Assignment 1: Supervised Learning

CS 7641 Machine Learning
Spring 2017

Karel Klein Cardeña

Datasets

Pima Indians Diabetes Dataset

This data set is a subset from the database owned by the National Institute of Diabetes and Kidney Diseases. The 768 instances selected from the superset were chosen with the constraints that the patient be a female at least 21 years old and of Pima Indian heritage. Each instance, corresponding to one person, contains eight numeric attributes describing factors such as plasma glucose concentration, body mass index, age, and other quantitative measures of relevance to the presence of diabetes. Each instance contains a binary label: "YES" if the patient has diabetes, and "NO" otherwise.

Early detection of diabetes is crucial in preventing or delaying the onset of serious health complications. An untreated diabetic will face an increased risk of kidney disease, high blood pressure, and stroke -- conditions whose effects may be irreversible and potentially fatal. Certain risk factors for diabetes, especially Type I, are inherited from both parents and thus correlated to the patient's genetic pool. It is thus very relevant to study an ethnic subset of the total population in order to deduce both the group's propensity to develop diabetes as well as to further assess the onset of diabetes as a function of genetics. Correctly forecasting diabetes using only simple health metrics could lead to a dramatic reduction in long-term detriment to a person's health by inciting them to get early treatment and diagnosis. Because the metrics in this study are very easy to obtain, a method of predicting diabetes using these attributes alone would be especially helpful to historically marginalized groups with little access to advanced medical institutions.

Census Income Dataset

This is subset of the data obtained from the UCI Machine Learning Repository, which itself is a subset extracted from the United States' 1994 census database found at www.census.gov. The dataset contains 5000 instances, each corresponding to one individual, each with 12 attributes and one binary class corresponding to income level. Thus, this data attempts to correlate attributes such as age, education, race, sex, and native country to a discretized income class: "above50" if the individual earns more than $50,000 per year, and "below50" otherwise.

The ability to predict income level based on demographics has proven to be essential for the understanding of consumer behavior and producing target marketing. Income prediction models allow businesses and institutions to use customer attributes to tailor their marketing strategy according to the income class they wish to target. A dental franchise, for example, could use customer demographics to send toothbrush offers based on predicted income. Indeed, it may be futile and even offensive to offer a $400 electric brush deal to an individual who earns $20,000 per year. Accurate prediction of income would thus lead to more a consumer-sensitive marketing strategy.

Machine Learning Context

While both datasets present a binary classification problem, each do so with fundamentally different data. In the case of the Diabetes set, we are given eight numeric attributes and only 768 instances, a ratio of 96 instances per attribute. In the Income data, we are given 12 attributes, almost all of which contain nominal values, and 5000 instances. This ratio of instances per attribute is more than four times that of the Diabetes. Given this inherent disparity, it will be interesting to assess what role the curse of dimensionality plays in prediction accuracy.

Further, the difference between the two sets' attribute types serves as another field for comparison across the five algorithms. Interestingly, the best classifier for the Income data correctly classified 3% better than its worst classifier, while this difference was 8% in the Diabetes data. Additionally, the Income data classifiers performed an average of 10% better than did those of Diabetes. Given the discrepancies in results , these data provide a good vehicle for comparison of the performance and bias of each learning algorithm.

