Min Ho Lee
#903122265
mlee432@gatech.edu

Supervised Learning Analysis

Introduction

In this assignment, the main focus is to investigate supervised learning such as Decision Tree, Neural Network, Boosting, Support Vector Machine, and k-Nearest Neighbors with datasets. The datasets used to investigate the algorithms are *spambase.arff* and *car.arff.* Weka was used as the tool to run all the algorithms, and all the data and graphs for model complexity and learning curves collected from it are stored in excel file "*Data*".

Datasets Description

spambase:

This dataset contains 4601 (1813 spam, 2788 non-spam) instances and 58 (57 continuous, 1 nominal class label) class attributes. Each instance denotes whether the e-mail was considered spam (1) or not (0) as its class, and most of the class attributes indicate whether a certain word or character was frequently occurring in the e-mail. This collection of spam e-mails came from their postmaster and who had filed spam, and Our collection of non-spam e-mails came from filed work and personal e-mails.

car:

This dataset is car evaluation dataset that is complete dataset containing 1728 (1210 *unacc*, 384 *acc*, 69 *good*, 65 *v-good*) instances and 7 attributes. The attributes denote buying price, price of maintenance, number of doors, capacity in terms of persons to carry, the size of luggage boot and estimated safety of the car, and the car can be evaluated from the result of the attributes.
The model evaluates cars according to the following concept structure:

CAR car acceptability
- PRICE overall price
  - buying buying price
  - maint price of the maintenance
- TECH technical characteristics

- COMFORT comfort
  - doors number of doors
  - persons capacity in terms of persons to carry
  - lug_boot the size of luggage boot
  - safety estimated safety of the car

Interesting about this Datasets:

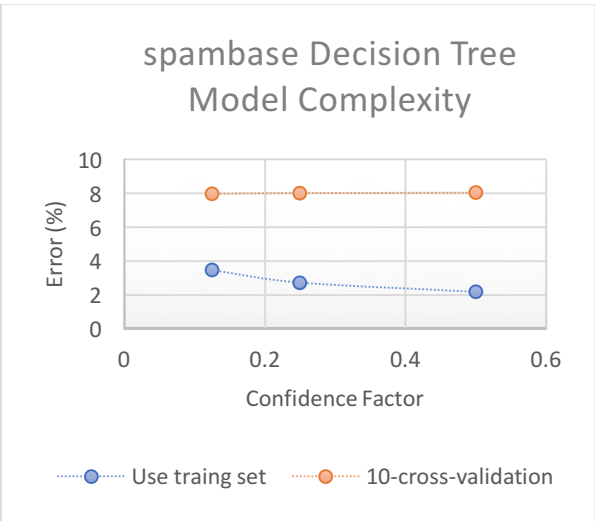
Both datasets are interesting because these datasets can be practically being used to filter spam e-mails and to evaluate car values. Nowadays, computer algorithms are taking over jobs what human-beings used to do. With more data, we can get better result in performance.

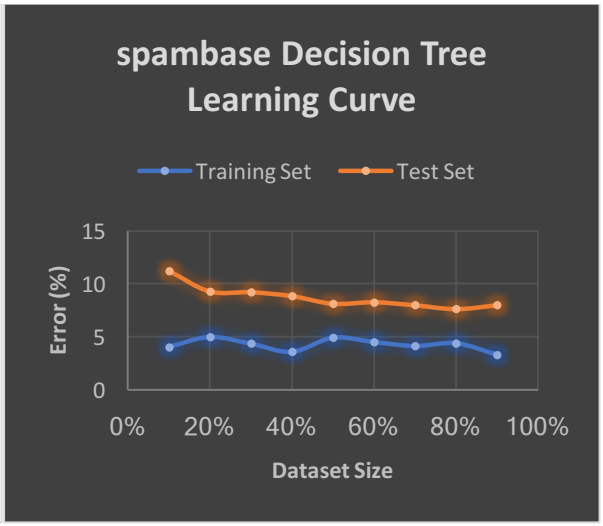
For the *spambase* dataset, filtering out spam e-mail is a common interest for all email system. Since it is the most common machine learning factor we are facing daily, it is interesting to know what algorithm makes more mistakes and which one is going to perform better than the others.

