Assignment 1: CS 7641

Gandharv Kashinath

February 2, 2014

Introduction

In this assignment five supervised learning algorithms were analyzed and evaluated for performance on two very different and interesting data sets. The first data set could be used to answer the simple question "Should I buy a particular car?" The decision to buy a car is based on several factors and this data set explores some of the most important ones and helps address the above question. The second data set has an interesting context as it contains the properties of 5 different types of glasses and it has been used in forensic studies to analyze and decide what kind of glasses were found ata crime scene.

This paper first gives a detailed overview of the data then outlines the methodologies and tools used to analyze these five algorithms followed by a presentation of the results and analysis from the experiments and finally concludes with some insights from the analysis.

Data Description

Car Evaluation Data Set (car):

This dataset is a collection of records on specific attributes on cars donated by Marko Bohanec in 1997 and was obtained from the UCI Machine Learning Repository. Using the 6 attributes the cars can be classified on a scale of 'unacceptable' to 'very good.' The following table summarizes the dataset;

Data Set Characteristics:	Multivariate	Number of Instances	1728
Attribute Characteristics:	Categorical	Number of Attributes:	6

Data Transformations:

The car dataset obtained has several nominal attributes that needs to be converted to numeric attributes in order to efficiently apply the various supervised learning algorithms. Presented below is a table summarizing these conversions:

Attribute	Nominal	Numeric Value
Buying	low	1
	med	2
	high	3
	vhigh	4
Maintenance	low	1
	med	2
	high	3
	vhigh	4
Doors	5more (>4)	5
Persons	more (>4)	5

Luggage Boot	small	1
	med	2
	big	3
Safety	low	1
	med	2
	high	3

Data Distribution:

After pre-processing the data, the data distribution can be visualized to gain some insights. From Figure 1 below an obvious attribution of safety and persons can be seen to 'unacceptable' classification of the car. Several other trends like low maintenance attributes to 'very good' classification makes this dataset intriguing and by employing supervised learning algorithms some useful analysis and predictions can be achieved.

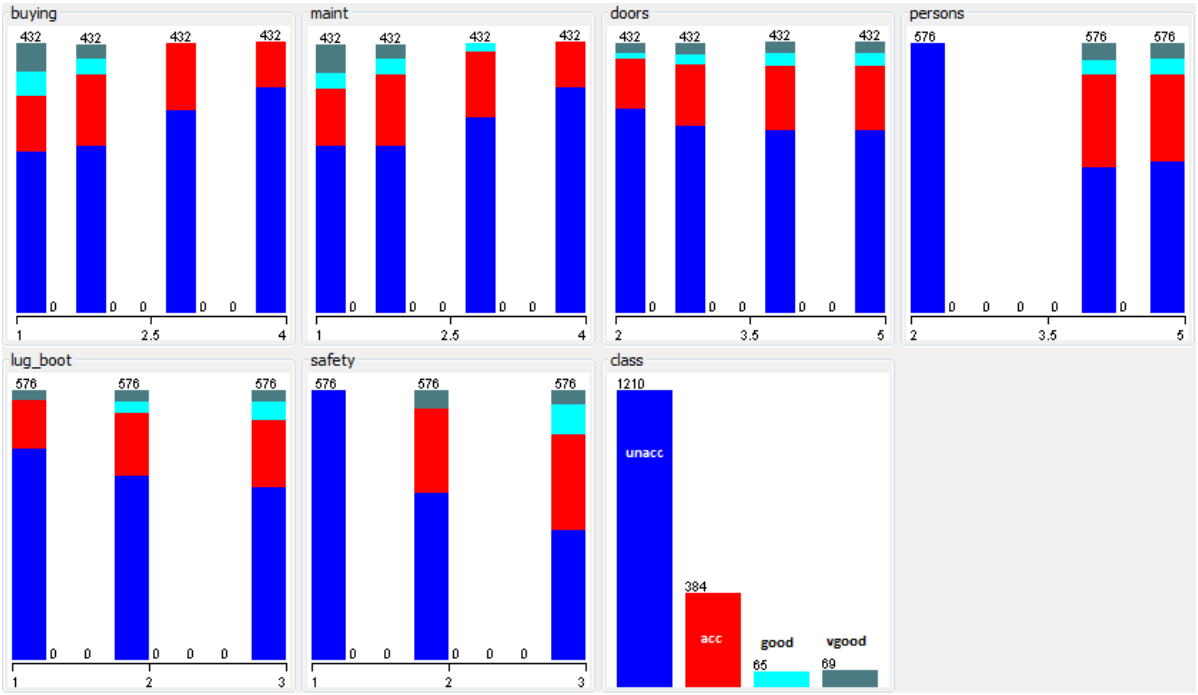

Figure 1: Car Evaluation dataset after pre-processing.

