8 attributes (categorical, real and integer), contains 4177 instances, and its output class has 29 members.
2. **Wine Quality**[2]: This is also from the UCI repository. The goal here is to model wine qualities, based on physiochemical tests. The dataset has 12 real attributes, with 4898 instances and an output class containing 11 members (2 of which have no member instances). **Figure 1** below shows the distribution of both data sets with respect to their respective output classes.

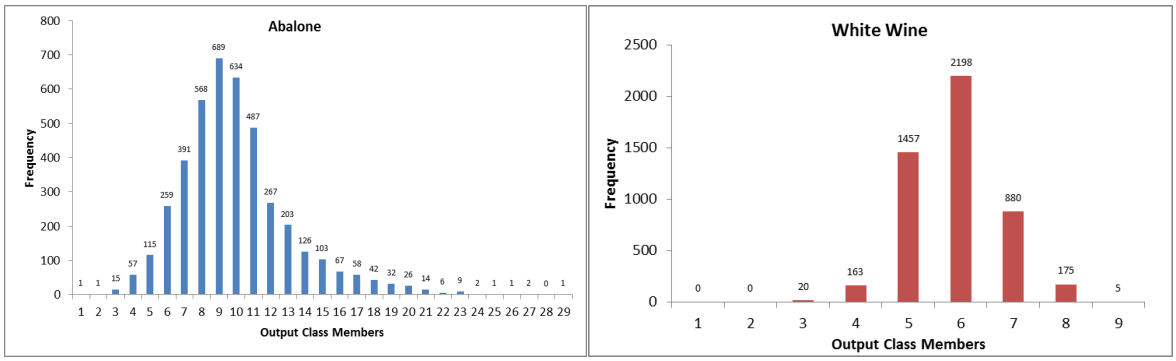

Figure 1: Output class distribution of data sets

Rationale for Dataset Selection: The objective was to pick 2 datasets with different types of features, which would produce contrasting results, whilst simultaneously showing the strengths and weaknesses of the various classification methods being tested.

From my research, I discovered that the Abalone data set is deceptively simple, yet is notoriously difficult to classify with high accuracy. The Wine dataset on the other hand is known to be fairly well behaved with regards to classification algorithms.

It is my hope therefore that analysing these datasets will yield interesting and contrasting results, and also allow for insightful analysis.

Implementation Methodology: All the experiments described in this report were carried out using **WEKA**[3] (version 3.8.1).

Data Pre-processing: The following steps were taken to prepare the data in both data sets for analysis:

- **Training & Test Sets**: First, both data sets were split into a training set (70%), and a test set (30%). All subsequent analysis was performed on the training sets, and the test sets were only used after each respective classification model had been created.

[1] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
[2] P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.
[3] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016

- **Attribute Selection**: To make analysis more manageable, the Wine training data was analysed using WEKA's **CfsSubSetEval** attribute evaluator, coupled with the **GreedyStepwise** search method (see *README.txt* file). This resulted in **9 significant attributes** being selected (including the output class). The other attributes were then stripped from both the training and test sets.

  For the Abalone training data, running this same analysis resulted in only 2 attributes being highlighted as significant. However, as this would severely limit the benefits of any experiments conducted, all 8 attributes were used nonetheless.

- **Normalization and Data Conversion:** Some of the classifiers used (Decision Trees, Neural Networks and SVM) require that data be normalized. As such, the test and training sets used with these classifiers were normalized (via the WEKA **Normalize** filter). Also, the output variables for both data sets are numeric, and had to be converted to nominal attributes. Finally, the Abalone data set has a categorical attribute (SEX) that was converted to numeric using manual one-hot encoding.

Hyper-Parameter Tuning: This critical activity was performed using a combination of WEKA's automated grid search (via its **CVParameterSelector** classifier) and some manual experimenting with values. Grid search would clearly produce optimal hyper-parameters, but is costly in terms of time. Furthermore, grid search is practically computationally intractable for my computing resources, especially for some of the iterative classifiers.

A problem I experienced in using the CVParameterSelector classifier is that only the selected best hyper-parameters are output, and no data on how this selection is made is displayed or made available. This makes it impossible to show the process through which these values were arrived in this report.

Manually experimenting with parameters on the other hand required that I iteratively set the parameter value(s), train a learner with these parameters, and observe the trends in classification accuracy as the parameter value(s) are increased/decreased, until no further improvement is observed.

Table 1 below shows the final hyper-parameters selected for the experiments. All other hyper-parameters were left at default values.

