Supervised Learning Analysis

Overview

With the explosion of Big Data over the last few years, Analytics has played an increasing role in different industries such as business and medicine. As Analytics becomes progressively more important and interwoven into our everyday lives, it is important for us to understand how different analytical techniques behave under a variety of circumstances to ensure we properly apply insights gleamed from the data. In this paper, I will examine different supervised learning techniques, and explore how they interact with different datasets. Specifically, I will review 1) K- Nearest Neighbors (KNN) 2) Decision Trees 3) Support Vector Machines (SVM) 4) Boosting 5) Artificial Neural Networks (ANN) and their application to the HR Analytics Dataset and the Wisconsin Breast Cancer Dataset found on Kaggle and UCI Machine Learning Repository respectively.

HR Analytics Dataset

The HR Analytics Data used for analysis is a random subset of the HR Analytics Data from Kaggle. The data includes nine attributes that are used to identify and describe company employees. Attributes include information such as salary level and tenure at the company. The data will be used to predict (or classify) whether an employee leaves. To measure the performance of each algorithm on the dataset, the kappa metric will be used. With the data imbalanced at a 2:8 ratio, an algorithm could get an 80% accuracy and still perform no better than the baseline classifier. The kappa metric measures the observed accuracy (algorithm accuracy) and compares it to the expected accuracy (baseline accuracy). This metric is better suited for imbalanced data.

Below are some reasons why I found the data to be interesting:

- **Mixed Feature Types:** The data contains numerical, ordinal, and nominal data. Some algorithms find it difficult to process categorical data, and therefore the data must be pre-processed before analysis.
- **Slightly Imbalanced Data:** The ratio of employees that left to those that stayed is approximately 2:8. A baseline classifier that categorizes each point to the majority has an accuracy of 80%. Therefore, the kappa metric will be used to determine “best”.
- **Relatively Low Feature to Sample Ratio:** The ratio of features to samples is approximately 1:830. The lower the ratio, the less likely the “curse of dimensionality” becomes an issue.
- **Application:** Coming from a consulting background, I know first-hand that employee retention rate is an issue for some companies. If analytics can help companies identify employees at risk for leaving, then they can do a better job of retaining top talent.

Wisconsin Breast Cancer Dataset

The Wisconsin Breast Cancer Data is from the UCI Data Repository. The data includes 30 attributes that are used to describe patient tumor cells. The attributes include different measurements, such as size and density, for each tumor. The data will be used to predict (or classify) each tumor as malignant or benign. To measure the performance of each algorithm on the dataset, the accuracy metric will be used.

Below are some reasons why I found the data to be interesting:

- **Balanced Dataset:** This dataset is more balanced than the HR Analytics Data. It would be interesting to see how different algorithms handle each dataset based on how balanced they are.
- **Additional Noise:** Additional noise was introduced into the dataset to see how the different algorithms handle the noise. The noise constitutes for roughly 10% of the data. The noise will manifest itself as irreducible error.
- **Application:** I have seen loved ones battle cancer and witnessed its effects on family and friends. I thought it would be interesting to see how analytics can be used in the fight against cancer.

Data Preprocessing

Before any analysis took place on the data, there were a list of standard preprocessing activities that were performed and decisions that were made to ensure that the data was analyzed properly. Below you can see a list of those activities and decisions:

- **Recursive Feature Elimination:** To perform feature selection, recursive feature elimination (RFE) was performed on both datasets. Feature selection helps to avoid the “curse of dimensionality” and helps to reduce the risk of overfitting the data. RFE found all features of the HR Analytics Dataset to be important for prediction. On the other hand, it found only 8 of 30 features of the Wisconsin Breast Cancer Dataset to be important for prediction. Only those features deemed as important will be used in analysis.
- **One Hot Encoding:** Some algorithms have a difficult time handling categorical data. One hot encoding transforms categorical data to a format that is easier to process. Below are the algorithms that required transformations of categorical variables and the reason why:
    - **KNN** – The KNN algorithm measures “distance” to determine similarity between instances. Categorical data are not numerical in nature, and therefore are difficult to use to calculate “distance”.
    - **SVM** – The SVM algorithm constructs a hyperplane through the feature space to classify the data. SVMs have a hard time mapping categorical data to the feature space.
    - **ANN** – The ANN algorithm uses an activation signal multiplied by some weight as input to each node. It is difficult to directly map a categorical variable with multiple levels to an activation signal.
- **Normalizing/Scaling**: Normalizing or scaling data is necessary for some algorithms to avoid any unwanted affects during data processing. Below are a list of algorithms that require normalization or scaling of the data and the reason why.
    - **KNN** – The KNN algorithm measures “distance” to determine similarity between instances, and features with different ranges can have an adverse effect on the distance calculation.
    - **SVM** – The SVM algorithm constructs a hyperplane through the feature space, and to ensure it is properly constructed the different features have to be proportional to one another.
    - **ANN** – The ANN algorithm has a single learning rate that affects all features the same. Therefore, each feature must be scaled to ensure the learning rate affects each feature the same
- **Cross Validation:** A 10-fold cross validation was the preferred method of training and validation for all models. This method tends to provide a better performance approximation than other traditional methods.