Glass Identification Data Set (glass):

This dataset is from the USA Forensic Science Service which has 6 types of glass classified in terms of their refractive index and oxide content. It was obtained from the UCI Machine Learning Repository. The following table summarizes the dataset;

Data Set Characteristics:	Multivariate	Number of Instances	214
Attribute Characteristics:	Real	Number of Attributes:	10*

Based on these 9 attributes (*1 was removed in data processing which was the ID of the glass) the type of glass can be classified as follows;

Type of Glass	Numeric Value
Building windows float processed	1
Building windows non float processed	2
Vehicle windows float processed	3
Vehicle windows non float processed	4
Containers	5
Tableware	6
Headlamps	7

Data Distribution:

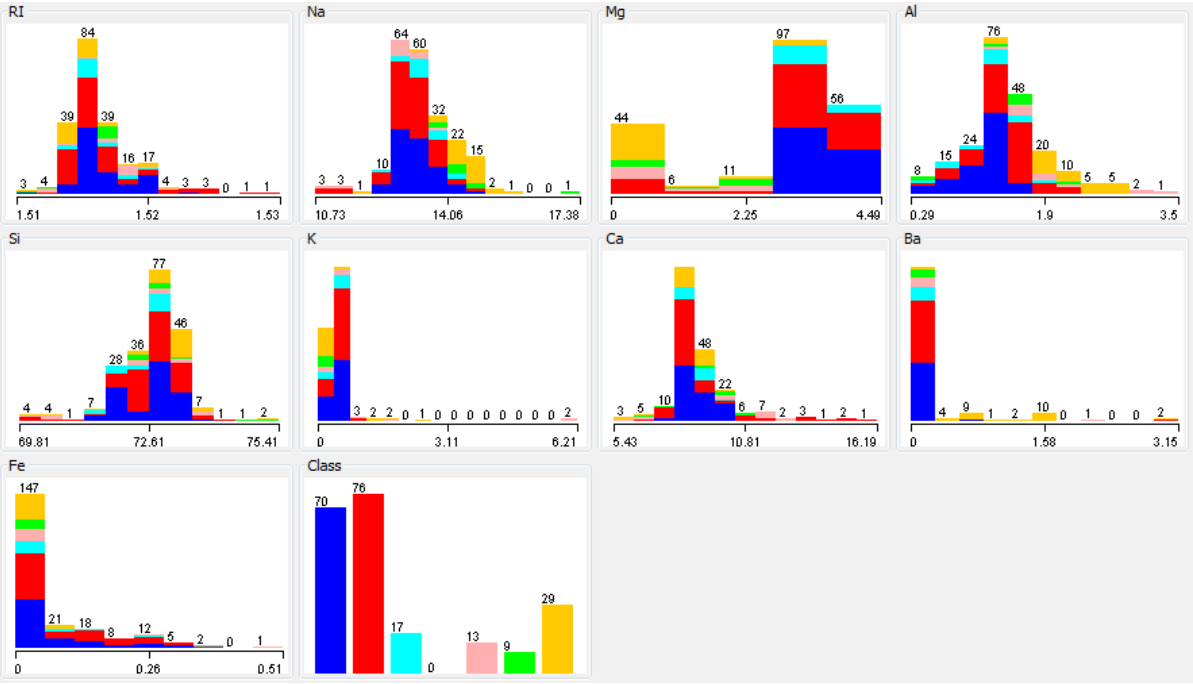

Figure 2: Glass Identification dataset after removing ID attribute.

From the above data distribution the following trends can be observed;

1) Most glasses in this data set have a refractive index between 1.51 and 1.52. So this may not be a critical input in classifying these glasses.
2) Building windows float and non-float processed tends to have 0 Barium content and this could be one of the important factors in the classification problem.
3) Similarly, Magnesium could be important in classifying containers, tableware and especially headlight glass since these have almost 0 Magnesium content.
4) Higher Aluminum content could also be critical in classifying containers, tableware and headlight glass.

5) Sodium content of 14.06 also forms a classification boundary for headlight glasses.
6) Finally, what makes this dataset particularly interesting is there are some obvious classification boundaries between window (both building and vehicle) and other glass attributionsbut within these 'sub-classes', the classification becomes more challenging. It will be interesting to see how the algorithms evaluated in this assignment perform with respect to this challenging dataset. If this data set was modified to have fewer classifications by grouping them in to broader categories (e.g. building windows, vehicle windows and others) then the problem would have been almost trivial due to the well augmented classification boundaries.