Table 1: Selected Hyper-Parameters for Various Classifiers and Respective Datasets

Classifier	Hyper Parameters	Tuning Method	Range Tested	Selected Value	
				Abalone	Wine
Decision Tree (J48)	C	Grid	0.1 – 0.5	0.1	0.5
	M	Grid	1 - 50	23	1
Neural Network (MultilayerPerceptron)	M	Grid	0.1 – 1.0	0.1	0.4
	L	Grid	0.1 – 1.0	0.1	0.1
	H	Manual	a, i, o, t	i	t
	N	Manual	50 – 500	500	500
Boosting (AdaBoostM1 with J48)	I	Grid	10 - 100	20	20
	C (for J48)	Manual	0.1 – 0.5	0.1	0.1
	M (for J48)	Manual	1 - 50	3	2
SVM (SMO)	Calibrator	Manual	N/A	Logistic	Logistic
	Kernel	Manual	N/A	PolyKernel	PolyKernel
	Exponent (PolyKernel)	Manual	1 – 4	2	1
KNN (IBk)	K	Manual	1 - 300	25	1

Analysis – Abalone Dataset

- **Decision Tree:** As shown in **Table 1** above, I used a pruning confidence factor (C) of 0.1 and the minimum number of leaves parameter (M) was set to 23. These were selected as the optimal values by WEKA's CVParameterSelector classifier.

  With these parameter values, learning curves were created by iteratively increasing the size of the training set, and then training the learner both with the entire training set, and also with 10-fold cross-validation (CV). **Figure 2** below shows the resultant learning curves (RMSE and Percent Incorrect)

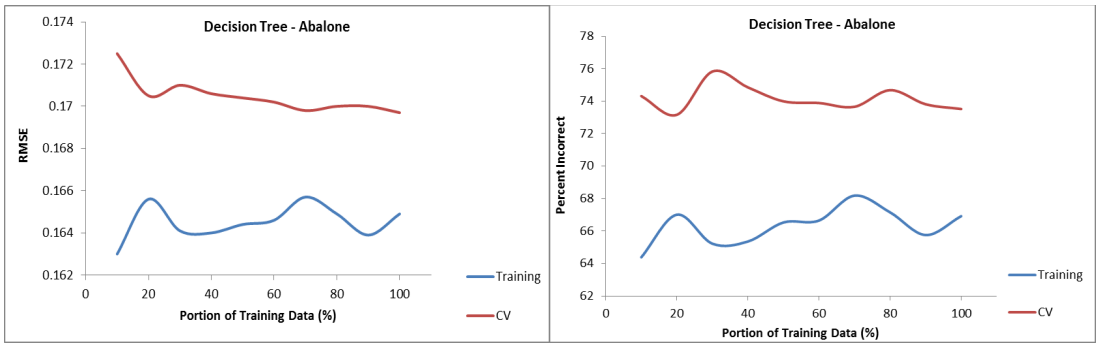

Figure 2: Decision Tree Learning Curves – Abalone

  I found these learning curves rather difficult to interpret, as it appears that both the training and CV curves stay relatively flat, and do really seem to be converging or diverging, regardless of how much training data is made available. On the one hand, the fact that the training error starts off quite high, even when only 10% of the training set is used, makes one inclined to see this as a case of high bias. However, the wide difference between the CV and training error can be interpreted as an indicator of high variance and overfitting. Quite frankly, these learning curves are inconclusive.

  What is clear however is that the learner does a pretty poor job of classifying the dataset. Given the bias that decision trees have towards top-down decision making, and considering the fact that there are 29 possible classification results, it appears that the learner is unable to make early distinctions between the various classes. Furthermore, all the attributes of the decision tree are real numbers and may be too granular, making it even more difficult for the decision tree to identify the boundaries between different output classes.

- **Neural Network**: It should be noted that the training and test sets used for these experiments were normalized, to prevent attributes with higher magnitudes from influencing the output unduly. Similar to the decision tree experiments, some hyper-parameters were selected using the CVParameterSelector classifier. However, I was able to decide on the number of iterations (N) by manually varying this parameter. **Figure 3** below shows the effect of varying the number of iterations on the RMSE and Percent of incorrect classifications (while keeping everything else constant).

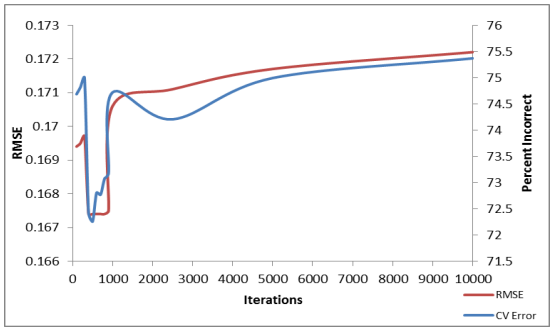

Figure 3: Parameter Selection (Number of Iterations) – Neural Network

It is quite clear from this graph that increasing the number of iterations has no appreciable effect on the accuracy of the model. It however has the negative effect of greatly increasing the amount of time required to train the learner. It is also clear that the optimal number of iterations is around 500. **Table 1** shows the other parameters selected.

