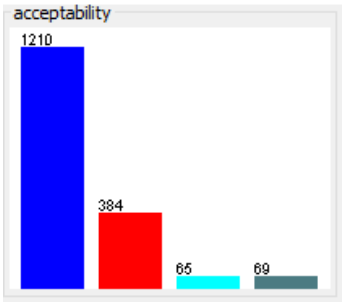
Jassimran Kaur Somal ***Assignment1***
GT-ID : jkaur9

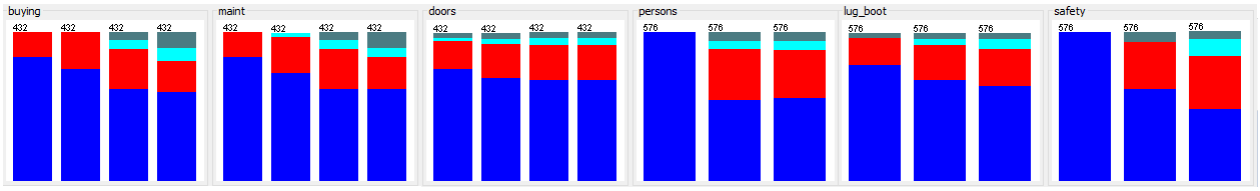
DATASETS

Car Evaluation Data Set: This dataset is obtained from UCI dataset repository and it has 6 attributes and 1728 instances. The task is to classify the car as Acceptable- 'acc', Good- 'good', Unacceptable- 'unacc' and Very Good- 'vgood' based on its price, technical/comfort features and safety.

Why it's interesting?

Having domain knowledge is really important to not only excel but one needs to know what you are really looking for. Hence, Car dataset was very simple to understand as I recently bought a car. Safety, luxury and cost are all important factors when buying car and their importance varies from person to person. This dataset has 6 attributes, minimizing the curse of dimensionality. With respect to machine learning, this dataset has close to 70% cars which are not acceptable but it still performs differently on few different algorithms despite uneven distribution.



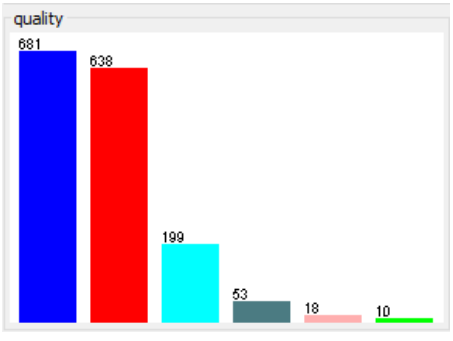


However, the distribution of all other attributes is even making it even more interesting.

Red Wine Data Set: This dataset is obtained from UCI dataset repository and it has 12 attributes and 1599 instances. The task is to classify the wine quality as a score between 0-10 based on physicochemical and sensory variables.

Why it's interesting?

Firstly, it gives me a chance to understand wines better and what make them all so different. This dataset is interesting because it has approximately same number of instances as car evaluation dataset but has double the number of attributes. I wanted to explore if curse of dimensionality would start kicking in. Also, classes are not balanced as there are much more normal wines than excellent or poor one. And it has to be classified over a range of 10 different values, so giving me chance to see how algorithms behave differently for 10 classifier values.



Decision trees

Car Evaluation data Set

	PRUNED Decision tree	UNPRUNED Decision Tree
Number of Leaves	100	131
Size of the tree	139	182
Accuracy %	95.04%	96.14%
Stratified cross-validation Accuracy	89.42%	91.81%
Time taken to build model (seconds)	0.02	0.01
Time taken to test model(seconds)	0.01	0.01
Confusion Matrix	**a b c d <-- classified as** 247 4 0 1 | a = unacc 7 73 0 1 | b = acc 0 1 12 1 | c = vgood 0 2 1 13 | d = good	**a b c d <-- classified as** 245 6 0 1 | a = unacc 2 78 0 1 | b = acc 0 1 13 0 | c = vgood 0 1 2 13 | d = good

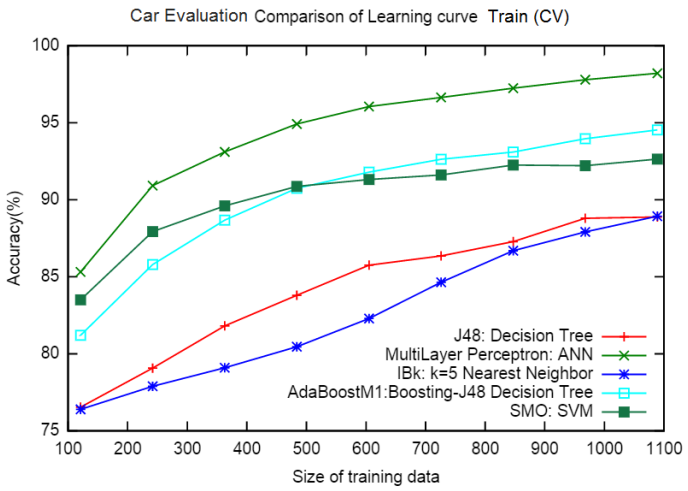
Pruning State	Confidence	# Leaves	Tree Size	Train%	Test%	Train Time	Test Time
Pruned	.125	71	98	92.56	92.56	.01	.01
Pruned	.25	100	139	94.87	95.04	.02	.01
Pruned	.5	116	161	95.95	95.59	.03	.01
Pruned	.75	131	182	96.61	96.14	.05	.01
Unpruned	---	131	182	96.61	96.14	.01	.01