NOTE:Feature Scaling- The glass dataset has a huge spread in the magnitude scale for each of the oxide attributes (e.g., Si and K). Feature scaling or data normalization was applied to this data set using the Weka normalization filter to standardize the range from [0, 1]. The data spread was not as significant for the car dataset hence all the tests for this data set were run without feature scaling.

In conclusion, these two data sets pose very distinct challenges; while the car evaluation data set provides simple integer inputs with no clear (linear)distinct classification boundaries for the data, the glass data provides a set of challenging continuous real valued inputs with a mixture of some obvious classification boundaries and some that are not so obvious. The glass data will definitely be the more challenging of the two datasets for the supervised learning algorithms.

Methodology, Algorithms and Tools

In this assignment 5 supervised learning algorithms were studied namely;

1) Decision trees with some form of pruning
2) Neural Networks
3) Boosting
4) Support Vector Machines (SVM)
5) k-nearest neighbors (*k*-NN)

All these algorithms are readily available on the *Weka 3.6* Data Mining software and this tool was exclusively used to perform the experiments and analysis for this assignment. In order to generate plots and pre-process the data MS Excel was employed and all the worksheets are attached for reference. For each of the 5 algorithms, a test centric approach was taken by running cross validation using 10 folds. The purpose of running cross validation is to build a more robust model that can provide reliable results for independent datasets (partially address overfitting) and also, to analyze the performance of each of these algorithms.

Experiments, Results and Discussion

Decision Tree:

Decision tree was the first algorithm employed due to its intuitive representation of the classification problems. By employing the decision tree algorithm, the most important variables in the dataset can be identified by examining the top few nodes of the decision tree. The *Weka* J48 tree, which is an

implementation of the C4.5 algorithm in Java, was used to study the datasets. One of the main drawbacks of the decision tree algorithm is the problem of creating over-complex trees that do not generalize well or in other words, overfit the training set. In order to overcome this problem, the J48 algorithm in Weka has built in to it a pruning algorithm. The “confidenceFactor” controls how much pruning is done and the “minNumObj” controls the minimum number of instances per leaf of the decision tree. In order to study the effects of these parameters, several cases with a combination of these two parameters were run and the results were analyzed.

For the car dataset, the performance of the J48 algorithm was extremely stable and did not show much variation in the in the root mean square (RMS) error with varying confidenceFactor [0.05, 0.25, 0.45] and minNumObj [2, 5]. The plot in Figure 3 suggests that the decision tree algorithm does not suffer from overfitting for this data set and for a distribution of these parameters the RMS error did not show significant variation. In fact,less pruning (higher confidenceFactor) yielded lower error suggesting that this dataset may not be prone to overfitting. The Decision Tree (J48) algorithm classified 97.6% of the instances correctly and took 0.01s to run suggesting that the algorithm worked really well for this data set. The RMS error was 0.1006 with a confidenceFactor of 0.35 and minNumObj of 2. This model can now be used to classify a car based on some simple questions thus simplifying the car buying process.

Visualizing (Figure 4) the decision tree reinstated the initial theory that persons and safety lie at the top of the decision tree and are one of the first decisions to be made before buying a car. ‘Buying’ followed by safety was also critical in deciding whether a car was classified as unacceptable or not.

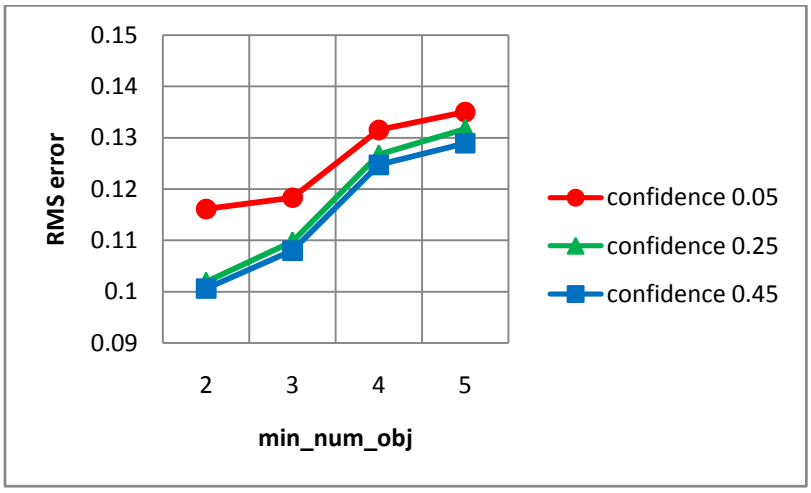

Figure 3: Overfitting analysis with J48 algorithm for cardataset.