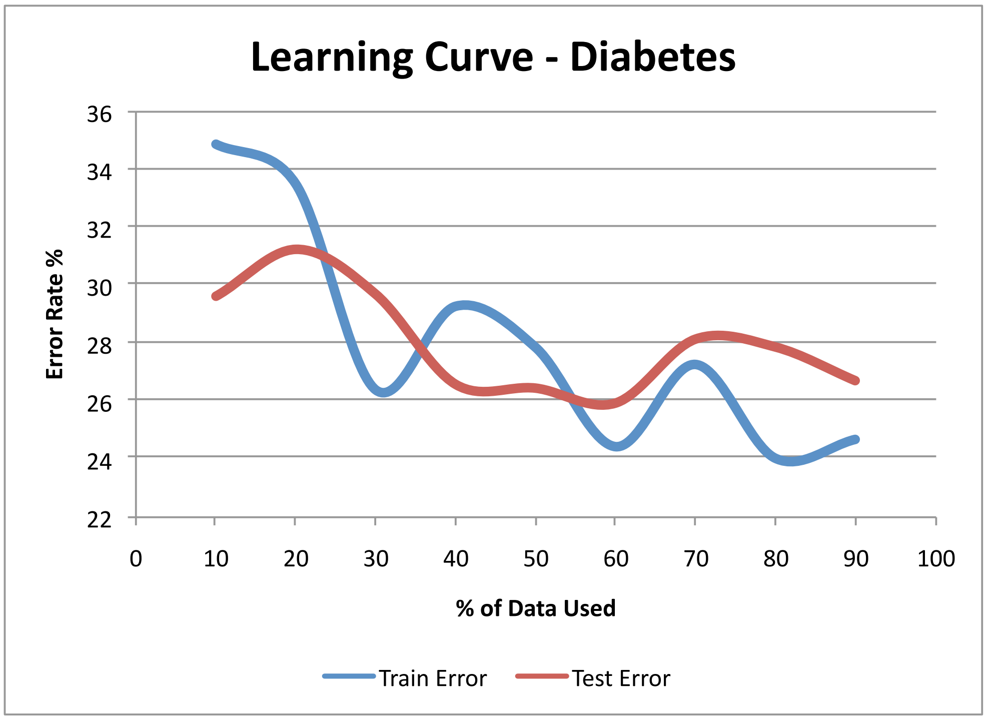
Decision Trees

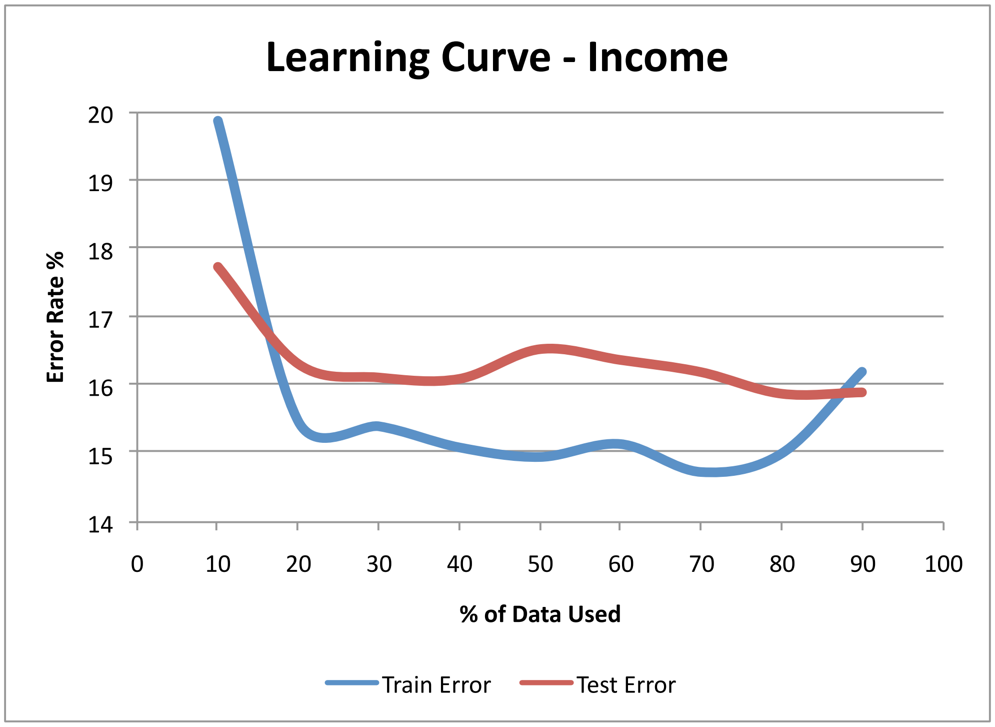
Decision Tree	Hyperparameter			Diabetes Error Rate		Income Error Rate	
Algorithm	Pruning	Pruning Factor	Min instances per leaf	Training	Testing	Training	Testing
J48	Yes	0.125	2	0.2404	**0.2476**	0.1500	0.1697
J48	Yes	0.25	2	0.2338	0.2590	0.1530	0.1700
J48	Yes	0.5	2	0.2532	0.2590	0.1570	0.1785
J48	No	-	2	0.2532	0.2655	0.1740	0.1887
J48	Yes	0.125	4	0.2403	0.2492	0.1500	**0.1687**
J48	Yes	0.125	8	0.2078	0.2508	0.1500	0.1707

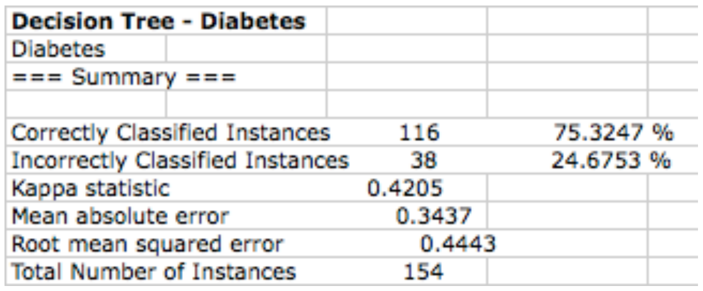
Decision trees proved to be one of the best classifiers in terms of Error Rate. To find the optimal hyper-parameters, various pruning factors were examined, as well as the minimum number of instances per leaf. In this analysis, a lower pruning factor corresponds to more aggressive pruning.

In the Diabetes data, we first hold min. instances/leaf constant in an effort to evaluate the effect of pruning. Interestingly, a pruning factor of 0.125 (the most aggressive of the set) results in the second highest training error rate at 24.04%, but the lowest testing error at 24.76%. By reducing the size of the decision tree, this pruning factor reduces the accuracy in the training set in an effort to avoid overfitting, and realizes its potential by best generalizing the test data. Once the optimal pruning factor was found, the instances per leaf could be varied to find optimality. The results showed that a minimum of 2 instances per leaf produced the lowest test error, despite having a minimum of 8 produce the lowest training error.

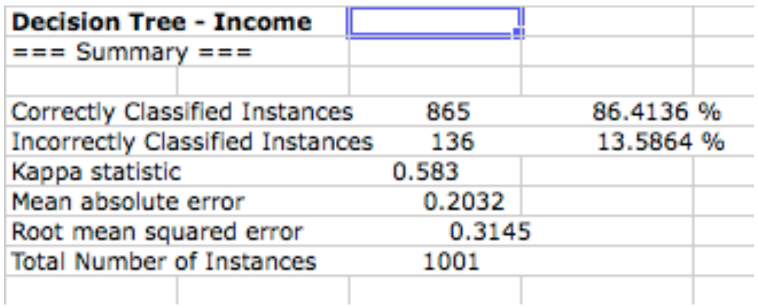
In the Income data, we likewise see that a pruning factor of 0.125 is most effective. When testing various min. instances/leaf, however, we find that a minimum of 4 produces the smallest test error from the set {2,4,6}. In a model complexity curve, this point corresponds to the minimum of the curve representing test error. This is likely an indication that as the complexity increases, the test set continues performing better until this inflection point is reached, whereupon an increase in complexity yields worse generalization. Given more time to conduct experiments, I would increase the range of pruning factor as well as the minimum instances per leaf to get a better sense of the relation between these magnitudes and the efficacy of the decision tree.







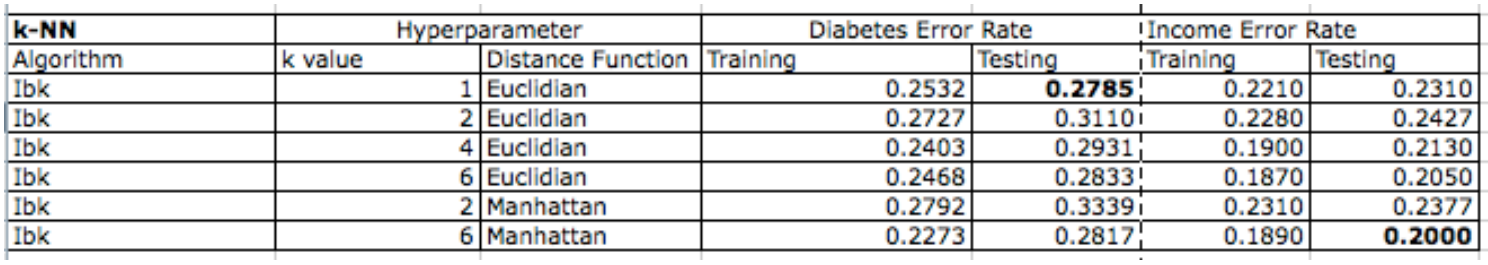
Decision Tree - Diabetes		
Diabetes		
=== Summary ===		
Correctly Classified Instances	116	75.3247 %
Incorrectly Classified Instances	38	24.6753 %
Kappa statistic	0.4205	
Mean absolute error	0.3437	
Root mean squared error	0.4443	
Total Number of Instances	154	