With these parameters, the learners were trained using the same methodology as previously described. **Figure 4** below shows the resulting learning curves. These curves are much easier to interpret. The training error is increasing with more data, while CV error is reducing, and the curves appear to be converging. This is indicative of high variance, and suggests that the neural network has over-fit the model, and would benefit from more data. However, the error rate of the learner is still very high.

This poor result is not very surprising, as the real and continuous nature of the datasets attribute, as well as their clustering around certain classes as shown in **Figure 1** indicate that it would likely be difficult to establish a hyper-plane that splits the data into its different classes. In other words, the Abalone dataset appears not to be linearly separable.

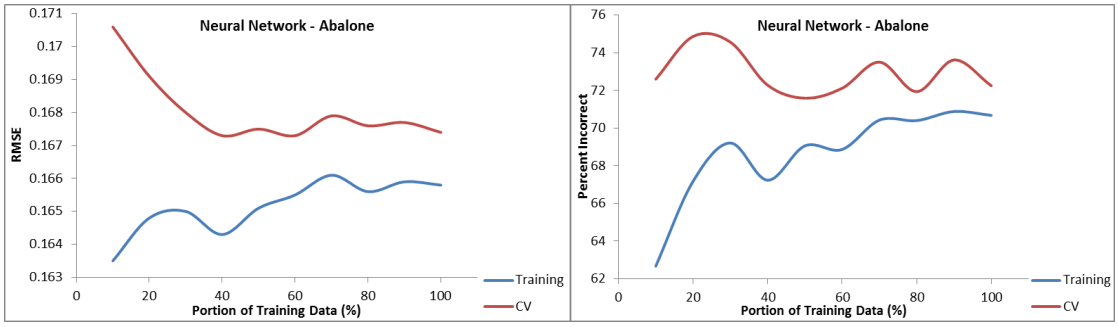

Figure 4: Neural Network Learning Curves - Abalone

- **<u>Boosting (with a Decision Tree)</u>:** The hyper-parameters used were selected based on a combination of grid search and manual fine tuning, as earlier described. The key facts to note here are that 20 iterations (I) was selected by CVParameterSelector as the optimal number of iterations (see **Table 1**). Also, I configured the decision tree to prune aggressively as suggested by Dr. Isbell in the problem brief. Learning curves for the Boosting classifier are shown in **Figure 5**.

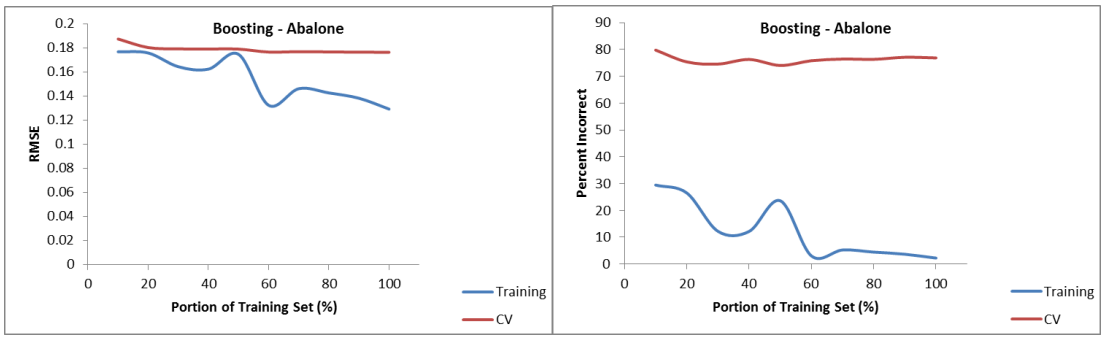

Figure 5: Boosting Learning Curves - Abalone

The result of this experiment is quite interesting, and rather counter-intuitive. One would expect that using boosting (using a decision tree) would lead to better results than that produced by a single decision tree. However, the results obtained by the decision tree are better than those generated by boosting. Also, the accuracy of training on the full data set seems to be improving as more data is made available, while the CV shows little improvement as data is added. This seems to indicate that the learner as trained with more and more of the entire training set, the AdaBoost algorithm iteratively manages to learn from

and correct its previous errors and approaches near-perfect classification. However, this accuracy is as a result of memorization (i.e. over-fitting), and doesn't translate to the cross-validated test data.