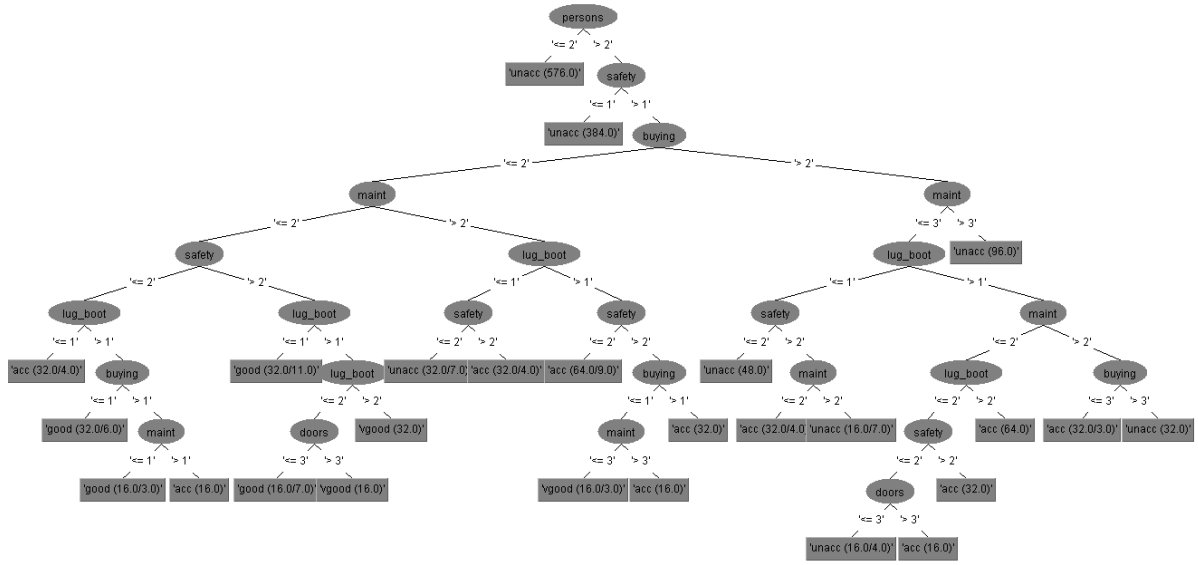

Figure 4: Decision tree from the J48 algorithm for car dataset.

For the glass dataset, the decision tree, J48 algorithm performed poorly. With the default pruning parameters confidenceFactor = 0.25 and minNumObj =2, the decision tree algorithm was able to classify 66% of the instances correctly with an RMS error of 0.29. No significant difference in the RMS error was observed by varying the pruning parameters confidenceFactor [0.05, 0.25 and 0.45] and minNumObj [2, 5]. The best performance of 69% correctly classified instances was obtained with confidenceFactor=0.25 and minNumObj=3.

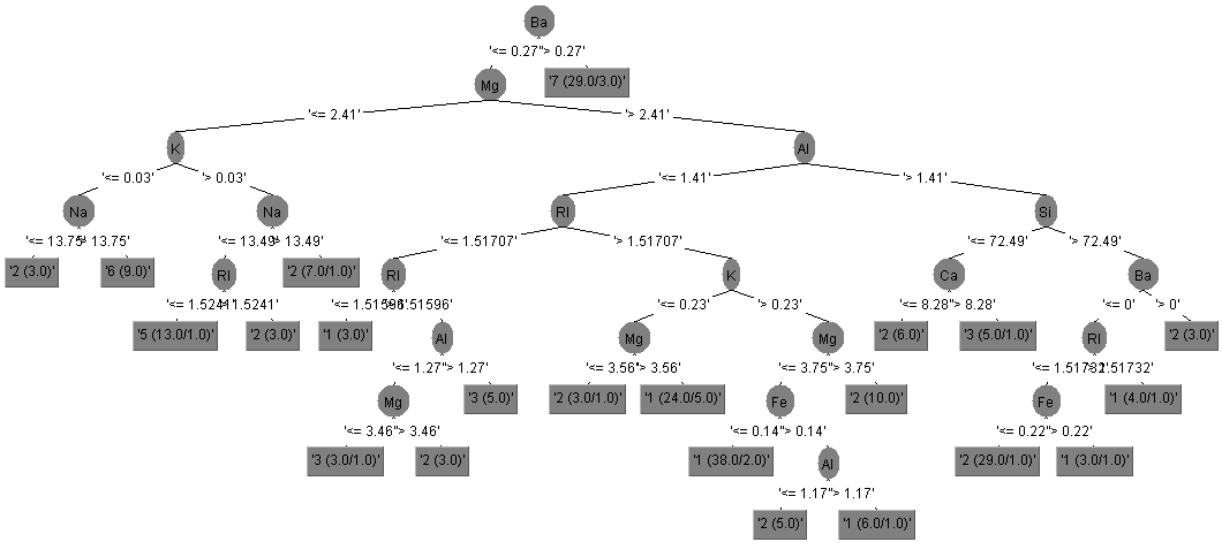

Figure 5: Decision tree from the J48 algorithm for Glass Identification data.