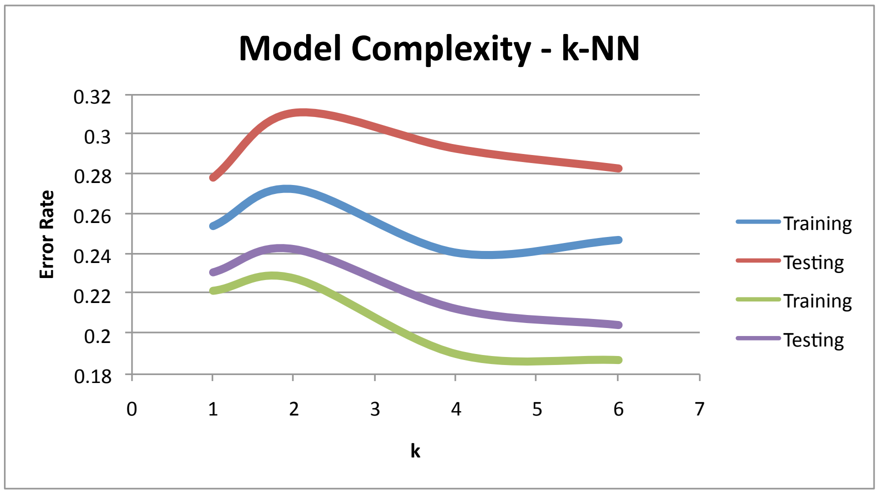
Decision Tree - Income		
=== Summary ===		
Correctly Classified Instances	865	86.4136 %
Incorrectly Classified Instances	136	13.5864 %
Kappa statistic	0.583	
Mean absolute error	0.2032	
Root mean squared error	0.3145	
Total Number of Instances	1001	

Once the optimal hyper-parameters were found, learning curves were constructed to visualize the effect of varying data size on training and test errors. In both cases, it is clear to see that as more data is fed to the the decision tree, the smaller the error rate gets. An interesting distinction of between the two datasets is the smoothness of their curves. In the Diabetes learning curve, we see a high degree of jitter and volatility in the error curves, while in the Income data we see smooth lines. This could in part be due to the relative sizes of the data sets: with a training sample size of 614 and testing sample size of 153, the Diabetes Decision Tree is likely to suffer in its ability to generalize unseen data. Using 90% of training data we can see that the training and testing error still hold a visible gap in error rate. However, we see that this gap is slowly converging. This leads to the belief that its high degree of variance may be remedied by utilizing a larger data set.