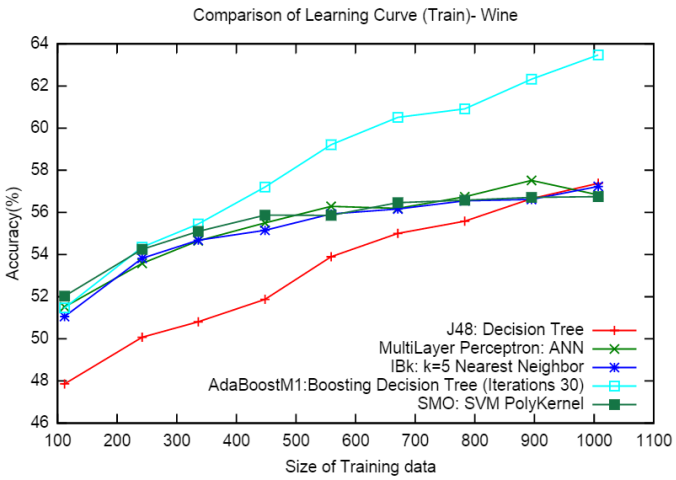
Wine Data Set

	PRUNED Decision tree	UNPRUNED Decision Tree
Number of Leaves	124	168
Size of the tree	247	335
Accuracy %	58.33% (confidence .125)	58.12%
Stratified cross-validation Accuracy	56.74	56.83
Time taken to build model (seconds)	0.05	0.04
Time taken to test model(seconds)	0.01	0.01

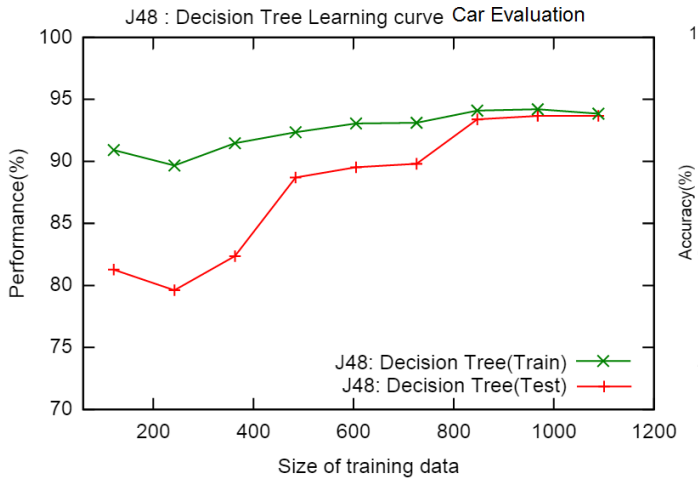
Pruning State	Confidence	# Leaves	Tree Size	Train%	Test%	Train Time	Test Time
Pruned	.125	124	247	84.45	58.33	.05	.01
Pruned	.25	144	287	86.42	57.5	.04	.01
Pruned	.5	160	319	87.66	55	.05	.01
Pruned	.75	165	329	87.84	55	.15	.01
Unpruned	---	144	287	86.41	57.5	.05	.01

Effect of Pruning: J48 pruning is post-pruning using subtree raising with a pruning confidence of 0.25. Pruning generally tends to improve performance on test set by reducing overfitting of training data as is the case with Wine dataset after confidence was decreased to .125 but in the Car Evaluation dataset that doesn't seem to be the case as unpruned tree gives better performance. I am hypothesizing that this could be attributed to high bias (under-fitting). And for the same reason, the cross-validation learning curve for training data shows poor performance of decision tree as compared to other algorithms for Car Evaluation dataset.





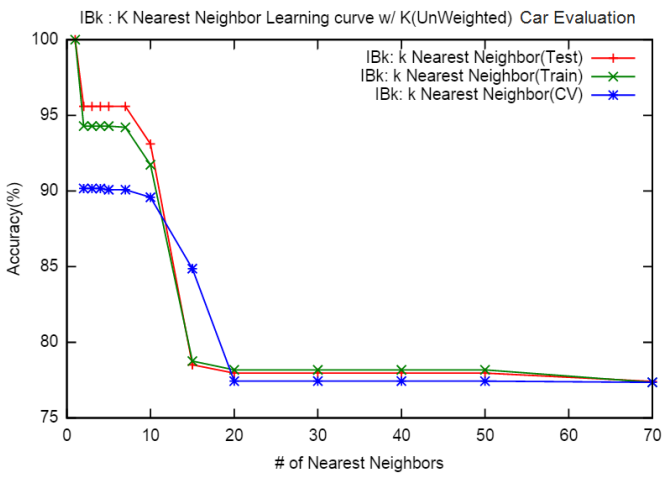
Cross-validation always helped but it was time consuming as well. For some algorithms like artificial neural networks with Epochs - 2000 or boosting with 500 iterations was very time consuming and it was not always possible to do it. The learning curves shown above for both datasets have one thing in common that accuracy definitely improves with more data.