Neural Networks:

In order to analyze the performance of a Neural Network algorithm on the two datasets used in this study, the *Weka*, Multilayer Perceptron classifier was used. This classifier uses backpropagation to

classify instances and the nodes in this network are all sigmoids. The Multilayer Perceptron algorithm was significantly slower than the other algorithms. This could be due to the complexity of the neural network and the time spent in the "backpropagation" step during the execution of the algorithm. In order to analyze the performance of this algorithm on the two datasets, the effect of varying the learning rate and momentum was studied. Learning rate is an important parameter for this algorithm as it is multiplied by the gradient determined by the "backpropagation" algorithm which is used to update the new values of the weights. The momentum parameter helps avoid local minima as much as possible. Hence, the effect of these two parameters can be critical in developing a stable and robust model. In this analysis a parameter sweep of learning rate [0.1, 0.3 and 0.5] and momentum [0.2, 0.3, 0.5 and 0.7] was conducted for both datasets. Along with the 10 fold cross validation, a 70% training data and 30% test data split was also compared and evaluated.

For the car dataset, the performance of the Multilayer Perceptron algorithm was good (but not as good as J48) and was able to classify 93% of the instances correctly with a RMS error of 0.1436 on the 10 fold cross-validation run. From the plot below, the lowest RMS error was reported by the learning rate 0.1 and momentum 0.2 (Figure 6).

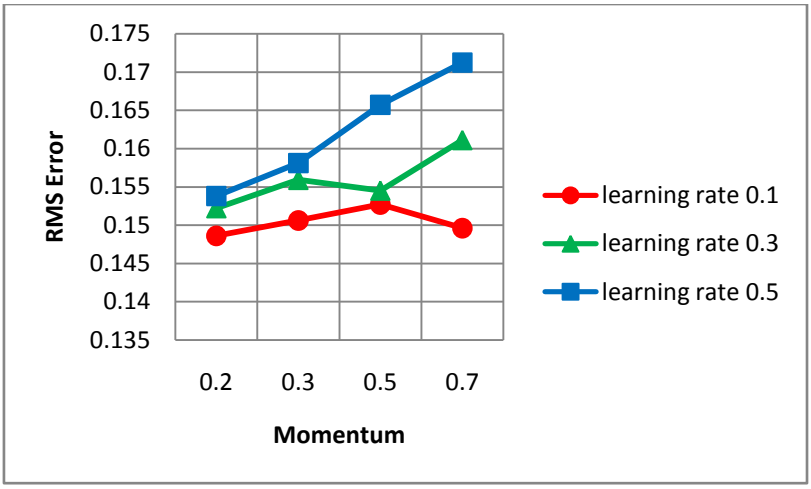

Figure 6: Multilayer Perceptron parameter sweep of learning rate and momentum for the car dataset.

In order to study the effect of varying the training set and testing set, the 70% training and 30% testing set data split was tried and only improved the performance 1% thus yielding a correct instance classification of 94%. The RMS error reduced by 3% suggesting no significant gain in performance can be made by using this model. The 10 fold cross validation model is more robust and sufficient for this dataset and can withstand overfitting better than the split training and test set.

For the glass dataset, the performance of the Multilayer Perceptron was bad and was only able to classify 72% of the instances correctly with the RMS error being 0.2469. Varying the learning rate and the momentum did not improve the results as shown below in Figure 7. Testing with the 70% training and 30% testing set for cross validation degraded the performance with only 58% of the instances being classified correctly with an RMS error of 0.30.