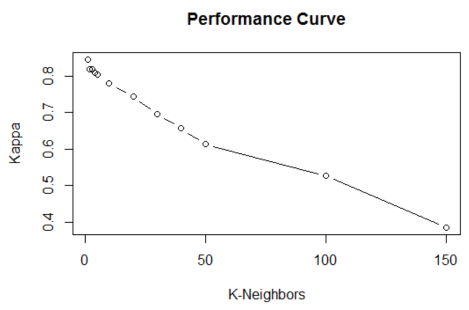
K-Nearest Neighbors (KNN)

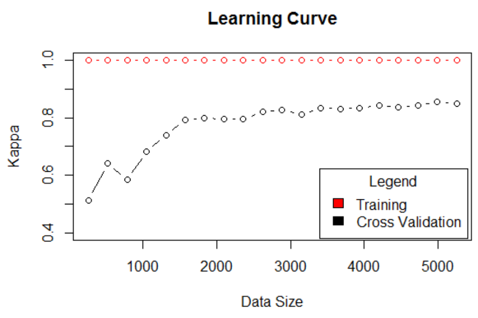
KNN determines the nearest (or similar) neighbors by measuring the distance between instances within the feature space. To measure similarities between instances, the Euclidean distance was applied to both datasets. New to the domains, I have no domain knowledge to pick one measure of distance over another and the Euclidean distance is often seen as the baseline or standard of measure. Therefore, Euclidean distance is an excellent starting point for analysis.

HR Analytics Data: Using the HR Analytics Data, the KNN algorithm was tuned over several k's, and performed best at k = 1. As the value of k increases, the algorithm progressively performed worse. Intuitively this makes sense, given the data. With the data being slightly imbalanced, the algorithm performs better at lower values of k. The chances of predicting the minority class correctly is higher at lower values of k, because there is less opportunity to include the majority class as a nearest neighbor. As the value of k increases, the algorithm is more likely to include more of the majority class and misclassify the minority class, due to the imbalance. Reference the "Performance Curve" chart below.

KNN's learning curves show that the cross validation performance is improving with increases in the dataset size. Because the learning curve has not plateaued, you can feed the algorithm more data and potentially improve its performance. It should be noted that the cross validation learning curve is not steep and it may take large amounts of data to get minor increases in performance. The training curve is consistently scoring a kappa of 100%. At k =1, each sample in the training set is predicting itself – thus the 100% kappa. The gap between the training curve and cross validation curve is due to variance. As mentioned earlier, training on more data will help to minimize those errors but it may come at a cost of a lot of data. Reference the "Learning Curve" chart below.

Given the current dataset and optimal tuning parameters (k = 1), the KNN algorithm scored a kappa of 84.62% during testing. To improve on the performance of the algorithm, the following approaches can be explored 1) Apply different kernels to map the feature space to a higher dimension 2) Apply different distance functions to redefine "similarity" between instances 3) Train more data to eliminate variance

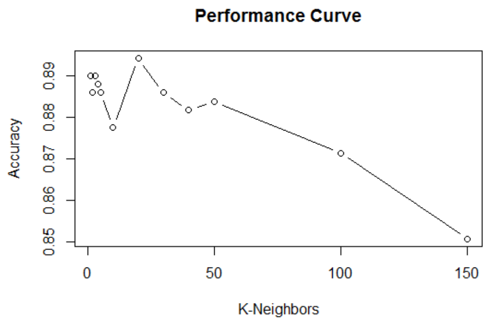


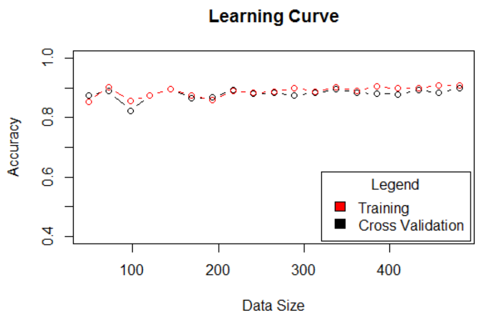


Wisconsin Breast Cancer Data: Using the Wisconsin Breast Cancer Data, the KNN algorithm was tuned over several k's and performed best at k = 20. As the value of k increases toward 20, the performance of the model improves, suggesting that the model underfits the data at k < 20. As the value of k increases beyond 20, the performance of the model gets worse, suggesting that the model overfits at k > 20. Reference the "Performance Curve" chart below.

KNN's learning curves converge early, but do not stabilize until 200 samples. The convergence of the two curves suggest that there is limited variance in the model. Also the convergence at a relatively high accuracy level shows that there is some but little bias. That error may also be due to the noise that was introduced into the dataset. Reference the "Learning Curve" chart below.