k-NN

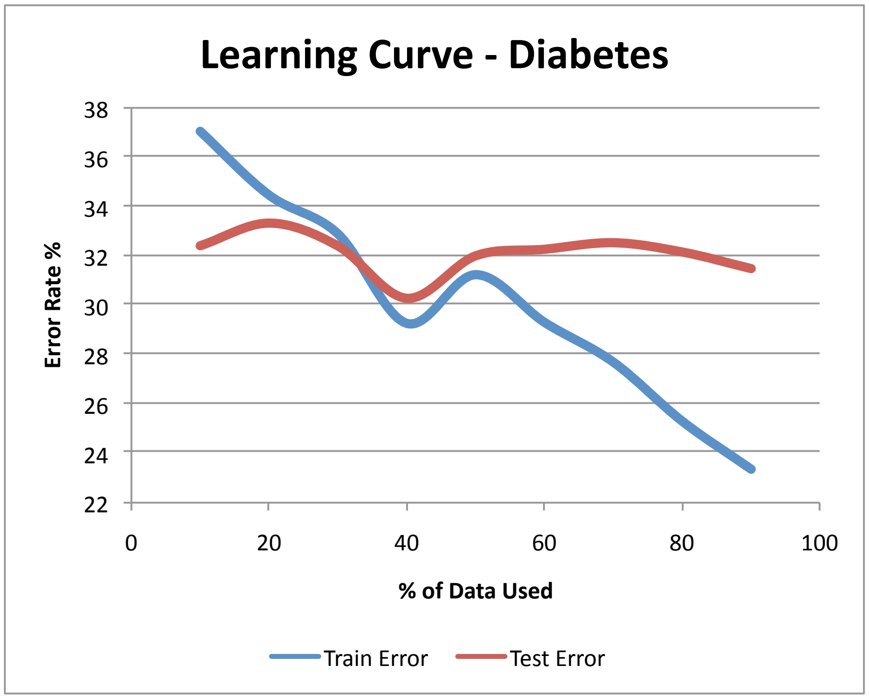


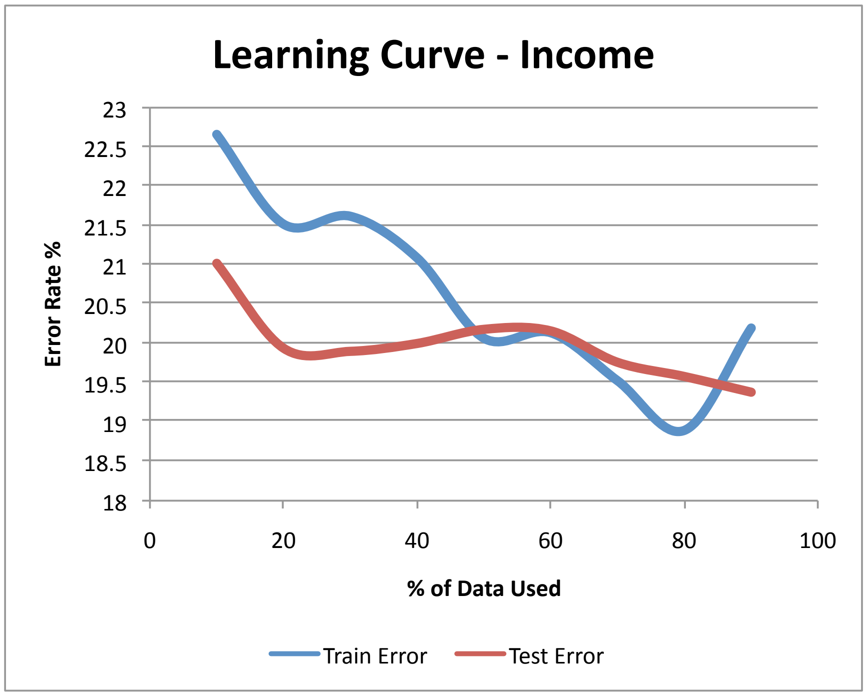
k-NN	Hyperparameter		Diabetes Error Rate		Income Error Rate	
Algorithm	k value	Distance Function	Training	Testing	Training	Testing
Ibk	1	Euclidian	0.2532	**0.2785**	0.2210	0.2310
Ibk	2	Euclidian	0.2727	0.3110	0.2280	0.2427
Ibk	4	Euclidian	0.2403	0.2931	0.1900	0.2130
Ibk	6	Euclidian	0.2468	0.2833	0.1870	0.2050
Ibk	2	Manhattan	0.2792	0.3339	0.2310	0.2377
Ibk	6	Manhattan	0.2273	0.2817	0.1890	**0.2000**



As two of the driving features of the k-NN algorithm, I looked to vary the value of *k* as well as the distance function in order to find the best hyper-parameters for each dataset. While using a Euclidian measure of distance, the Diabetes data found a *k* value of 1 to yield the best test error rate of 27.85% while the Income data found *k* = 6 as the optimal setting. Analyzing the model complexity curve (blue/red lines = Diabetes, purple/green = Income), we can see all error rates seem to be decreasing as we increase the value of *k*. This leads me to believe that testing a greater range of *k* would allow one to see the full body of the complexity curve and find that the optimal value of *k* for Diabetes is indeed greater than 1. The allotted time to conduct these experiments did not allow for this phenomenon to be confirmed.

Using the optimal hyper-parameters, the model is trained on 80% of the data using 3-fold cross validation, and tested on 20% of the data. As can be seen in the summary of the results, *k*-NN had an error rate of 32.47% and 17.98% on the Diabetes and Income datasets, respectively. These numbers are considerably higher than those of the Decision Tree algorithm. In respect to the Income data, I believe this is to be in part due to the high dimensionality of the data. With 12 attributes and 2 classes, it is likely that some of these attributes do not provide information gain to the classification problem. As a result, many instances who share these irrelevant attributes may be near each other without, in fact, being in the same class. The nearest neighbor search will thus result in a high rate of false matches. The Decision Tree, conversely, specializes in finding attributes that yield the highest information gain and so we can expect it to carry a much lower error rate than *k*-NN in a high dimensional space. This belief is supported by the observed error rates on both sets of test data.





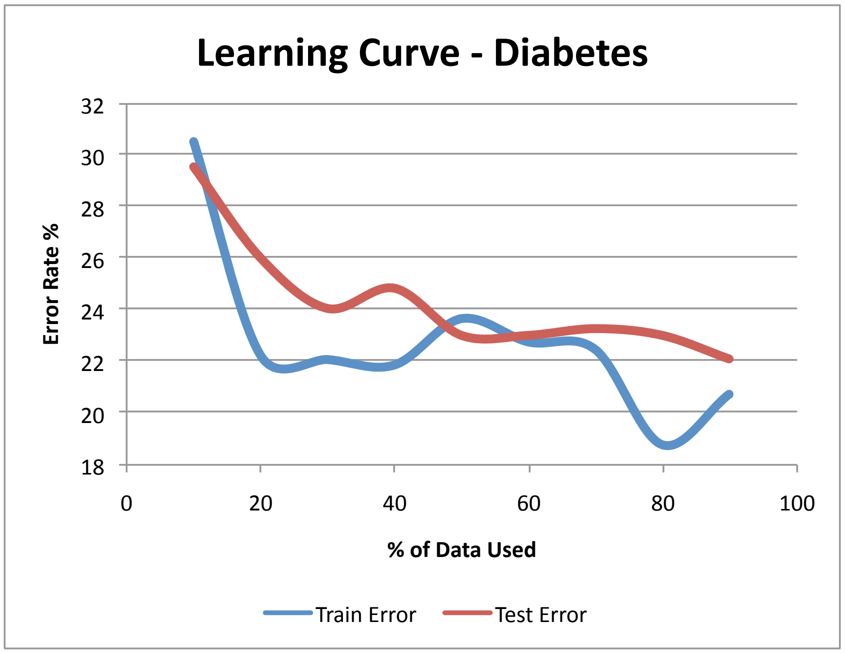
k-NN - Diabetes		
=== Summary ===		
Correctly Classified Instances	104	67.5325 %
Incorrectly Classified Instances	50	32.4675 %
Kappa statistic	0.3042	
Mean absolute error	0.3252	
Root mean squared error	0.5689	
Total Number of Instances	154	

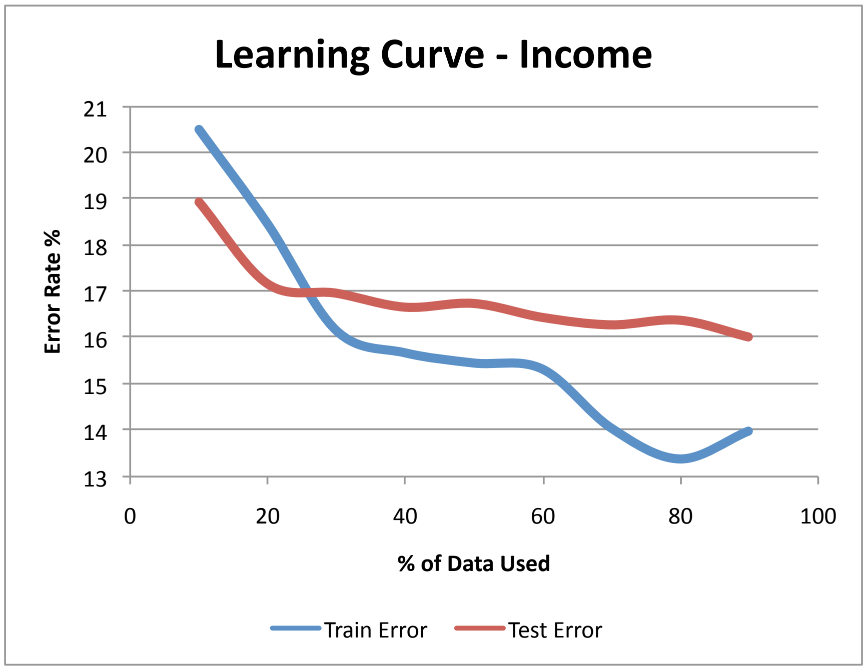
k-NN - Income		
=== Summary ===		
Correctly Classified Instances	821	82.018 %
Incorrectly Classified Instances	180	17.982 %
Kappa statistic	0.5124	
Mean absolute error	0.2123	
Root mean squared error	0.3558	
Total Number of Instances	1001	