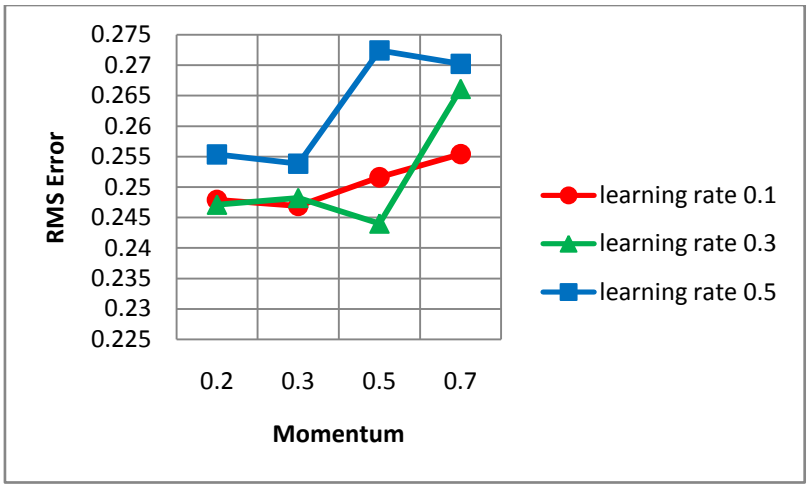

Figure 7: Multilayer Perceptron parameter sweep of learning rate and momentum for the glass dataset.

The default settings for the number of hidden layers in *Weka* for the Multilayer perceptron is the mean value of the input and output layers. Varying these hidden layers between 2, 15 and default for both data sets did not improve the results significantly. By increasing the number of hidden layers the complexity of the model also increased and the simulations ran longer (e.g. glass data set took 2x longer to run the 15 hidden layers compared to the default). Overall, the performance of the Neural Network algorithm was not as good as the decision tree algorithm for both the datasets. While the car dataset results were acceptable the glassdataset results were poor.

Boosting:

The goal of boosting is to improve the accuracy of any given learning algorithm and in this assignment the AdaBoostM1 algorithm in *Weka* was applied to the J48 decision tree classifier. In AdaBoost each training pattern receives a weight that determines its probability of being selected for a training set for an individual component classifier. If a training pattern is accurately classified; then its chance of being used again in a subsequent component is reduced. Conversely, if the pattern is not accurately classified, then its chances of being used again are raised. In this way, AdaBoost hones in on the difficult patterns. In *Weka*, AdaBoostM1 allows the user to control the number of iterations for the boosting algorithm. Although, being aggressive with the pruning will not hurt the boosting algorithm, it was observed that the aggressively pruned simulations did not give starkly different results. Hence, the best performing decision tree settings were used before applying boosting.

For the car dataset, the AdaBoostM1 algorithm improved the performance dramatically. By applying boosting, 99% of the instances were classified correctly with an RMS error of 0.0636. This was an improvement in performance of 37% compared to a standalone J48 decision tree algorithm. By applying the boosting algorithm the timing to run the classification model increased from 0.01s to 0.09s which is not very significant for this data set but could be significant for bigger data sets that take longer.

For the glass dataset, the AdaBoostM1 algorithm improved the performance of the standalone J48 algorithm by 15%. The RMS error reduced from 0.2788 (standalone J48) to 0.2372 with boosting. This

algorithm was now able to successfully classify 78% of the instances correctly as opposed to the 69% from the standalone J48 decision tree algorithm. The standalone performance was very poor to begin with but improved with boosting. AdaBoostM1 in *Weka* allows the user to set the number of iterations to run the boosting. Changing this value from the default 10 to 1000 further improved the performance by 2% and was able to classify 80% of the instances correctly. This may not be worth pursuing since it increases the run time by 8x and the performance gains are not substantial.

From the above two results, it can be concluded that for datasets that are not too large boosting algorithms like AdaBoost definitely help improve the performance of classifiers. For large datasets the boosting algorithm may require higher computation time but if it improves performance as dramatically as the above two cases then it must be used.

Support Vector Machines (SVM):

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. In order to study this algorithm the two datasets used in this assignment were classified using the *LibSVM* library available in *Weka*. *LibSVM* allows the users to study the effects of using different kernels while implementing the SVM algorithm. As part of this assignment the performance of the SVM algorithm was studied using 4 different kernel functions namely; linear, polynomial, radial and sigmoid.

For the car dataset, the SVM algorithm performed well and was able to classify 96% of the instances correctly with an RMS error of 0.1339. This performance was achieved using the radial kernel function. The plot below (Figure 8) shows the RMS error for various kernel functions. The run time for this algorithm was 0.14 s which was higher than some of the simpler algorithms like; J48 decision tree and *k*-NN.