Given the current dataset and optimal tuning parameters (k = 20), the KNN algorithm scored an accuracy of 90.73% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Apply different kernels to map the feature space to higher dimensions 2) Apply different distance functions to redefine "similarity" between instances 3) Add additional features to reduce bias





Decision Trees

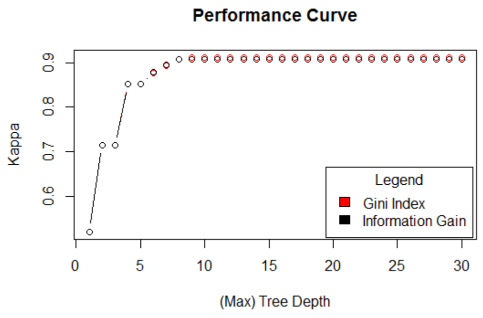
Decision Trees tend to overfit data if they become too large. To prevent this from happening, one can implement pre-pruning or post-pruning. Post-pruning was the method chosen to prune the decision trees built from the HR Analytics Dataset and the Wisconsin Breast Cancer Dataset. Unlike pre-pruning, post-pruning can be implemented with limited domain knowledge and can be automated to find the best tree. Post pruning was performed during the search over the hyper-parameter (max) tree depth. The tree is built to the specified max tree length, and pruned back to find an optimal tree.

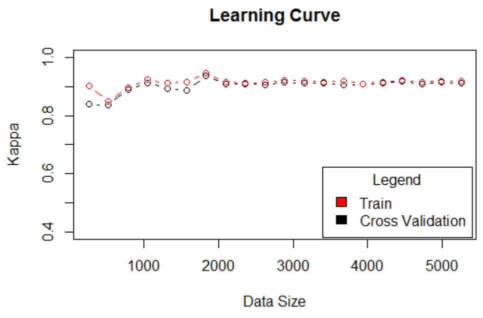
HR Analytics Data: Using the HR Analytics Data, the decision tree model was tuned over the hyper-parameter (max) tree depth and the impurity metrics information gain and gini index. The difference in performance due to the impurity metrics were marginal (0.46%), and the performance curves associated with each metric were very similar. The results are consistent with findings stating that these metrics are interchangeable for most modeling problems. For the rest of the analysis, information gain will be used. Reference the "Performance Curve" chart below.

Unlike the impurity metrics, the tree's depth had a larger impact on performance. Tree depth is closely related to how well a decision tree fits the data, and therefore will have a larger impact on performance. The decision tree performance is optimal at a tree depth of nine. The "Performance Curve" chart shows that the performance plateaus after tree depth reaches a value of nine. When a max tree depth is reached that is greater than nine, the trees are pruned back to the optimal tree depth. Reference the "Performance Curve" chart below.

The learning curves show as the size of the dataset increases the training error and cross validation error converge to a high kappa value (low error rate). The convergence of both errors curves indicates that there is minimal error due to variance, and the high kappa value indicates that there is low systematic error due to bias. It also should be noted that the learning curves converged early, suggesting that a smaller dataset would produce similar results. Reference the "Learning Curve" chart below.

Given the current dataset and optimal tuning parameters (tree depth = 9), the decision tree algorithm scored a kappa of 91.75% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Add additional features to remove bias.



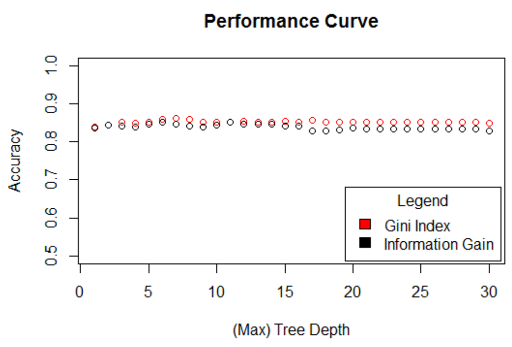


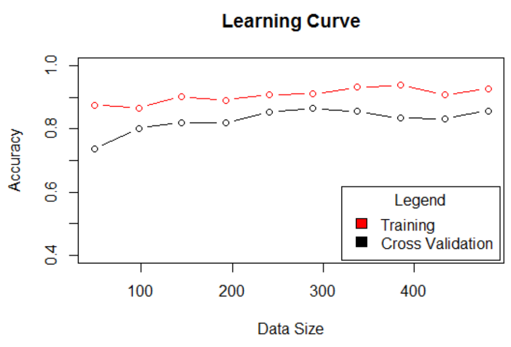
Wisconsin Breast Cancer Data: Using the Wisconsin Breast Cancer Data, the decision tree model was tuned over the hyper-parameter (max) tree depth and the impurity metrics information gain and gini index. Consistent with the HR Analytics Data analysis, neither impurity metric had an advantage over the other, having a marginal difference of only 0.85%. The "Performance Curve" chart reconfirms the fact that each metric is interchangeable. Going forward, information gain will be used in the analysis. Reference the "Performance Curve" chart below.

The "Performance Curve" chart shows that the optimal max depth value is 6, but for the most part the performance curves are flat. This suggests that the patterns in the data are relatively simple, and can be captured with lower tree depths or less partitions in the data. The difference in test performance between the optimal max depth (max depth = 6) and a max depth of one is 5.85%. If there is a preference for a simpler model, one could reduce the tree depth at a minor cost to performance. Reference the "Performance Curve" chart below.