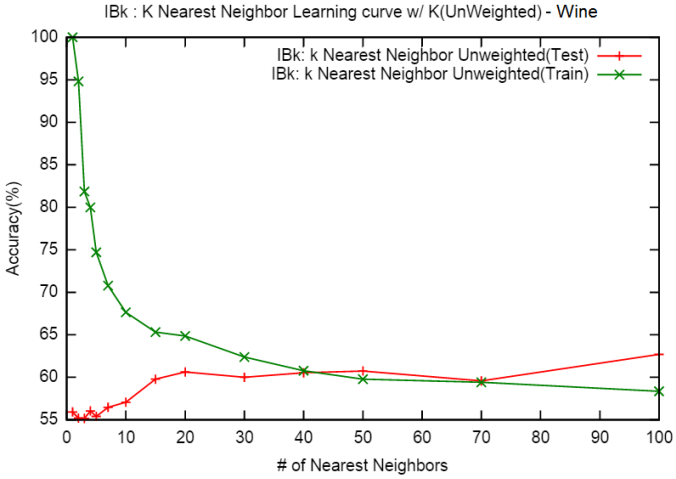


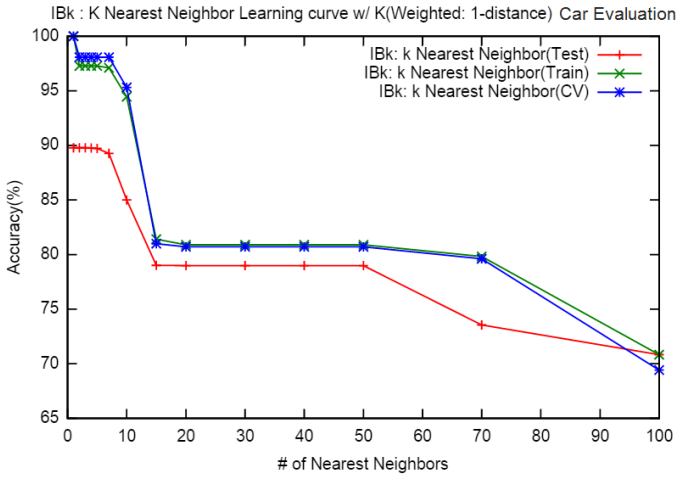
Decision tree algorithm performs better with more instances. So we can assume with more instances, this tree will have more potential branches and get more and more accurate. However for Wine dataset, training and test dataset doesn't seem to converge indicating high variance problem. It is also easy to observe that the training vs testing times follow what we would expect for an eager learner. And hence the training phase takes more time than the testing phase.

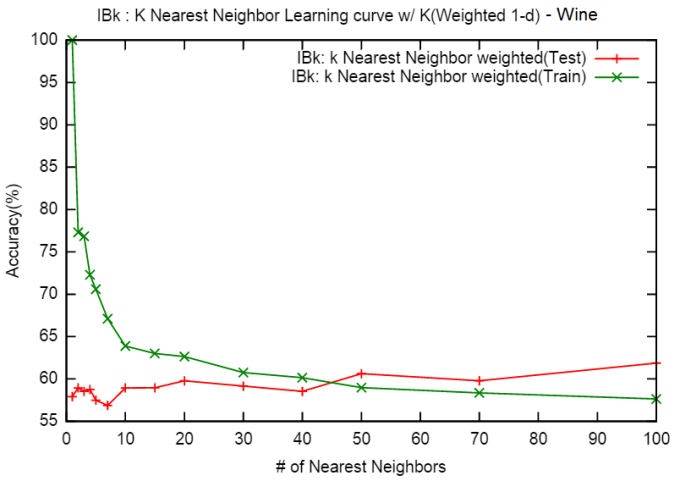
k-nearest neighbors:

I used IBk algorithm to implement K nearest neighbor.