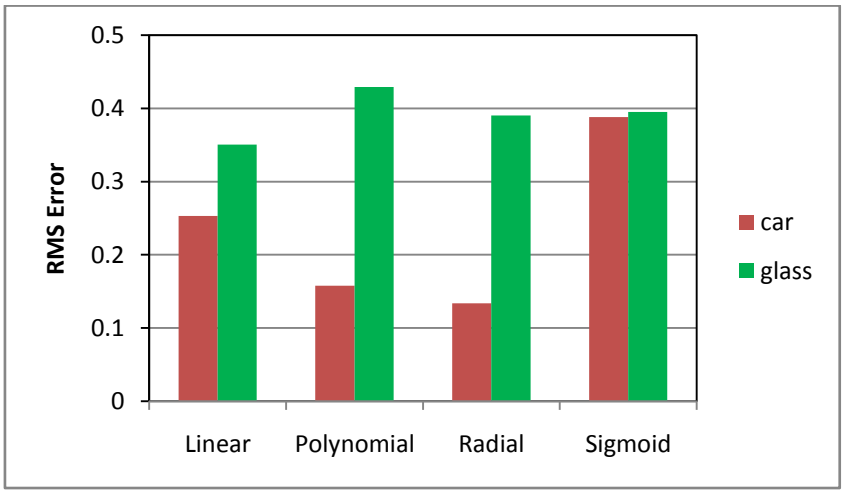

Figure 8: LibSVM algorithm performance for various kernel functions for the car and glass dataset.

For the glass dataset, the *LibSVM* algorithm performed extremely poorly. It was able to classify 57% of the instances correctly with an RMS error of 0.3505. This was achieved by the linear kernel. The run time for this algorithm was 0.06s.

k-Nearest Neighbor (k-NN):

In *k-NN classification*, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k*nearest neighbors. If*k*=1, then the object is simply assigned to the class of that single nearest neighbor. *k*-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. In order to analyze this algorithm, the *Weka* IBk algorithm was applied to the two datasets that were studied in this assignment. The parameters that control this algorithm are; the value of *k* and the nearest neighbor search distance function. In this assignment a parameter sweep of *k*[1, 3, 5, …, 15] and the Euclidean and Manhattan distance for each *k* was performed and the results were analyzed.

For the car dataset, the *k*-NN algorithm worked well and it was able to classify 98% of the instances correctly with an RMS error of 0.134. The two distance algorithms namely, the Manhattan and the Euclidean performed similarly (Figure 8) with Manhattan algorithm having the highest correctly classified instances and the Euclidean having the lowest RMS error.

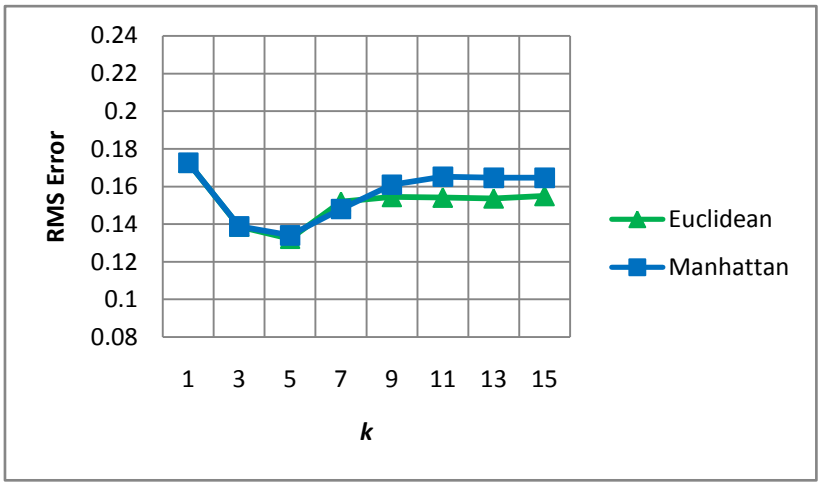

Figure 10: *k*-NN algorithm comparison with varying *k* and distance algorithm for car dataset.

From the above plot (Figure10) it can be concluded that the best choice of *k* for this dataset is 5 as it has the lowest RMS error and was able to classify the highest number of correct instances. The run time for this was extremely small and was the fastest among all the other algorithms.

For the glass dataset, the *k*-NN algorithm did as well as the Neural Network algorithm but not as well as the AdaBoostM1. It was able to classify 72% of the instances correctly with an RMS error of 0.234. From the plot below (Figure 11), it can be concluded that the Manhattan distance algorithm worked slightly

better than the Euclidean distance algorithm. The best choice of *k* for this dataset is 3 since this had the lowest RMS error and the highest classification percentage. Again, the time taken by this algorithm was extremely low and was fastest compared to the other algorithms.