The decision tree's "Learning Curve" chart shows that the training and cross validation learning curves have plateaued. This suggests that more data will not improve model performance. Also, the relatively high accuracy with smaller datasets and relatively consistent accuracy across the different data sizes, reconfirms the notion that the patterns in the data are relatively simple and are represented well throughout the data. With small amounts of data, the model was able to pick-up on the patterns of the "true data" (non-noisy data), receiving a high level of accuracy early. Reference the "Learning Curve" chart below.

Given the current dataset and optimal tuning parameters (tree depth = 6), the decision tree algorithm scored an accuracy of 88.29% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Remove features to reduce variance in the model.





Support Vector Machines (SVM)

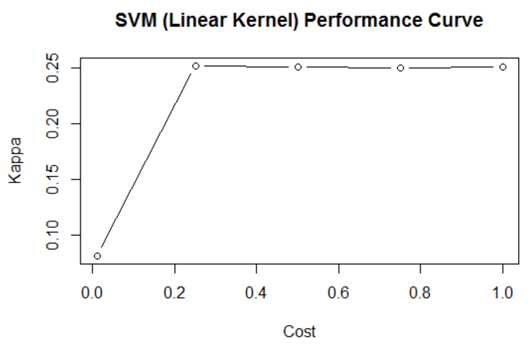
HR Analytics Data: Using the HR Analytics Data, the SVM algorithm was trained over a range of cost values and different kernels – linear kernel and radial kernel. For various cost values, the linear kernel performed poorly. Its poor performance, suggests that the data is not linearly separable. Reference the "SVM (Linear Kernel) Performance Curve" chart below.

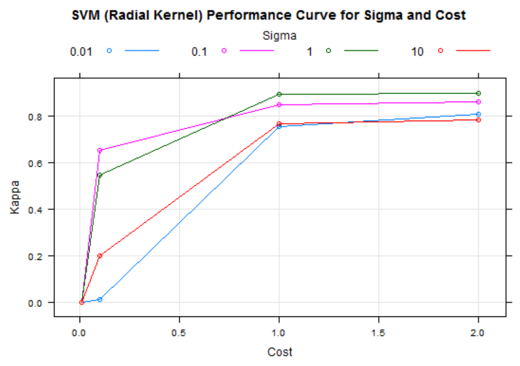
The radial kernel performed substantially better than the linear kernel. The radial kernel has an additional parameter, sigma, that was also tuned. The radial kernel's performance improves with increases in the cost parameter, and then plateaus when it reaches its optimal value of 1. Reference the "SVM (Radial Kernel) Performance Curve for Sigma and Cost" chart below. Using a cost value of 1, various values of sigma were also tested. The optimal value of sigma is also 1. When the value of sigma is increasing towards 1, the performance of the algorithm improves - suggesting that the model is underfitting. For values of sigma exceeding 1, the performance of the algorithm gets worse - suggesting that the model is overfitting. Reference the "SVM (Radial Kernel) Performance Curve for Sigma" chart below.

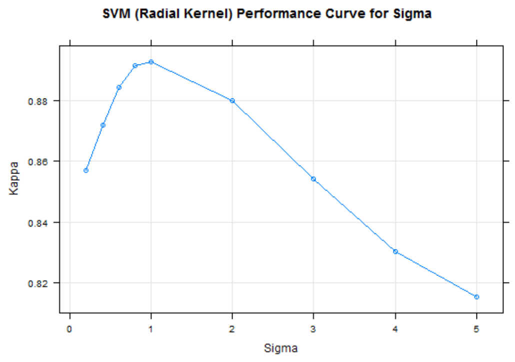
SVM's learning curves for the radial kernel are coming together but have not yet converged. At the current trajectory, it seems that both curves will converge at a high kappa level. Training on more data will reduce variance, causing the curves to converge. As expected, the training curve has a low error rate (high accuracy). The model is being tested on the same instances that were used to create the radial boundaries. Also, the fact that a radial kernel is performing well with a moderate sized sigma (sigma = 1), indicates that the minority instances are locally grouped together throughout the feature space as opposed to being evenly dispersed throughout the feature space.

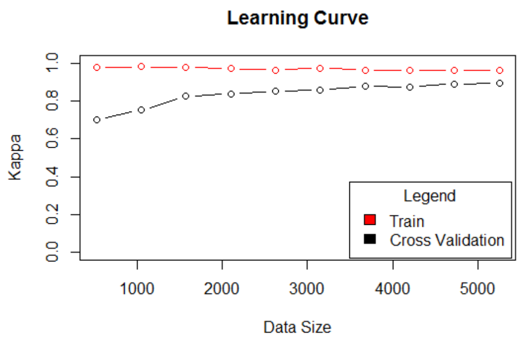
Given the current dataset and optimal tuning parameters (kernel = radial, cost = 1 , sigma = 1), the svm algorithm scored a kappa of 90.16% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Apply different kernels to map the space to a higher dimension 2) Train more data to reduce variance

.