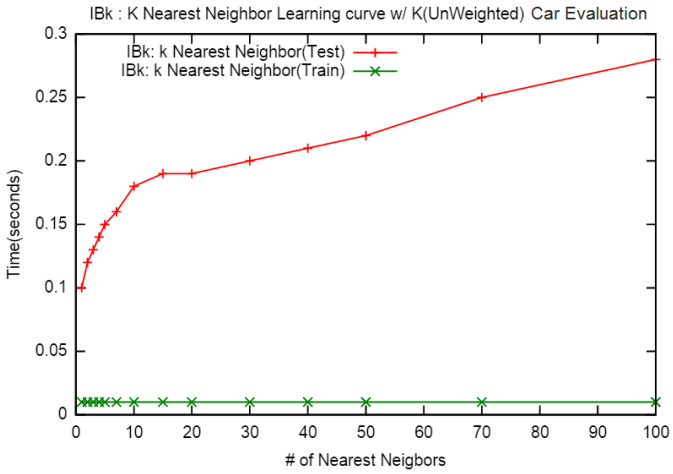


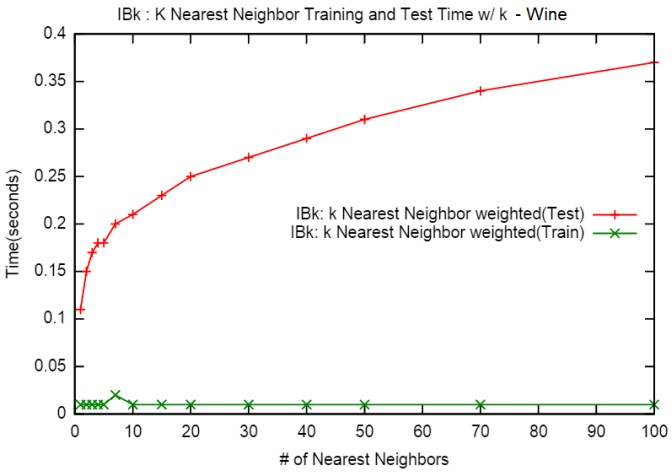


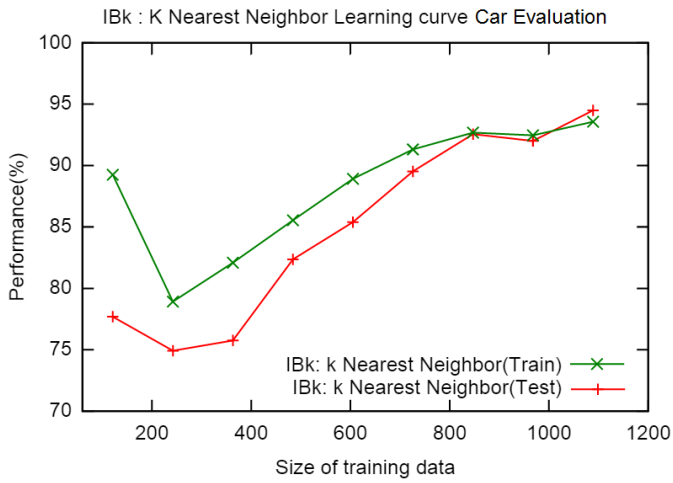


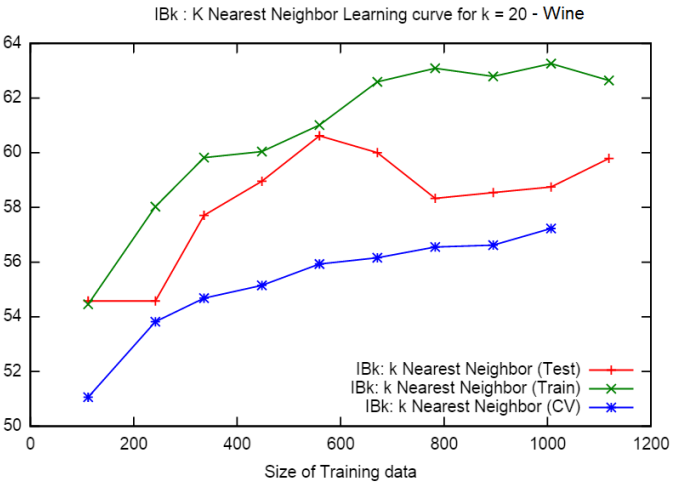
We can see that error when k = 1 is always 0 for the training sample. This is because the closest point to any training data point is itself. But ideally k should be large enough so that error rate can be minimized and small enough so that only nearby samples are included without noisy decision boundaries and with low cost. Hence I picked up k = 5 for car Evaluation dataset as performance started decreasing after k = 5 and for Wine dataset, there seems to be a bit of overfitting with training data, however with almost k =20 nearest neighbor's, it strikes the Bias-Variance Tradeoff. Biggest challenge here is to get right values of k.

Wine dataset showed better improvement when neighbors were weighed as per their distance with few neighbors which is as expected, otherwise it defeats the purpose of weighing distances.





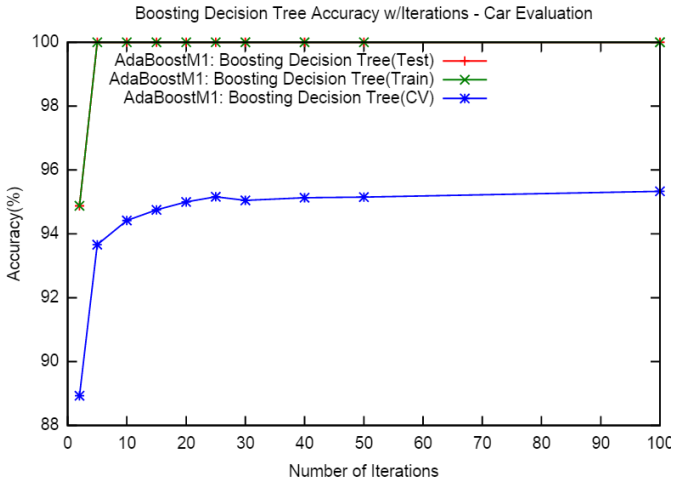


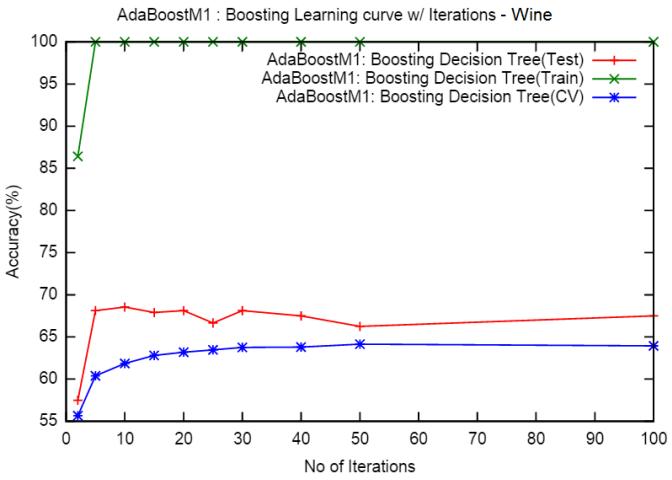


It should be noted that Knn is actually a lazy learner and hence the testing times are very high compared to training times. And Knn always performs better with more data as sparse dataset will not have a lot of neighbors to compare with.

Boosting:

AdaBoostM1 algorithm was used for performing boosting on the same decision tree – J48 that was used in the Decision Tree at the top.





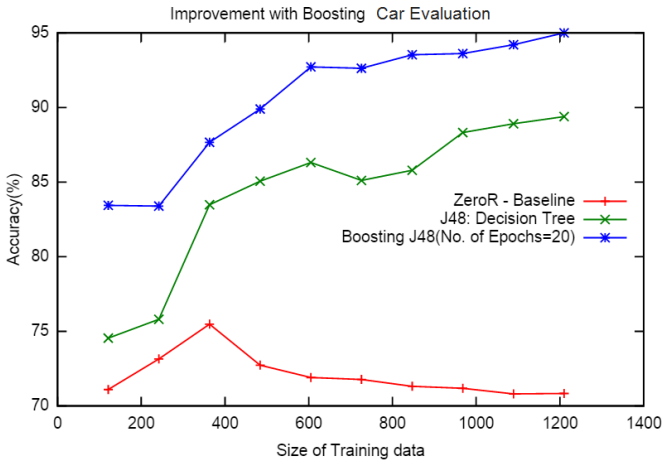
Boosting with even 20-30 iterations for Car Evaluation dataset improved the performance of the decision tree considerably (5-8%) as shown in the above graph. Given my understanding of boosting technique, this is the trend I expected. However, not much improvement was noticed with Wine dataset. Wine dataset performed best with 69.37% accuracy when confidence for pruning was set at .01 with 30 Iterations. It's the best performance on this dataset.