- **Support Vector Machines (SVM)**: The data here was also normalized before being used for the experiments. Popular opinion on Piazza is that SVM is an iterative algorithm. However, I was unable to find any parameter that controls how many iterations the SVM runs for in WEKA. The hyper-parameters here were selected manually, and I observed no significant difference when different calibrators and kernels were used. I therefore decided to stick to WEKAs default calibrator and kernel (see **Table 1**). For the selected PolyKernel kernel, I also observed that increasing the exponent value led to longer running times, but didn't yield much improvement in classification. **Figure 6** below shows the learning curves obtained from this experiment.

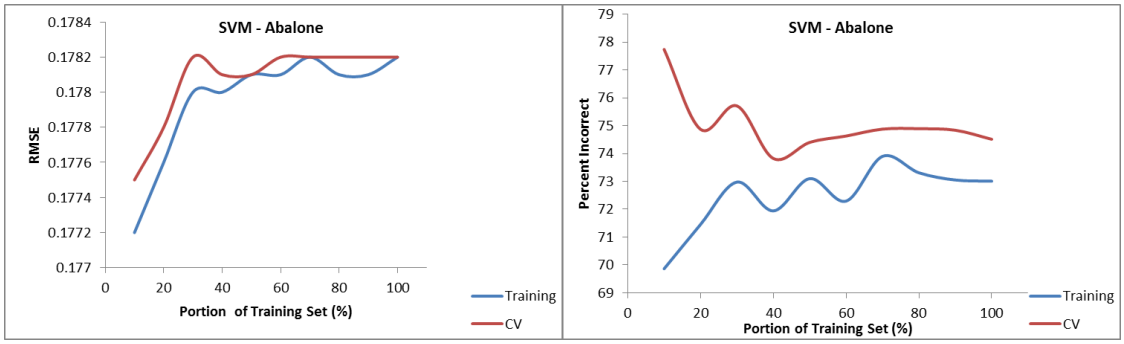

Figure 6: SVM Learning Curves - Abalone

Similar to the Neural Network training curves, these are indicative of over-fitting, and suggest that obtaining more training data might be beneficial.

- **KNN**: **Figure 7** below shows the result of hyper-parameter tuning on the number of neighbours (K) parameter, and explains the choice of 25 for K.

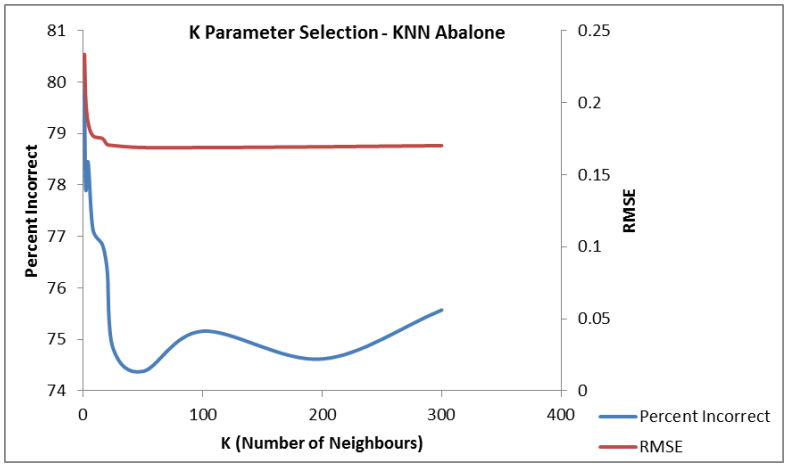

Figure 7: K Parameter Selection for KNN - Abalone

This high number of neighbours is an indication that the dataset is highly clustered, and therefore probably difficult to separate into classes. The learning curves in **Figure 8** below and the results of the KNN classification bear this prediction out.

The CV and training learning curves are both practically horizontal, and widely separated. This means that no improvement is made by the classifier as more data is added. The misclassification rate is also very

high, which indicates that the model performs very poorly. Cleary, the Abalone dataset does not lend itself very well to the KNN algorithm.