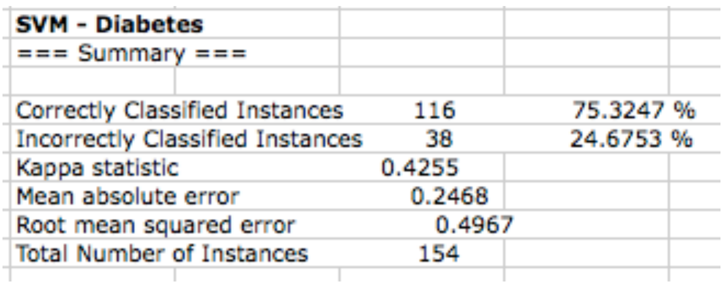
Support Vector Machines

SVM	Hyperparameter		Diabetes Error Rate		Income Error Rate	
Algorithm	Kernel	C value	Training	Testing	Training	Testing
SMO	PolyKernel	1	0.2013	0.2394	0.136	0.1650
SMO	Puk	1	0.1883	0.2491	0.177	0.1995
SMO	RBFKernel	1	0.3506	0.3469	0.157	0.1815
SMO	PolyKernel	2	0.1883	**0.2312**	0.132	0.1605
SMO	PolyKernel	4	0.1948	0.2329	0.134	**0.1598**

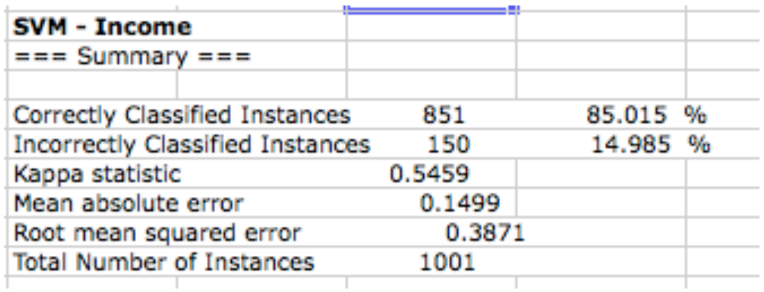
In order to find the type of Support Vector Machine that best classified the training data, I was interested in evaluating the effect of the Kernel and that of the complexity parameter C. Holding C constant, I experimented with a polynomial kernel, the Pearson universal kernel, and the RBF kernel. Across both datasets, the polynomial kernel showed dramatically better error rates, especially compared to the RBF kernel: 11% and 2.6% better in Diabetes and Income, respectively. Using cross-validation, increasing the complexity parameter also improved the performance of SVM. In the Diabetes testing set, we see the error rate move from 23.94% down to 23.12%, and back up to 23.29% as the C value is changed from 1 to 4.







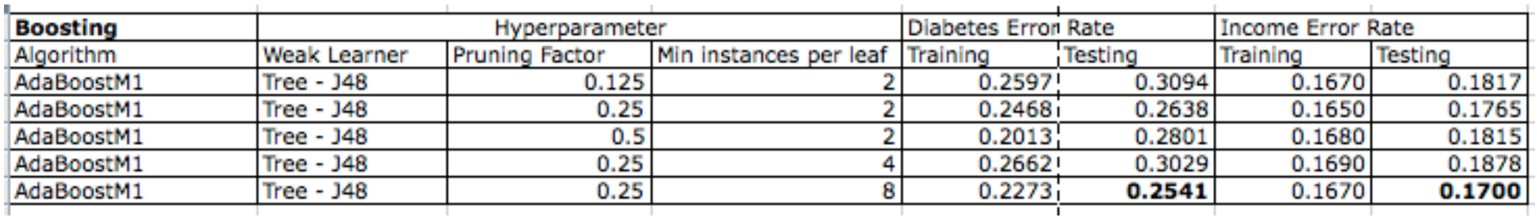
SVM - Diabetes		
=== Summary ===		
Correctly Classified Instances	116	75.3247 %
Incorrectly Classified Instances	38	24.6753 %
Kappa statistic	0.4255	
Mean absolute error	0.2468	
Root mean squared error	0.4967	
Total Number of Instances	154	



SVM - Income		
=== Summary ===		
Correctly Classified Instances	851	85.015 %
Incorrectly Classified Instances	150	14.985 %
Kappa statistic	0.5459	
Mean absolute error	0.1499	
Root mean squared error	0.3871	
Total Number of Instances	1001	