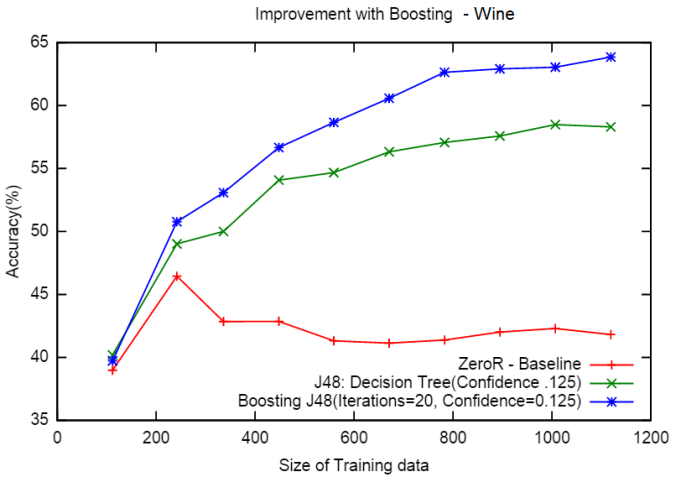
Wine Dataset

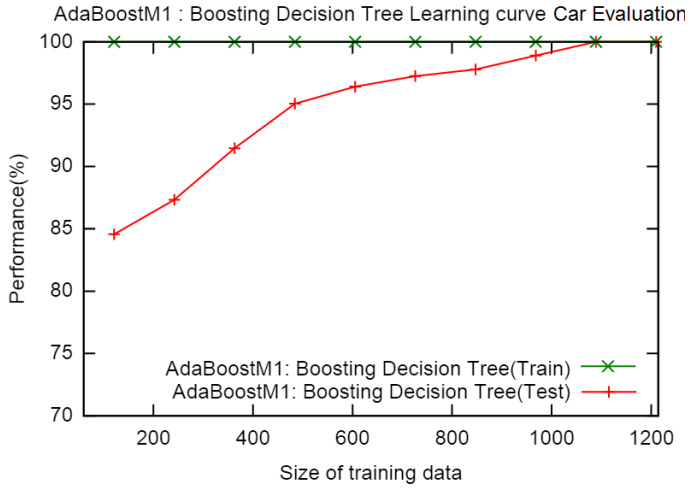
Confidence	Tree size	Train %	Test %	Training Time	Iterations
.005	209	100	69.16	11.21	200
.01	231	100	69.16	5.76	100
.01	249	100	69.37	1.23	30
.01	273	100	68	2.87	50
0.05	259	100	67.5	1.87	30
0.125	200	100	69.37	12.19	200
Unpruned	321	100	68.75	9.75	200

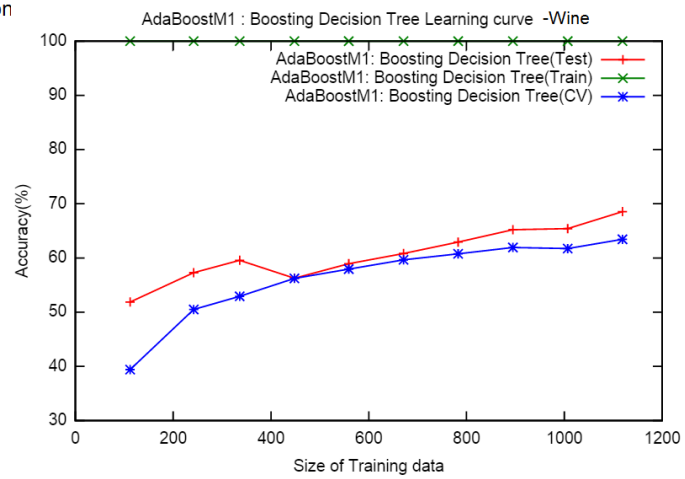
Increasing the number of iterations improved accuracy initially but saturated close to 95 % accuracy on test data for car evaluation dataset and close to 63% for wine dataset.

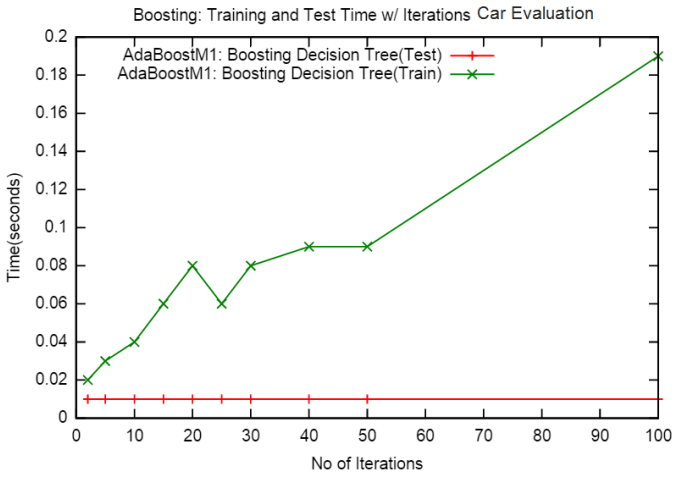
As expected, for both the datasets, boosting definitely improves the performance of decision trees. It is also easy to observe that the training time increases with more iterations and testing time remains the same which is in line with behavior of eager learner. For wine dataset, training data is over fitted which is again a high variance issue.