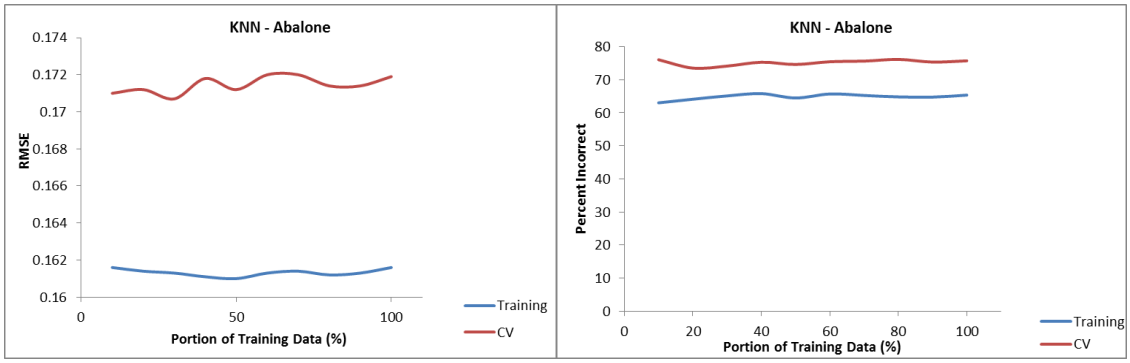

Figure 8: KNN Learning Curves - Abalone

- **<u>Conclusion - Abalone Dataset</u>**

The final experiment run with this dataset was to test the model created by each classifier with the hold-out test set. This would allow for the evaluation of their performance on data that they had never seen, and also to help determine which classifier performs best on the Abalone dataset. This was also performed iteratively, against the different models that were created with varying amounts of the training set. **Figure 9** below shows the results.

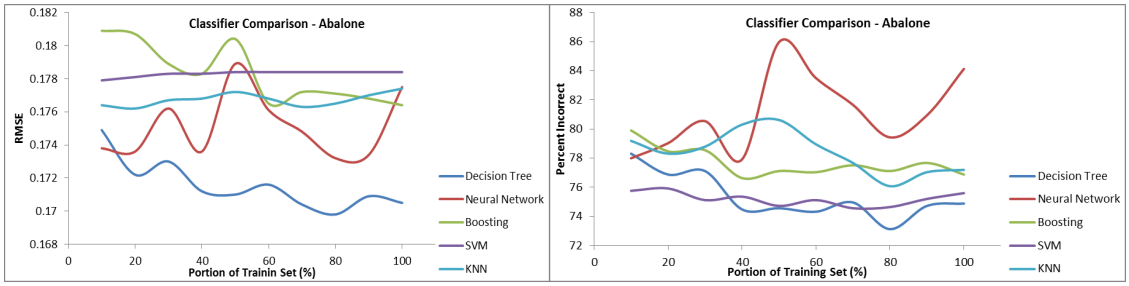

Figure 9: Comparison of Classifier Test Results - Abalone

The first challenge was to determine what metric to use to quantify performance. I settled on the **Root Mean Squared Error (RMSE)** as it a more statistically formal measure of accuracy, and it is not unduly influenced by the size of the dataset (as opposed to the percent of incorrectly classified instances, for example).

Based on this, and on the RMSE graph on the left in **Figure 9**, it is safe to conclude that Decision Trees performed best on the Abalone dataset, as it produced the most accurate results (see the right side of the RMSE graph). The other classifiers are fairly close together in terms of performance, with Boosting in 2nd place and Neural Networks and KNN more or less tied for 3rd. The worse performing algorithm on the Abalone dataset is SVM.

The chart also shows that boosting performs worse on the abalone dataset than decision trees, which, again, I find quite odd.

It is also worth mentioning that both Decision Trees and Boosting (which is based on decision trees here), seem to be improving as more data is used to train the learner. The other 3 classifiers on the other hand are fairly unresponsive, even as more data is used to train the model.

Regardless of which algorithm performed best in comparison to the others, they all performed rather poorly in the general, with the best of them (Decision Trees) achieving a correct classification rate of approximately 25.1%. The Abalone dataset's reputation is well deserved it seems.

My theory is that this is simply because there is probably no real correlation between the physical characteristics of an abalone (or at least those contained in the dataset), and its age. However, it is quite possible that there are other attributes, physical and non-physical (e.g. food consumption rates, etc.) that could be included in the dataset, which would be more useful in predicting the age of abalones.

Analysis – Wine Selection Dataset

- **Decision Tree:** The CVParameterSelector selected pruning confidence factor (C) here is 0.5, as opposed to 0.1 for the Abalone dataset. This hints at a fundamental difference in the nature of both datasets. For one, it shows that the wine decision tree model does not need to be pruned so aggressively to optimize its performance. This in turn seems to indicate that it classifies the data effectively and efficiently.

  The learning curves for this classifier are also markedly different from the previous data set, as shown in **Figure 10** below. The incorrectly classified instances drop for both the training and CV curves as the size of the training set increases. Also, both curves have relatively low error rates to start with, and both are steadily declining.