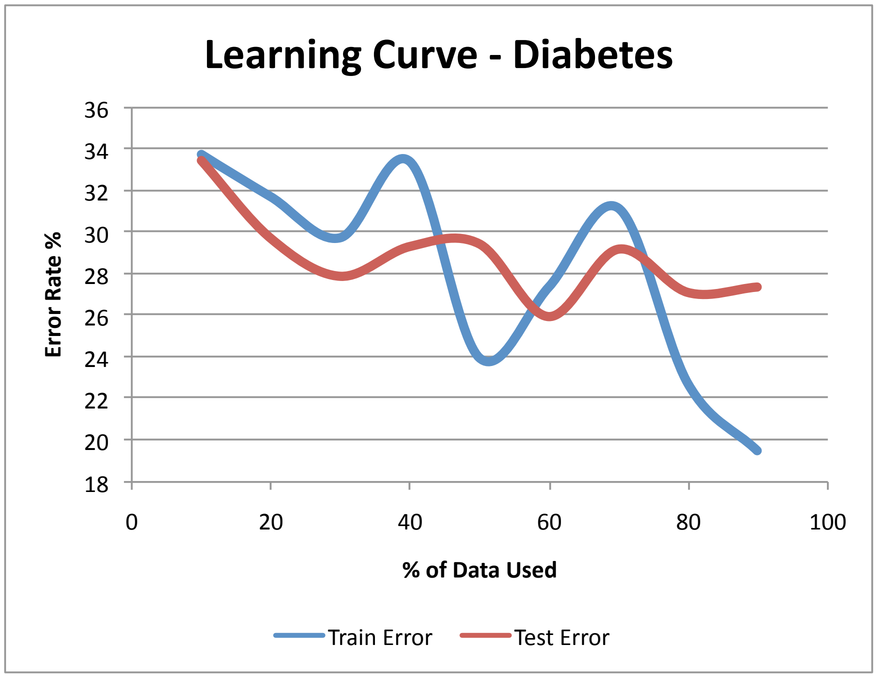
With the optimal parameters in place, a learning curve was plotted in each data set. Looking at the Income learning curve, we see that the test error decreases very slightly as a function of the percentage of data used. This feature is the impetus for the hypothesis that more data will not alleviate the testing error in this dataset. The SVM inductive bias tells us that distinct classes will be separated by wide margins. The income classes of “above50” and “below50”, however, imply that an individual who earns $49k and one who earns $51k will be classified differently, despite potentially sharing a majority of attributes. As a result, distinct classes may be separated by very small margins. SVM will struggle in these examples regardless of the amount of data it trains with.

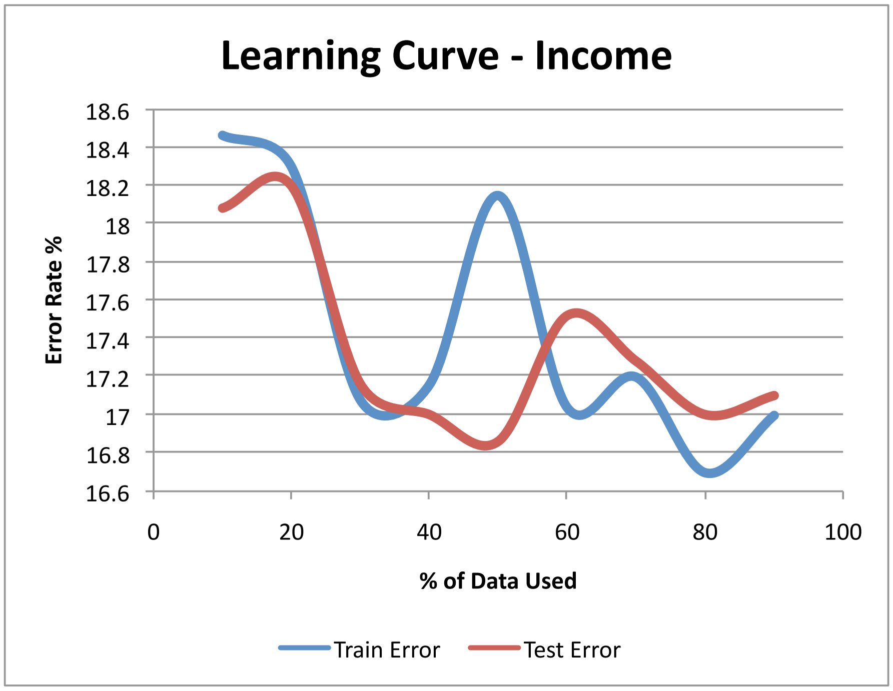
Boosting

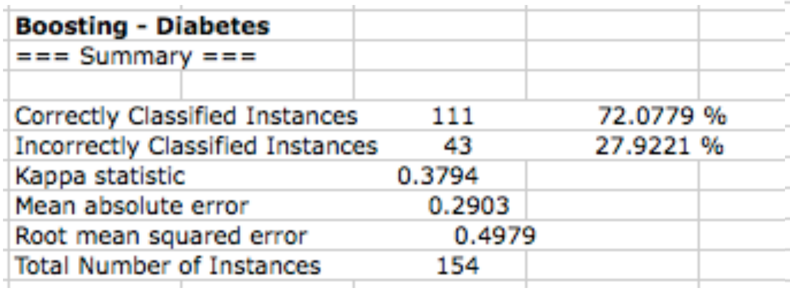


Boosting	Hyperparameter			Diabetes Error Rate		Income Error Rate	
Algorithm	Weak Learner	Pruning Factor	Min instances per leaf	Training	Testing	Training	Testing
AdaBoostM1	Tree - J48	0.125	2	0.2597	0.3094	0.1670	0.1817
AdaBoostM1	Tree - J48	0.25	2	0.2468	0.2638	0.1650	0.1765
AdaBoostM1	Tree - J48	0.5	2	0.2013	0.2801	0.1680	0.1815
AdaBoostM1	Tree - J48	0.25	4	0.2662	0.3029	0.1690	0.1878
AdaBoostM1	Tree - J48	0.25	8	0.2273	**0.2541**	0.1670	**0.1700**

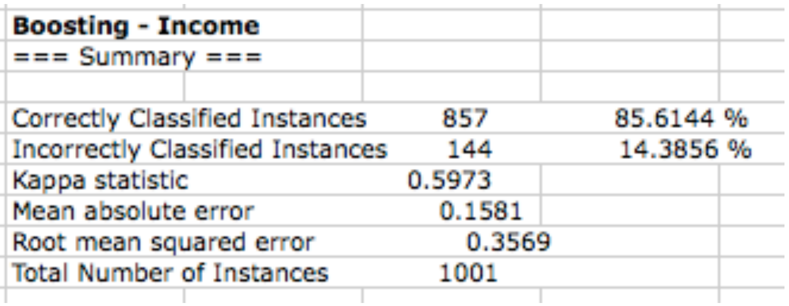
Using the J48 Decision Tree algorithm as the weak learner, I performed a series of Boosting experiments with variations in the pruning factor and minimum instances per leaf parameters. Interestingly, Boosting performed worse across all hyper-parameter variations than did the Decision Tree alone. Given the relatively small amount of sample data used, it is possible that with a larger set the error rates would prove lower than the standalone Decision Tree. Given the aforementioned problem faced by SVM with the Income data, it is again probable that this overlapping of both classes hinders the typical lowering of error that Boosting provides over its weak learner. It would be interesting to discretize the Income data with higher granularity, such that we may have 10 income levels as opposed to 2.