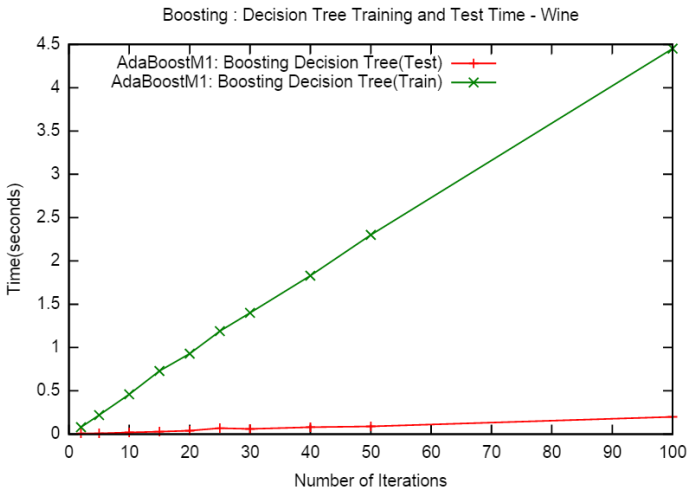










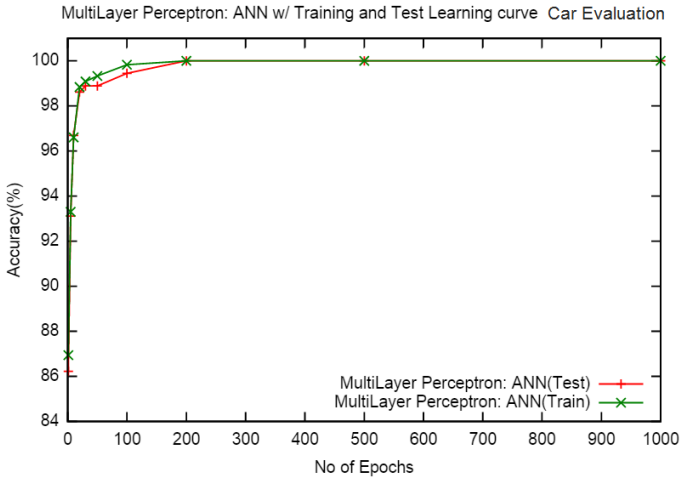


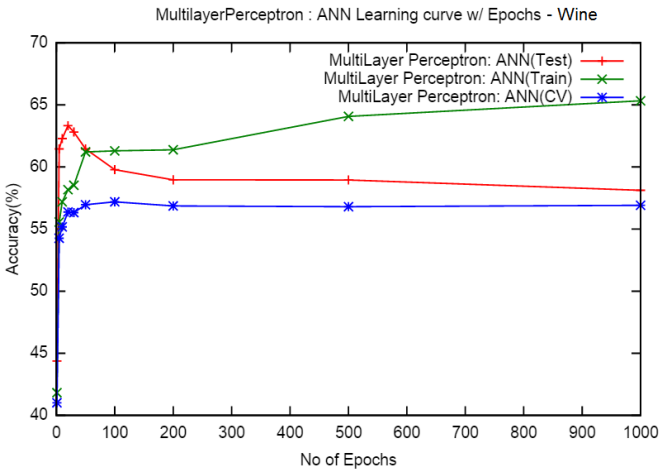
Boosting improves with more iterations but there is an added cost of increase in time, so its important to strike the balance between both the factors.

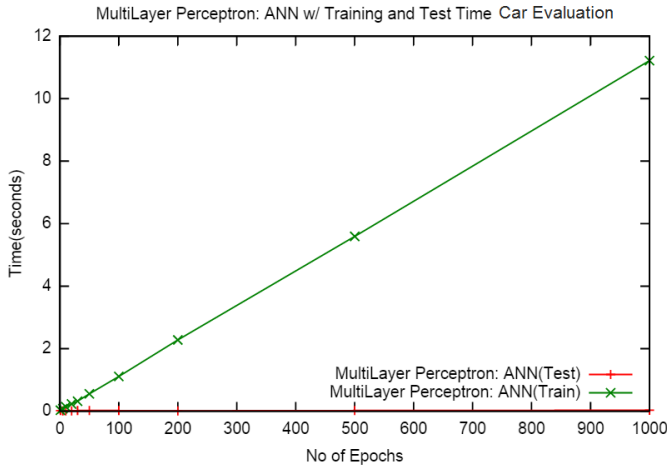
Neural networks:

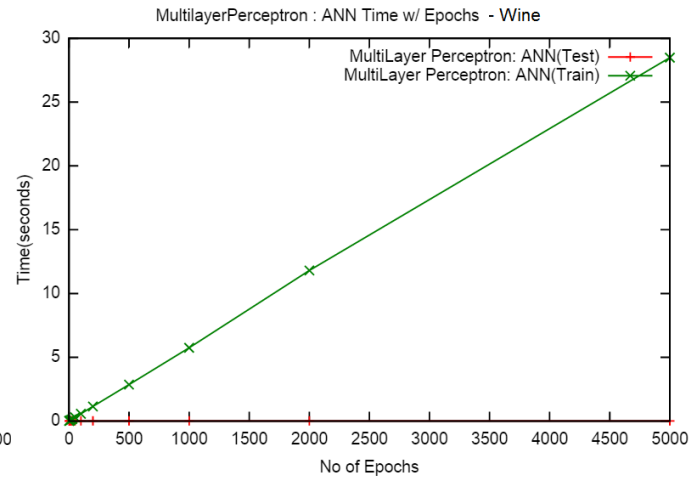
I used Multilayer Perceptron to implement Neural Networks. The experiment that I ran were quite interesting especially since Multi-Layer Perceptron performed best in terms of accuracy for car evaluation datset but worst in terms of time consumed, especially when number of Epochs were increased. Wine dataset was very resistant to neural networks and even with increased number of iterations, changes in momentum or learning rate, it didn't perform well. I hypothesize high variance as the reason.

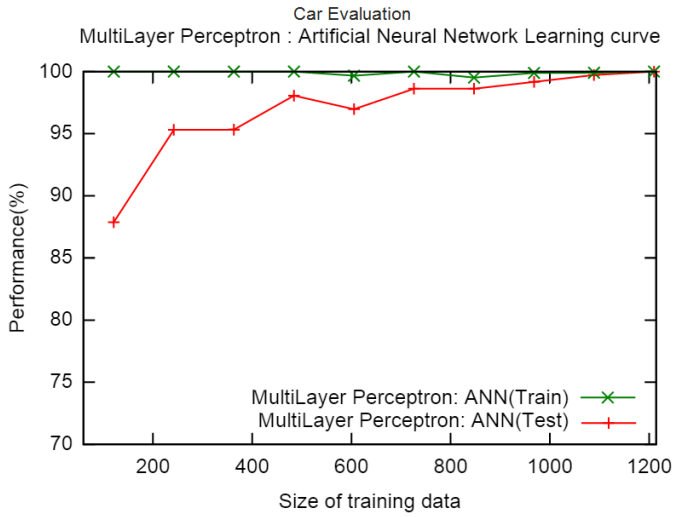
The time taken for the neural networks shows neural networks are eager learners. As seen above, time taken to test the model remained .01 seconds. As an example, car evaluation run with 5000 epochs took 54.17 seconds to train.