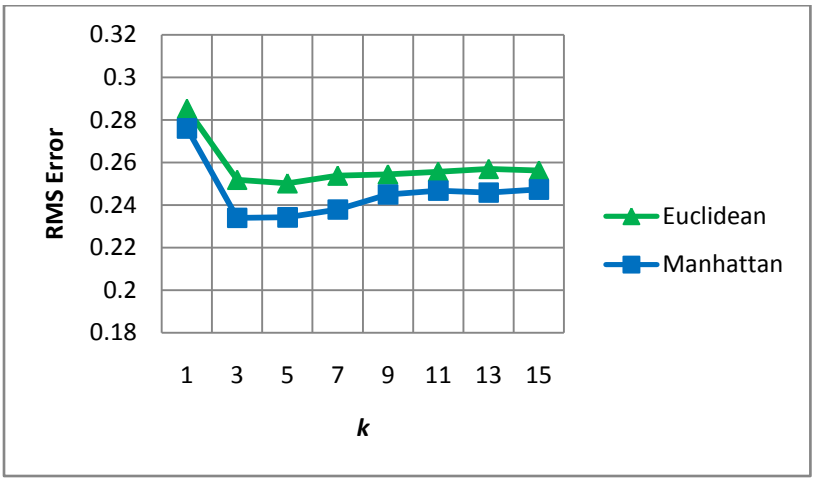

Figure 11: k-NN algorithm comparison with varying k and distance algorithm for glass dataset.

Conclusion

Presented below is a summary of the performance of the 5 algorithms studied in this assignment applied to the two chosen datasets.

Data set	Algorithm	Lowest RMS Error	% Correctly Classified Instances	Time (s)
Car Evaluation	Decision Tree	0.1006	97.6	0.01
	Neural Network	0.1486	93.5	1.39
	Boosting	0.0636	99.1	0.09
	SVM	0.1339	96.4	0.14
	k-NN	0.1340	98.4	0.0001
Glass Identification	Decision Tree	0.2788	69.2	0.01
	Neural Network	0.2469	71.5	0.38
	Boosting	0.2372	78.0	0.05
	SVM	0.3505	57.0	0.06
	k-NN	0.2340	72.0	0.0001

From the above table and the discussion and analysis of previous sections, the following conclusions can be drawn;

1. Overall the car evaluation dataset worked better with all algorithms. All of the five algorithms were able to classify more than 90% of the instances correctly. On the other hand, the glass identification dataset was more challenging and the best performance was only at 78% of correctly classified instances.

2. The best performing algorithm for the both the datasets was the J48 decision tree with boosting. This algorithm has the highest percentage of correctly classified instances and the lowest RMS error.
3. From the decision tree visualization for both data sets, although the initial candidates of important features were validated, some other features which were not apparently important were revealed. For example, the car evaluation dataset revealed "buying" to be an important attribute which was not very obvious from the data distribution. In the glass identification dataset, K (potassium) was among the top few nodes of the decision tree.
4. 10-fold cross validation used for these algorithms is a good validation approach to avoid overfitting and to generalize the model to new testing samples. A simple comparison with 70% training and 30% testing data was carried out for the car evaluation data and it was observed that the performance of this setting only improved the classification performance by 1%. This suggested that the 10-fold cross validation was a good approach to test and validate the model due to the random picking of the training, validation and testing data.
5. The SVM algorithm performed extremely poorly on the glass identification data set. For the SVM algorithm, *Weka* has access to 4 kernel functions and considering the linear kernel did the best job among all the kernels, it could be concluded that some variation of the linear kernel may be explored to get better performance. Also, other kernel functions besides the 4 available could be investigated. The overall poor performance of the SVM algorithm suggests that the glass dataset is a high-dimensional problem and better kernels could provide a better solution.
6. The Neural Network algorithm, although complicated, did not yield the best performance. This algorithm took the longest time to build the model compared to all the others and this could be attributed to the time taken by the linear solver to invert a matrix. The complexity of the hidden layers could have caused the model to create a complex model for both the data sets.
7. The *k*-NN algorithm was the fastest to run on both data sets. This was the second best algorithm after the decision tree with boosting. An optimum *k*of 5 for the car evaluation data set and 3 for the glass identification data set could be chosen. The Manhattan distance function performed slightly better than Euclidean distance for both data sets.
8. The Decision Tree algorithm was a good starting point for both the data sets as it gave a visual representation of the attributes and classification. The performance of this algorithm was good for the car evaluation data set while being poor for the glass identification data. Overfitting was addressed with both the 10-fold cross-validation approach and the picking of some parameters that control overfitting.

The five algorithms studied in this assignment are interesting examples of supervised learning techniques which were applied to two real-world datasets. Some algorithms worked better than others for a given data. Choosing an algorithm that works for a given dataset requires some experimentation and also knowledge about the data can be critical in making this choice.