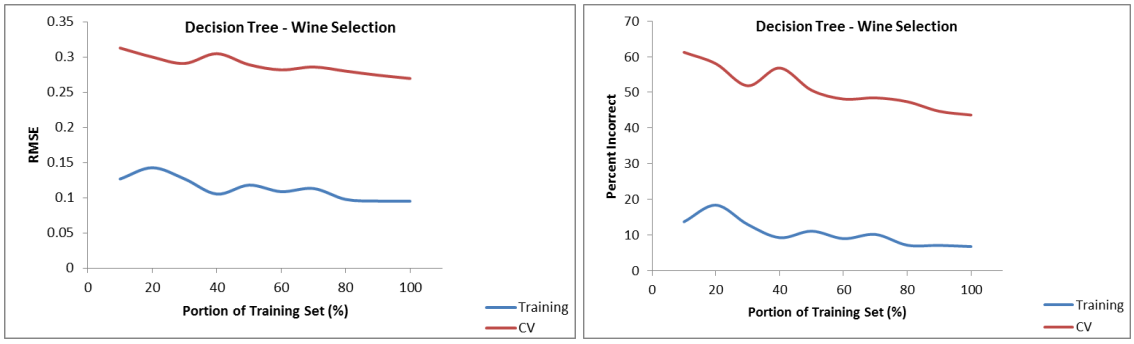

Figure 10: Decision Tree Learning Curves – Wine Selection

I believe that these trends all taken together neither indicates under- or over-fitting, but rather show that the model is accurate, and would benefit from an even larger training set.

- **Neural Network**: **Figure 11** below shows the neural network learning curves for the wine dataset. Similar to the previous classifier, it shows a model where CV and training error are moving in sync, and are fairly low, which I consider evidence of a good model

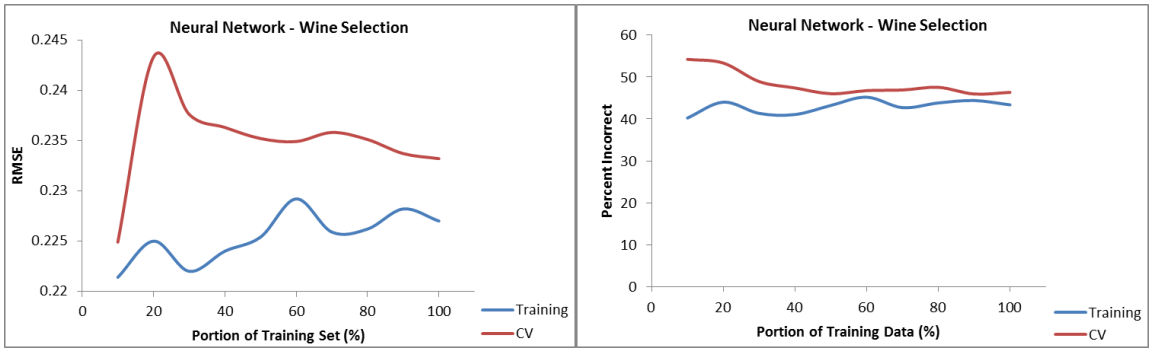

Figure 11: Neural Network Learning Curves – Wine Selection

- **Boosting (with a Decision Tree):** In Figure 12, training the boosting classifier with all training data seems to lead to perfect classification of the data, which is similar to what was observed with the abalone dataset (in **Figure 5**, where the training curve approached 100% accuracy as more data was added). This leads to me believe that boosting is very effective at learning from and correcting its errors, once it is exposed to some data. However, a fair bit of overfitting occurs on the CV curve in both datasets, which seems to be a vulnerability of boosting. However, in this case a larger training set would probably lead to very high real-world classification success.

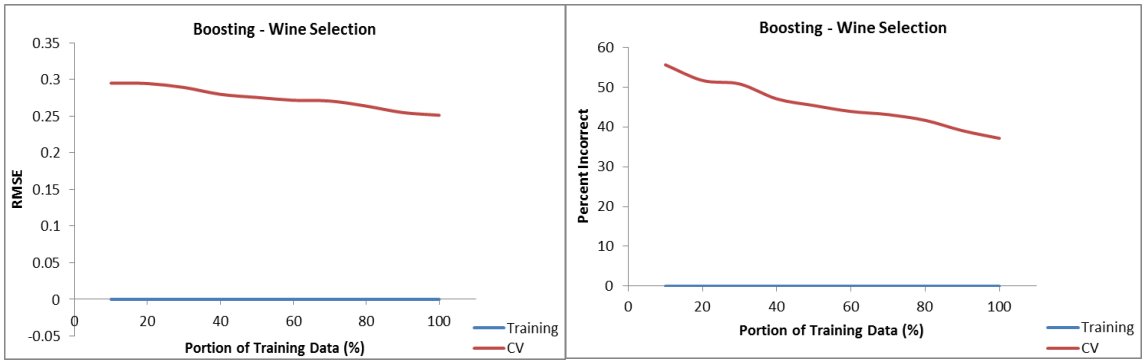

Figure 12: Boosting Learning Curves – Wine Selection

Also, it is noteworthy that boosting (with decision trees) outperforms decision trees on this dataset, which is what one would intuitively expect. A decision tree is a weak learner, and boosting should essentially magnify the performance of such learners.