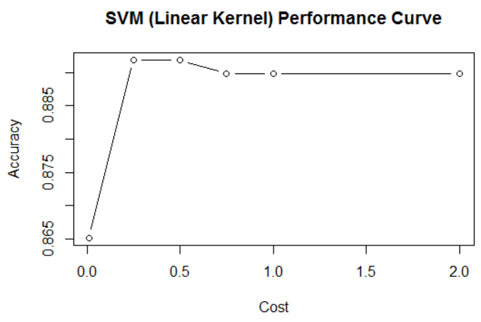


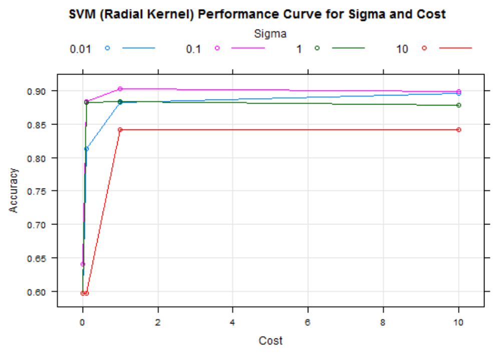
Wisconsin Breast Cancer Data: Using the Wisconsin Breast Cancer Data, the SVM algorithm was trained over a range of cost values and different kernels – linear kernel and radial kernel**.** The linear kernel performed well with an accuracy of 88.29% at a cost value of 0.25 during testing. The high accuracy suggests that the data is close to being linearly separable. Reference the “SVM (Linear Kernel) Performance Curve” below.

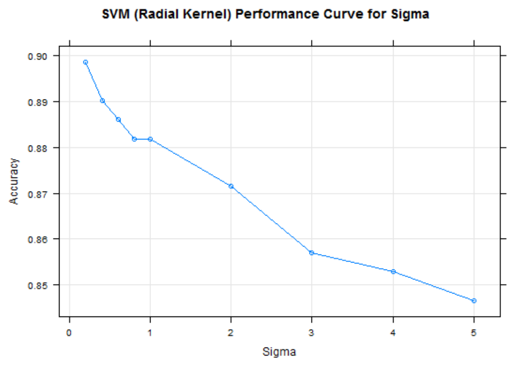
The radial kernel’s performance was slightly better than the linear kernel at an accuracy of 91.22% during testing. The performance curves show that the kernel is at peak performance when the cost is 0.01. Reference the “SVM (Radial Kernel) Performance Curve for Sigma and Cost” chart below. At optimal cost, the radial kernel performs best when sigma is small (sigma = 0.2). Reference the “SVM (Radial Kernel) Performance Curve for Sigma” chart below.

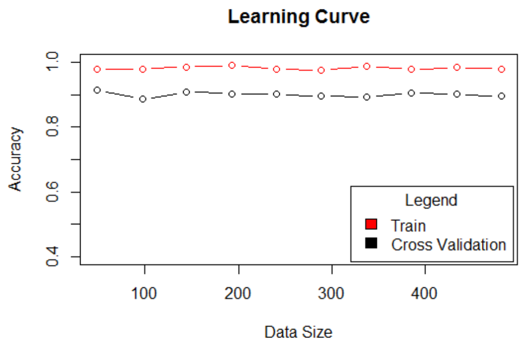
The learning curve tells an interesting story. The training curve consistently scores a high accuracy level because it overfits the data. The radial kernel creates boundaries around the same data that it is testing on – thus the high training curve accuracy rate. The gap in accuracy between the training and cross validation learning curves, is mainly due to the added noise. Despite the noise, the algorithm is still able to find the pattern in the “true” dataset. The curves are also consistent across the different data sizes. This reconfirms the simplicity of the patterns in the “true” dataset, and evidence that the dataset may be close to being linearly separable – a linearly separable dataset can be thought of as being a simple pattern.

Given the current dataset and optimal tuning parameters (kernel = radial, cost = 0.01 , sigma = 0.2), the svm algorithm scored a accuracy of 91.22% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Apply different kernels to map the data to higher dimensions.









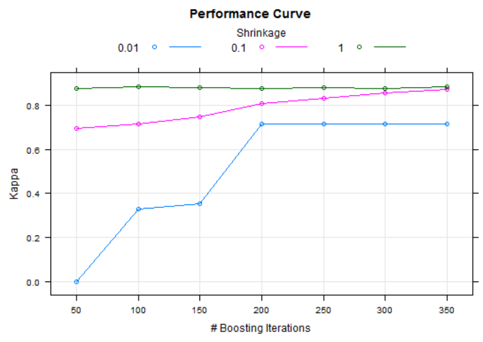
Boosting (Decision Trees)

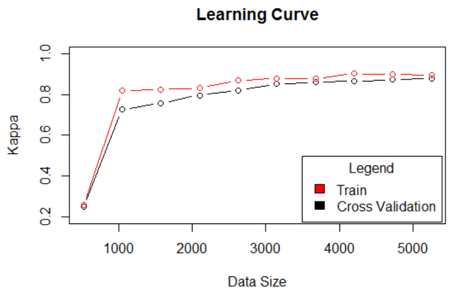
HR Analytics Data: Using the HR Analytics Data, The boosting algorithm was trained over several iterations and learning rates (shrinkage). From the performance chart it is evident that the algorithm's performance improves with more iterations and higher learning rates. With more iterations, the algorithm has the opportunity to develop more weak learners over the data which strengthens the classification capabilities of the method; and with higher learning rates, the algorithm places more weight on misclassifications each iteration which forces the algorithm to learn faster. Reference the "Performance Curve" chart below.