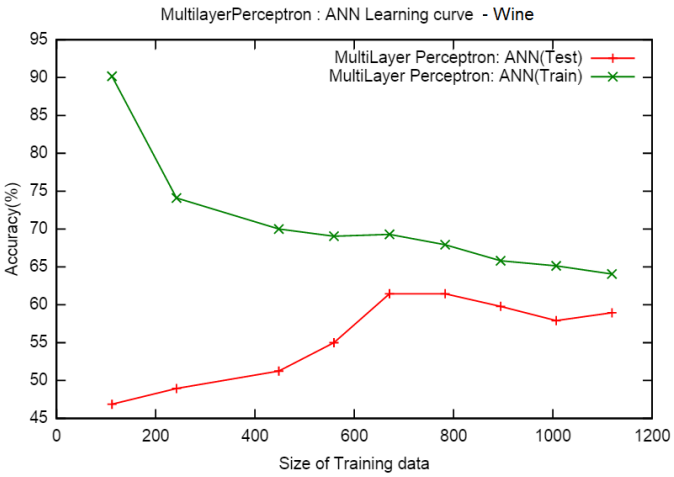












Support Vector Machines

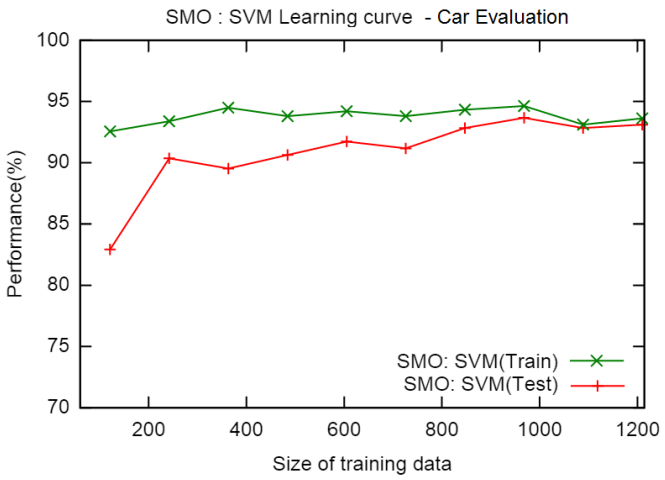
I used SMO to implement Support Vector Machines with three different kernels –PolyKernel, Puk and RBFKernel.

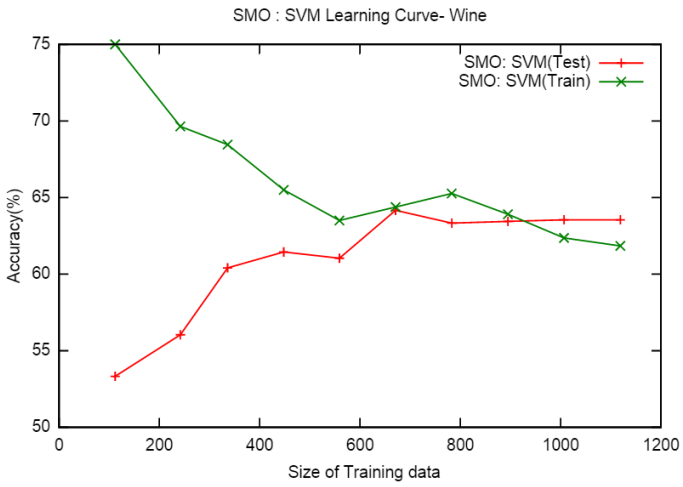
Car Evaluation DataSet

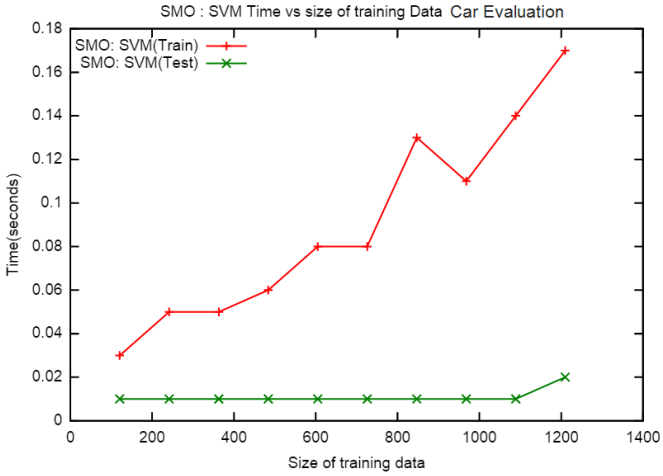
Kernel	Kernel Property	Train %	Test %	Training Time	Testing Time
PolyKernel	Exponent = 1	93.63	93.11	.19	.01
PolyKernel	Exponent = 2	99.91	99.72	.9	.24
PolyKernel	Exponent = 3	100	100	1.12	.48
RFBKernel	Gamma = .05	80.57	80.16	.89	.96
RFBKernel	Gamma = 0.1	96.69	96.69	.71	.78
RFBKernel	Gamma = 0.5	99.91	100	1.4	1.38
RFBKernel	Gamma = 1.0	100	100	2.68	3.13
Puk	Sigma 1.0 ; Omega 1.0	100	100	1.82	2.42
Puk	Sigma 1.0 ; Omega 2.0	100	100	1.29	1.64
Puk	Sigma 0.1 ; Omega 2.0	95.53	95.59	1.24	2.0