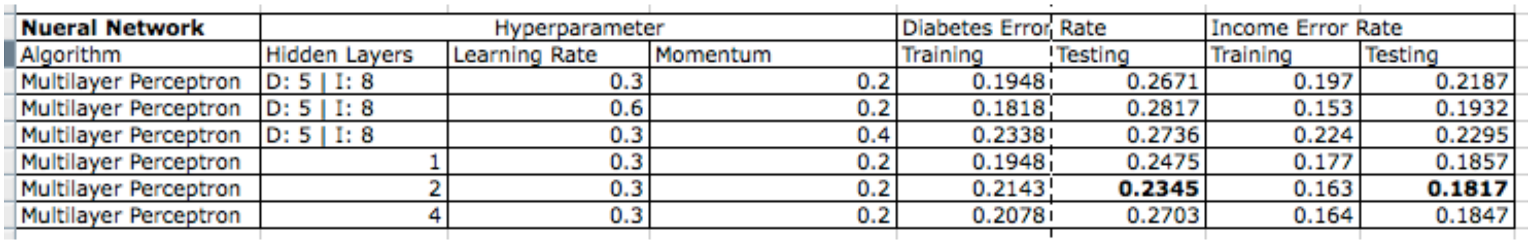


Boosting - Diabetes		
=== Summary ===		
Correctly Classified Instances	111	72.0779 %
Incorrectly Classified Instances	43	27.9221 %
Kappa statistic	0.3794	
Mean absolute error	0.2903	
Root mean squared error	0.4979	
Total Number of Instances	154	



Boosting - Income		
=== Summary ===		
Correctly Classified Instances	857	85.6144 %
Incorrectly Classified Instances	144	14.3856 %
Kappa statistic	0.5973	
Mean absolute error	0.1581	
Root mean squared error	0.3569	
Total Number of Instances	1001	

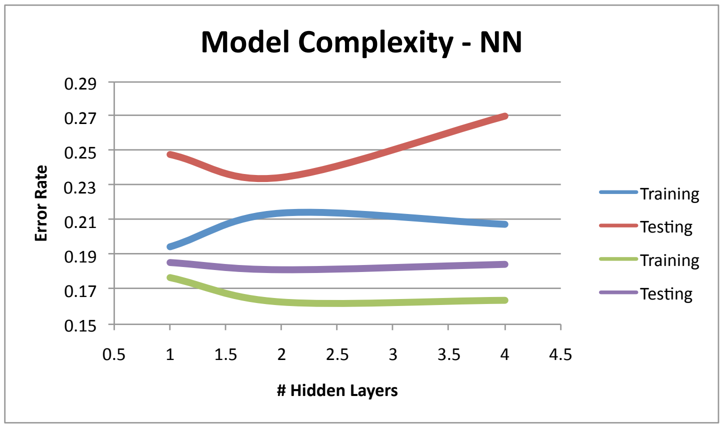
Neural Networks

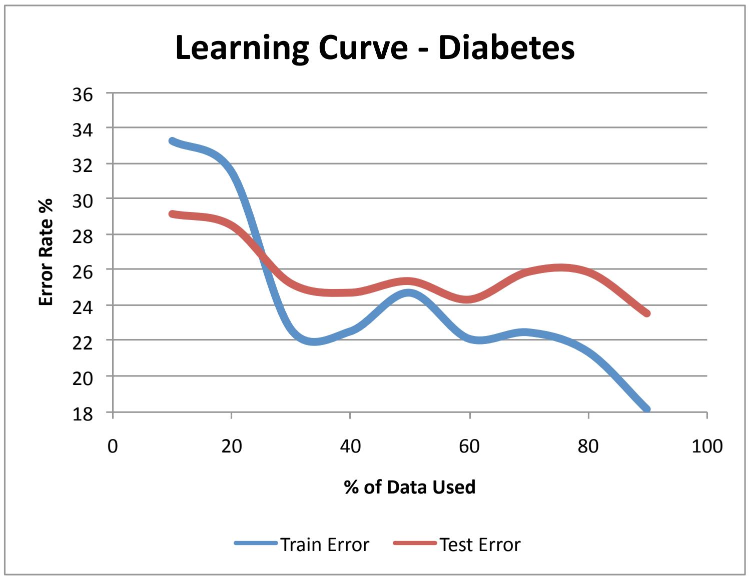


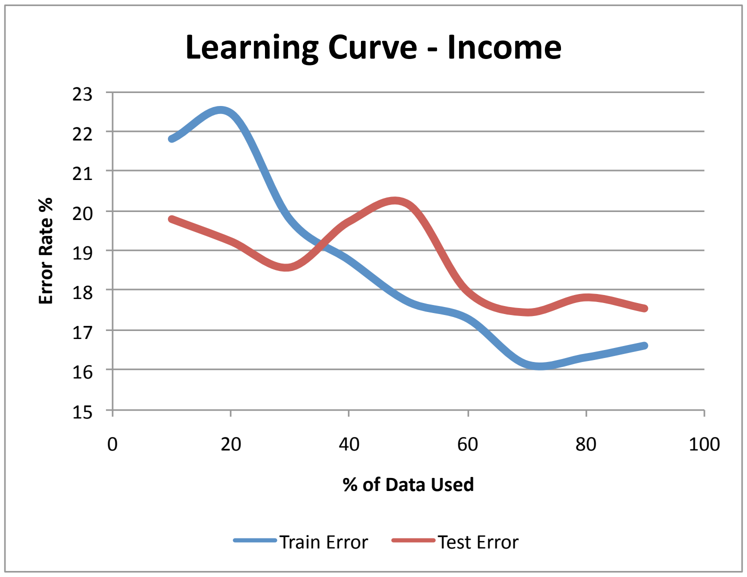
Nueral Network	Hyperparameter			Diabetes Error Rate		Income Error Rate	
Algorithm	Hidden Layers	Learning Rate	Momentum	Training	Testing	Training	Testing
Multilayer Perceptron	D: 5 | I: 8	0.3	0.2	0.1948	0.2671	0.197	0.2187
Multilayer Perceptron	D: 5 | I: 8	0.6	0.2	0.1818	0.2817	0.153	0.1932
Multilayer Perceptron	D: 5 | I: 8	0.3	0.4	0.2338	0.2736	0.224	0.2295
Multilayer Perceptron	1	0.3	0.2	0.1948	0.2475	0.177	0.1857
Multilayer Perceptron	2	0.3	0.2	0.2143	**0.2345**	0.163	**0.1817**
Multilayer Perceptron	4	0.3	0.2	0.2078	0.2703	0.164	0.1847