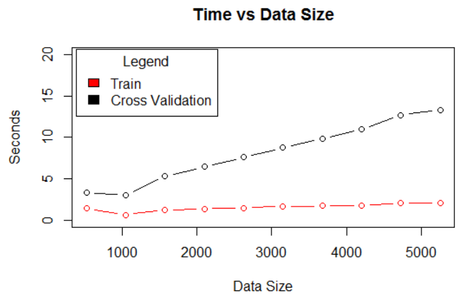
As the algorithm processes more data, the learning curves eventually converge. This indicates that with enough data, the model can minimize any errors due to variance. This is no surprise given how the algorithm works. The boosting algorithm is an ensemble method that makes predictions over several weighted weak classifiers. By averaging its predictions over several classifiers, the boosting algorithm can effectively reduce variance in its estimates. Reference the "Learning Curve" chart below.

With more data the boosting algorithm can reduce error introduced by variance, but it comes at a cost. At large amounts of data, the time it takes to train with cross validation grows at a much faster rate, than it does training without cross validation. Reference the "Time vs Data Size" chart below. Intuitively, that makes sense. Cross validation must loop through the data 10 times (k-folds = 10), whereas training the model without cross validation loops through the data once. In environments or scenarios that are time sensitive, a compromise can be made with 3500 samples. At 3500 samples, training time has decreased dramatically but maintains close to optimal performance.

Given the current dataset and optimal tuning parameters (shrinkage = 1, iterations = 100), the decision tree algorithm scored a kappa of 86.04% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Use a different underlying classifier other than decision trees





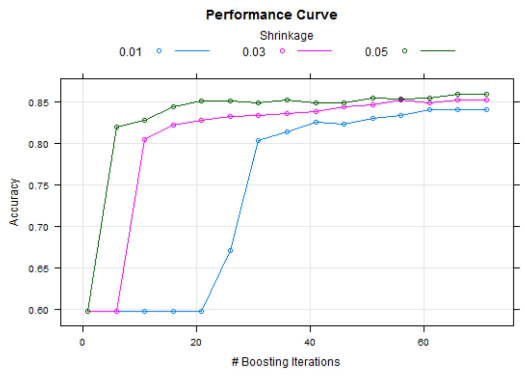


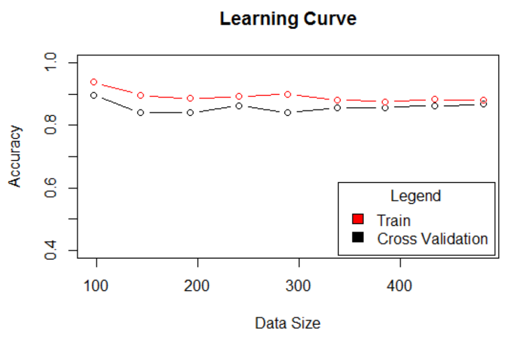
Wisconsin Breast Cancer Data: Using the Wisconsin Breast Cancer Data, the boosting model was tuned over several iterations and learnings rates (shrinkage). The charts are similar to those produced by HR Analytics Data in that the performance of the model improves with increases in iterations and learning rate. With more iterations the model has more opportunities to learn the data, by creating additional weak classifiers each iteration. With higher learning rates, the model learns the data faster by putting more weight on misclassifications, forcing the model to focus on those instances on the next iteration. Reference "Performance Curve" chart below.

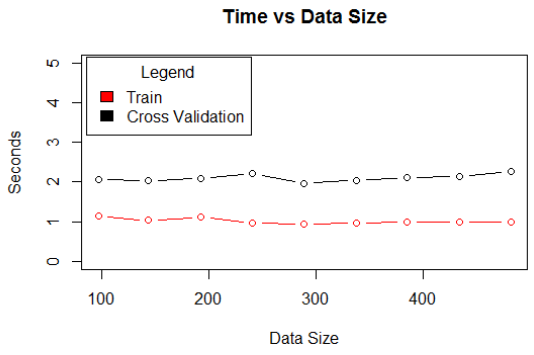
The learning curves show that given enough data, the curves will converge, indicating little to no variance. As mentioned above, this is no surprise. The boosting algorithm, makes its classifications by averaging over several weighted weak classifiers. By averaging its predictions over several classifiers, the boosting algorithm reduces its variance. Reference the "Learning Curve" chart below.

The model training times across various dataset sizes are consistent. Because the datasets is relatively small (< 500), there is not going to be a big difference in training times between the different dataset sizes. Consistent with the HR Analytics Data, the model takes more time to train with cross validation than it does training without cross validation. This is due to cross validation having to loop over the data several times. Reference the "Time vs Data Size" chart below.