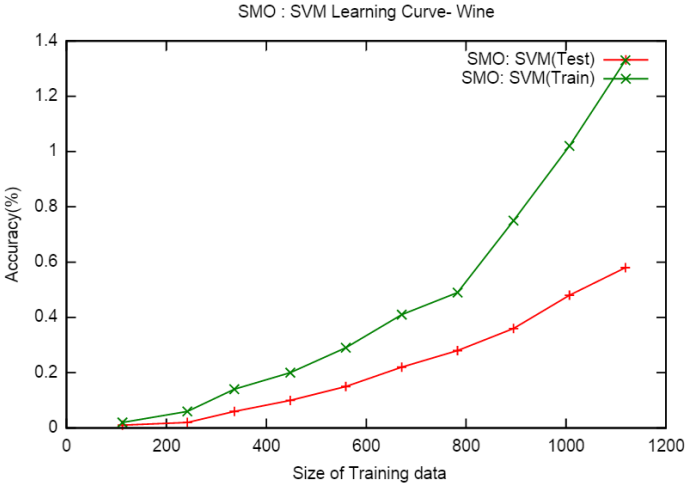
Wine DataSet

Kernel	Kernel Property	Train %	Test %	Training Time	Testing Time
PolyKernel	Exponent = 1	56.03	62.5	0.14	0.01
PolyKernel	Exponent = 2	57.19	62.29	1.04	.62
PolyKernel	Exponent = 3	60.05	62.70	1.05	.65
PolyKernel	Exponent = 4	61.84	63.54	1.21	.56
RFBKernel	Gamma = .05	53.79	61.25	.98	1.13
RFBKernel	Gamma = 0.1	54.42	61.87	0.8	1.1
RFBKernel	Gamma = 0.5	56.47	63.12	0.9	1.0
RFBKernel	Gamma = 1.0	59.16	62.91	.78	.97
Puk	Sigma 1.0 ; Omega 1.0	66.39	63.12	.92	1.19
Puk	Sigma 1.0 ; Omega 2.0	65.59	63.95	1.0	1.13
Puk	Sigma 0.5 ; Omega 2.0	61.39	65.00	.87	1.14
Puk	Sigma 0.1 ; Omega 2.0	73.45	65.41	1.15	1.89



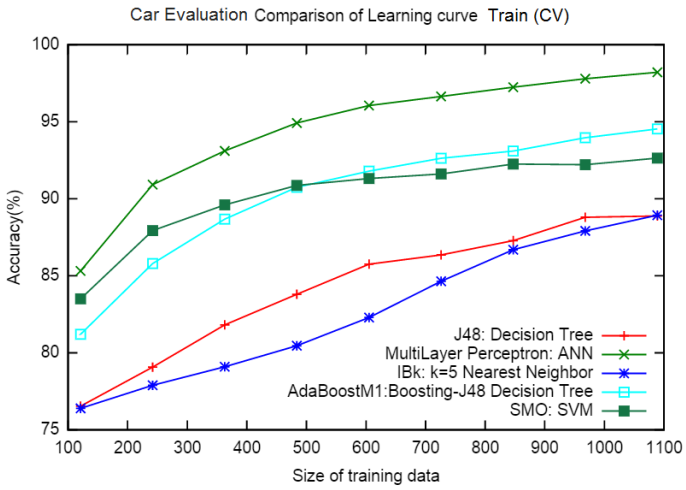


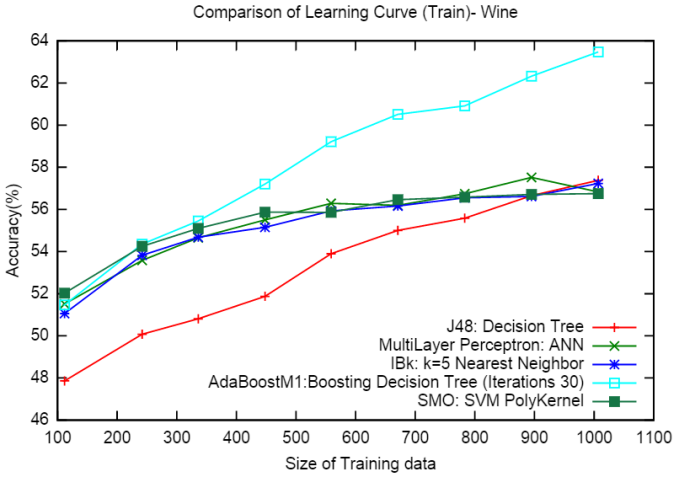




Performance of Support Vector machines improve with more data and it should be noticed that even the testing time increases a little bit with more data, so it's a mix of lazy learner and eager learner.

CONCLUSION





Cross-validation always helped but it was time consuming as well for some algorithms like artificial neural networks with Epochs - 2000 or boosting with 500 iterations and it was not always possible to do it. The learning curves shown above for both datasets have one thing in common that accuracy definitely improves with more data. However for Wine dataset, some algorithms like Support Vector Machine, Artificial Neural Networks and Nearest Neighbor showed a mere improvement of 3-4% and performed badly. My hypothesis is that there are too many attributes and dataset is small and we need to predict quality which can have as many as 10 different values. It's a bad combination of both high bias and variance.

For Wine dataset, boosting the decision tree with pruning confidence set to 0.01 performed the best(69.37%). It's best because it is efficient time wise 1.23 seconds (Iterations = 30) and tree size of not too big (249). Boosting especially reduced overfitting by using strong pruning. Second best performing algorithm was Support Vector machine with Puk which is Pearson VII function-based universal kernel but it was not as efficient time wise.

For Car Evaluation Dataset, most of the algorithms performed well and believe it is because this dataset is not plagued by curse of dimensionality due to 6 attributes. However, Artificial Neural networks performed the best with less than 50 epochs and default learning rate and momentum. They are complicated structures but hidden layers worked well for this dataset.