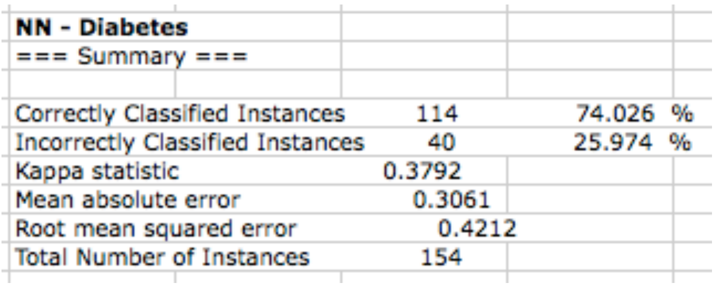
Three hyper-parameters were tested in the Neural Networks experiments: number of hidden layers, learning rate, and momentum. While changing the learning rate and momentum did alter the error slightly, the largest improvement was observed in altering the number of hidden layers. Specifically, both datasets performed best when using only two hidden layers.

Looking at the Diabetes model complexity curve specifically (red/blue) , we can clearly tell that after the inflection point *h* = 2, the training error continues to decrease, while the testing error decidedly increases. From this data we can deduce that the NN model begins to overfit when the complexity increases beyond 2 hidden layers.

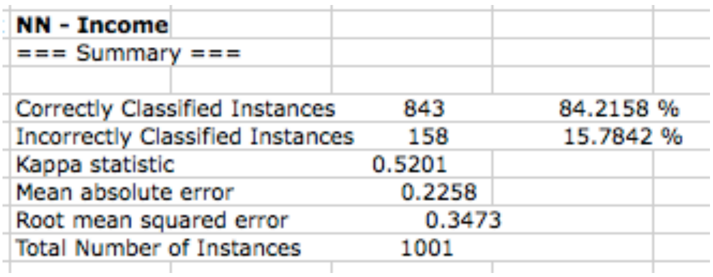








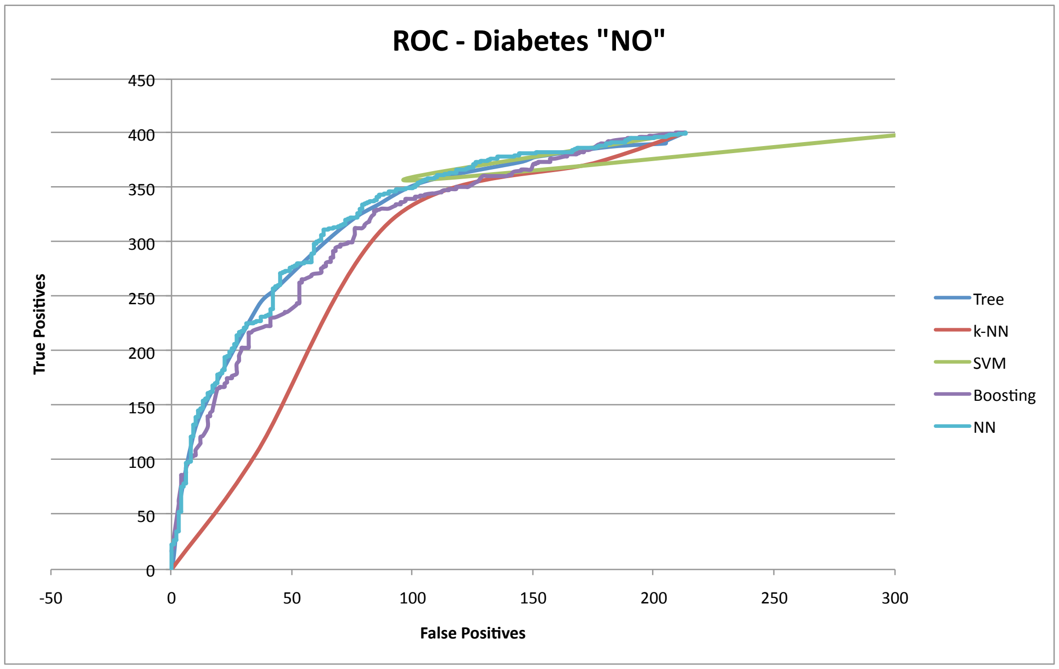
NN - Diabetes		
=== Summary ===		
Correctly Classified Instances	114	74.026 %
Incorrectly Classified Instances	40	25.974 %
Kappa statistic	0.3792	
Mean absolute error	0.3061	
Root mean squared error	0.4212	
Total Number of Instances	154	



NN - Income		
=== Summary ===		
Correctly Classified Instances	843	84.2158 %
Incorrectly Classified Instances	158	15.7842 %
Kappa statistic	0.5201	
Mean absolute error	0.2258	
Root mean squared error	0.3473	
Total Number of Instances	1001	

Although the Neural Network with the optimal hyper-parameters performed better than several of the other learners, it failed to outperform the basic Decision Tree. In terms of training time it is also costly; the NN took 10,500 seconds for 3-fold cross validation on a training set of 4,000 instances, whereas the Decision Tree took 3 seconds. Given more time to experiment, it would be interesting to see the performance of the NN when stripping features from the datasets.

ROC



The ROC curve for the Diabetes “NO” interestingly displays the relative performance of the 5 learners. The decision tree and NN had very similar curves. Taking into consideration their relative test error rate (24.67% Tree, 25.97% NN), training time difference, and model complexity, the Decision Tree algorithm proved the best performer. However, the suggested experiments in this review should be carried out before coming to a firm conclusion. Indeed, forecasting the onset of diabetes correctly is of extreme importance, as someone could forego preventative treatment if incorrectly classified.