Given the current dataset and optimal tuning parameters (shrinkage = 0.05, iterations = 71), the boosting algorithm scored an accuracy of 86.34% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Use a different underlying model, other than decision trees, to create weak classifiers.







Artificial Neural Networks (ANN)

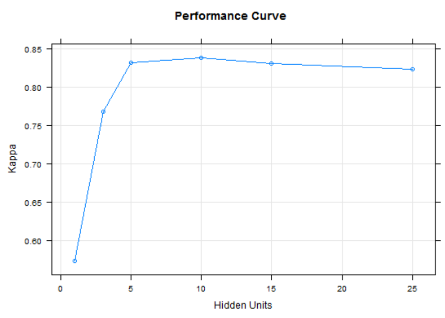
Most complex datasets can be solved with one or two layers. Based on the simplicity of the data being analyzed, ANN was trained on one layer over various numbers of nodes.

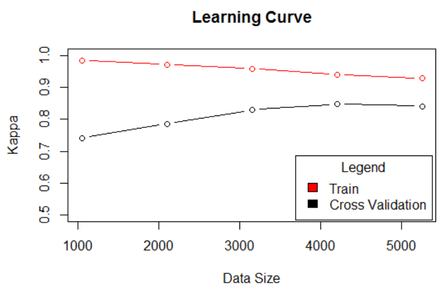
HR Analytics Data: Using the HR Analytics Data, the ANN algorithm was tuned over several nodes on one layer and performed best when there was five nodes. As the number of nodes increased towards five, the performance of the model improved, suggesting that the model was underfitting the data. As the value of nodes increases beyond five, the performance of the model slowly gets worse, suggesting that the model is overfitting the data. Reference the “Performance Curve” chart below.

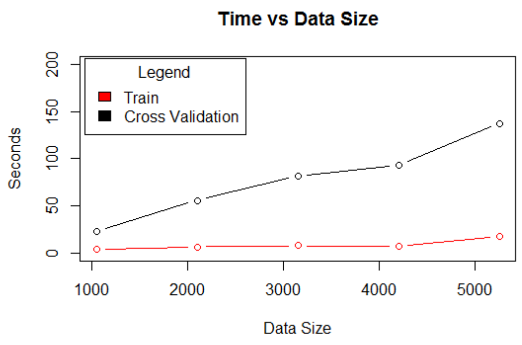
The learning curves show that the cross validation and training curves are converging with increases in dataset size. Training on more data will essentially lower error due to variance, resulting in the curves converging. Reference the “Learning Curve” chart below.

As the size of dataset gets larger, the time it takes to train the model increases much faster for cross validation than it does when training without cross validation. This is the case, because cross validation loops over the data “k-folds” to find validation error. Reference the “Time vs Data Size” chart below.

Given the current dataset and optimal tuning parameters (nodes = 5), the ANN scored a kappa of 83.70% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Train on more data to reduce error caused by variance





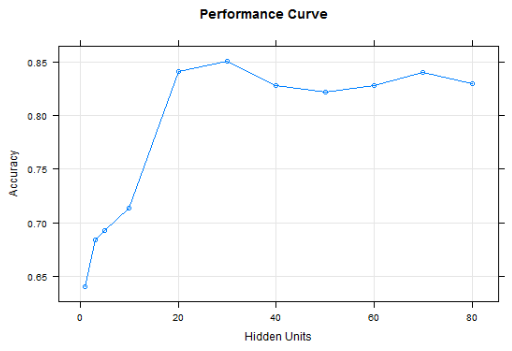


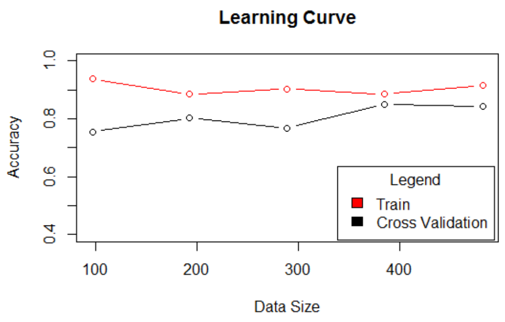
Wisconsin Breast Cancer Data: Using Wisconsin Breast Cancer Data, the ANN algorithm was tuned over several one layer nodes and performed best when there was 30 nodes. The performance curve is similar to the one created when ANN was trained on the HR Analytics Dataset. As the model approaches its optimal configuration (30 nodes) it underperforms due to underfitting, and as it exceeds its optimal configuration it underperforms due to overfitting. Reference the "Performance Curve" chart below.