As an aside, an interesting observation that I made about iterative learners is the amount of time that they take to learn as the number of iterations increases. **Figure 13** below shows that for both Neural Networks and Boosting, training time increases linearly with the number of iterations.

It also appears from **Figure 13** that boosting trains much faster than neural networks. However, this is a premature conclusion, as there are many variables that could account for this, such as the hyper-parameters used with each classifier, the characteristics of the data set, etc.

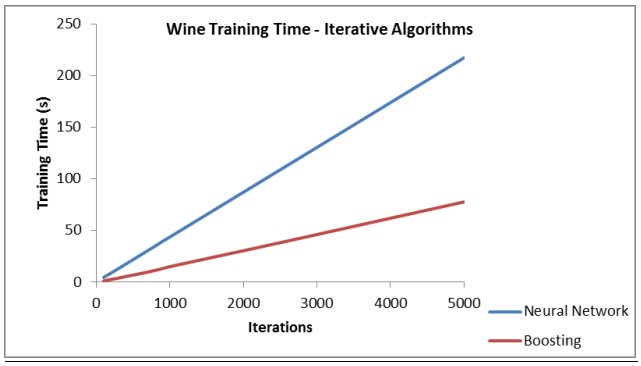

Figure 13: Training Time for Iterative Algorithms on Wine data set

- **Support Vector Machines (SVM)**: The learning curves here were also rather baffling, as RMSE and Incorrect Classification both seem to be behaving in opposite manners (see **Figure 14**).

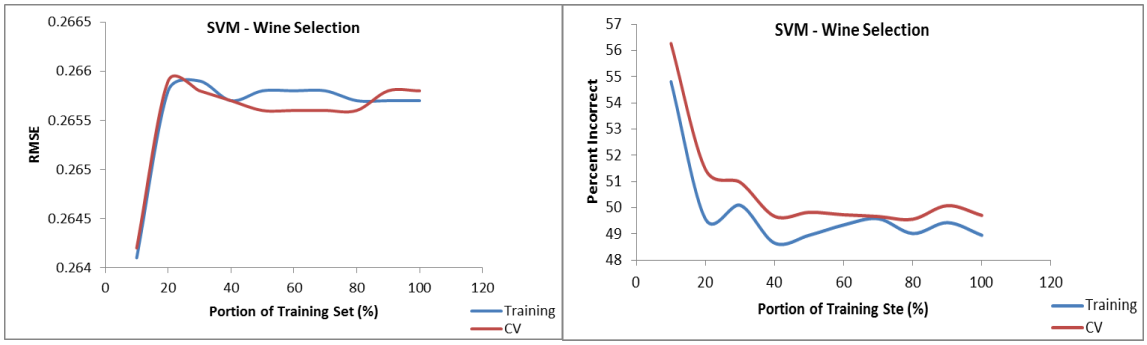

Figure 14: SVM Learning Curves – Wine Selection

However, I observed from the confusion matrix in **Figure 15** below that all the wines are classified as 2 types (classes 5 and 6), which is incorrect as is shown in **Figure 1**. Upon further investigation and research, I came to realise that the SVM implementation in WEKA (know as SMO) only works for binary output classes. Unfortunately, when I discovered this it was too late to make any changes to my analysis.

```
=== Confusion Matrix ===

   a   b   c   d   e   f   g   h   i   j   k   <-- classified as
   0   0   0   0   0   0   0   0   0   0   0 |   a = 0
   0   0   0   0   0   0   0   0   0   0   0 |   b = 1
   0   0   0   0   0   0   0   0   0   0   0 |   c = 2
   0   0   0   0   0   6  10   0   0   0   0 |   d = 3
   0   0   0   0   0  60  50   0   0   0   0 |   e = 4
   0   0   0   0   0 548 471   0   0   0   0 |   f = 5
   0   0   0   0   0 350 1176  0   0   0   0 |   g = 6
   0   0   0   0   0  47 579   0   0   0   0 |   h = 7
   0   0   0   0   0  14 114   0   0   0   0 |   i = 8
   0   0   0   0   0   0   3   0   0   0   0 |   j = 9
   0   0   0   0   0   0   0   0   0   0   0 |   k = 10
```

Figure 15: SVM Confusion Matrix – Wine Selection

- **KNN**: Unlike the abalone dataset, the optimal number of neighbours (K) selected here by the CVParameterSelector classifier is 1. This indicates straight away that the data set can easily be classified using very few neighbours, which in turn shows that the boundaries between classes are well defined in the data. **Figure 16** below shows the learning curves for the classifier.