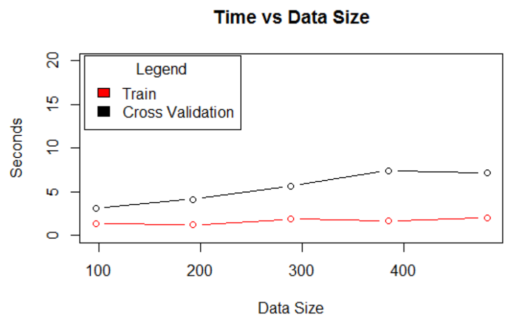
The learning curves that were generated using the dataset showed consistent performance across various dataset sizes. This suggests that the errors introduced by variance and bias are consistent across different sizes of data. This may be due to the simplicity of the true data's pattern and the ease of which the pattern can be captured in smaller datasets. Reference the "Learning Curve" chart below.

The size of the data adversely affects the time to process the data, when performing cross validation. This is similar to the results that were received when the HR Analytics Data was analyzed. The cost of doing cross validation affects small data sets (<500) the same way it affects large datasets (> 5000). Because performance is consistent across different data sizes, the data size can be reduced to reduce time consumption without loss of performance. Reference the "Time vs Data "chart below.

Given the current dataset and optimal tuning parameters (nodes = 30), the ANN algorithm scored an accuracy of 86.83% during testing. To improve on the performance of the algorithm, the following approaches can be explored: 1) Remove features to eliminate errors due to variance.







Dataset Summaries

This section is dedicated to highlighting conclusions draw about each dataset (HR Analytics Data and Wisconsin Breast Cancer Data), based on the different algorithms that were performed.

HR Analytics Data: Overall, the decision tree performed better than the other algorithms with a kappa of 91.75% during testing. You can see a comparison of performances below:

	KNN	Decision Tree	SVM (Linear)	SVM (Radial)	Boosting	ANN
Kappa	84.62%	91.75%	22.94%	90.16%	86.04%	83.70%

Given that the data was slightly imbalanced, all the algorithms handled the data well and was able to find the patterns in the data without the need to under sample or over sample the data. When using the Linear Kernel, the SVM algorithm did not do very well, suggesting that the data is far from following a linearly separable pattern. On the other hand, the KNN algorithm performed well with a small k and the SVM algorithm with a radial kernel performed well with a moderate value of sigma. This is evidence supporting the fact that the minority set is locally grouped throughout the feature set.

Wisconsin Breast Cancer Data: Overall, the SVM with a radial kernel performed better than other algorithms with an accuracy of 91.22% during testing. You can see a comparison of performances below:

	KNN	Decision Tree	SVM (Linear)	SVM (Radial)	Boosting	ANN
Accuracy	90.73%	88.29%	88.78%	91.22%	86.34%	86.83%

The biggest conclusion gleamed from the different analysis is that the data is nearly linearly separable. The data performed comparably well with SVM using a linear kernel. It is possible that the true pattern of the data is linearly separable, and the noise introduced into the data is causing the error. The simplicity of the data and the fact that its nearly linearly separable is also supported by the fact that a decision tree of depth = 1 performs within 6% of the most optimal decision tree.

Algorithm Summaries

This section is dedicated to highlighting conclusions drawn for each algorithm (KNN, Decision Tree, SVM, Boosting, ANN) based on how they performed on both datasets.

KNN: For KNN, increasing k decreases variance and increases bias. The opposite is true if you decrease k. This is reflected in the learning curves of both datasets. For the HR Analytics Dataset, the value of k is low. The training curve is consistently at 100% indicating low bias, but the gap between the training curve and cross validation curve is large indicating high variance. With a higher value of k, the Wisconsin Breast Cancer Dataset tells a different story. The training curve is no longer at 100% indicating that some bias is introduced, but the training and cross validation curves have converged indicating little to no variance.

KNN also favors lower values of k, when the data is imbalanced, and locally grouped throughout the space. As explained earlier, higher values of k allow for the opportunity to include more of the majority class when classifying the minority class. Therefore a smaller k is favored.

Decision Tree: When constructing decision trees, often times information gain and index gini are interchangeable, providing similar results. In the analysis of the HR Analytics Dataset and Wisconsin Breast Cancer Dataset, both information gain and index gini were used to construct the trees. In both cases, the performance was very similar.

SVM: When compared to the linear kernel, the radial kernel has low bias. For both datasets, the HR Analytics Dataset and the Wisconsin Breast Cancer Dataset, the learning curve for training error consistently scored a 100% across various data sizes, showing evidence of it low bias nature. With the kernel being low bias, it is easy for it to overfit the data. For the Wisconsin Breast Cancer Dataset, the learning curve for training error was able to fit the noise in the dataset to achieve a 100% accuracy when the radial kernel was used.

Boosting: By design, the boosting algorithm has low variance. In both datasets, the HR Analytics Dataset and Wisconsin Breast Cancer Dataset, the learning curves converged, which is evidence of low variance. The algorithm builds several weak learners and generates classifications by performing a weighted average over the weak learners. The act of averaging the weak classifiers, reduces the variance in the algorithm's response.

Also, increasing the number of iterations and rate of learning increases performance of the algorithm. They both effectively dictate how much is learned by the algorithm, by managing how many times it passes over the data and how much focus to put on misclassifications.

ANN: Increasing nodes within neural network can increase the complexity of the network causing it to easily overfit the data. The algorithm is also very sensitive to data size, even at smaller datasets.