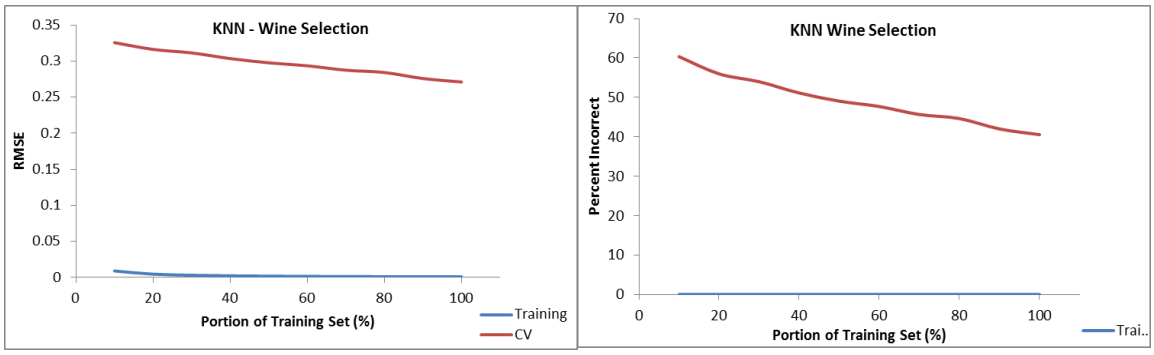

Figure 16: KNN Learning Curves – Wine Selection

This model seems to be seriously over-fit, as the training set perfectly classifies the data, while the CV set improves with more training data. It would most likely be improved by increasing the size of the training set.

- **Conclusion - Wine Selection Dataset**

As with the previous dataset, a final experiment run on the hold-out test set to estimate their real-world performance and also to determine which learning algorithm performs best on the Wine dataset. **Figure 17** below shows the results.

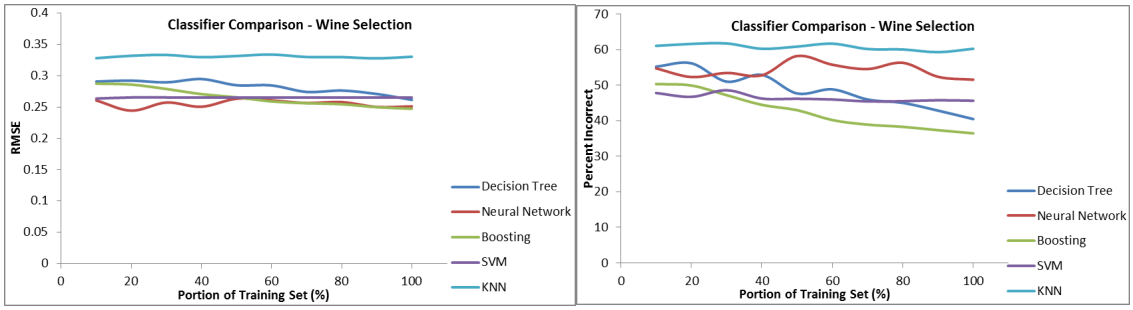

Figure 17: Comparison of Classifier Test Results – Wine Selection

As before, using RMSE as a measure of accuracy and considering the charts, a number of observations jump out. Firstly, it is clear that all the classifiers performed relatively well on this dataset, as all the lines are bunched close together. It can also be concluded that increasing the size of the training set did not have as much impact here as it did on the abalone learners, as most of the lines here are practically horizontal.

With regards to comparative performance, the best performing classifier is Boosting, closely followed by Neural Networks in 2nd. Decision Tree comes 3rd, closely followed by SVM, while KNN brings up the rear.

Table 2 below summarizes the results of the learners on both datasets.

Table 2: Summary of the Performance of Classifiers on both Datasets

Position	Abalone Dataset			Wine Quality Dataset		
	Classifier	RMSE	Classification Error	Classifier	RMSE	Classification Error
1st	Decision Tree	0.1705	74.88%	Boosting	0.2474	36.46%
2nd	Boosting	0.1764	76.87%	Neural Network	0.2511	51.56%
3rd	KNN	0.1774	77.19%	Decision Tree	0.2616	40.48%
4th	Neural Network	0.1774	84.13%	SVM	0.2653	45.65%
5th	SVM	0.1784	75.60%	KNN	0.3305	60.27%