For the car evaluation dataset, this kind of data has been used in the actual field nowadays. However, sometimes it disappoints people who use this kind of system because sometimes it makes mistakes (errors). Therefore, it is interesting to see in which case it makes mistakes and which algorithm will perform the best.

Decision Tree:

spambase (58 attributes, Number of Instances: (Training: 3220, Testing: 1381))								
				Use training set		10-Cross-Validation		
Pruning?	Confidence Factor	Number of Leaves	Size of the tree	Error(%)	Build Time(s)	Error(%)	Build Time(s)	Testing Error(%)
No	-	145	289	1.9559	0.18	8.1031	0.18	
Yes	0.125	65	129	3.4772	0.2	7.9478	0.18	7.029
Yes	0.25	84	167	2.7321	0.18	8.0099	0.2	
Yes	0.5	114	227	2.1732	0.21	8.041	0.23	

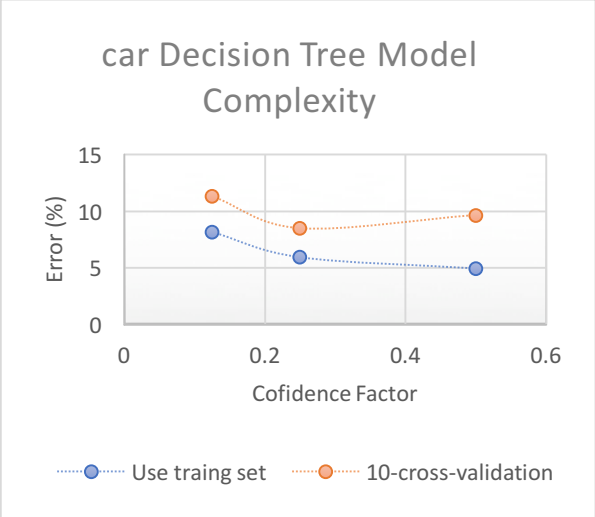


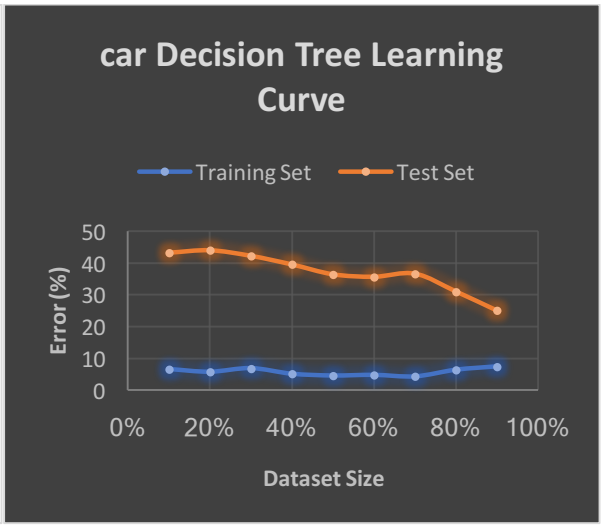


As we can see from the graph for the dataset *spambase* above, the error rate goes up as confidence factor goes up with 10-crosss validation for *spambase* dataset. The error from 10 cross validation is shown with the lowest confidence level, 0.125. Low confidence level represents it does more pruning, which results smaller number of leaves and size of the tree.

With pruning, a tree is removing specific branches or stems to benefit the whole tree, therefore it decreases not only number of leaves and size of trees, but also it reduces the error rate. Also, as we can see from the learning curve, the error for the training set is surprisingly low even with the small data set size. So, overall, the result we can conclude from the graph is that the dataset *spambase* contains a lot of irrelevant attributes.

car **(7 attributes, Number of Instances:(Training: 1210 Testing: 518))**								
				Use training set		10-Cross-Validation		
Pruning?	Confidence Factor	Number of Leaves	Size of the tree	Error(%)	Build Time(s)	Error(%)	Build Time(s)	Testing Error(%)
No	-	143	194	3.9669	0	9.2562	0	
Yes	0.125	66	91	8.1818	0	11.3223	0	
Yes	0.25	89	122	5.9504	0	8.5124	0	23.3591
Yes	0.5	107	146	4.9587	0	9.6694	0	





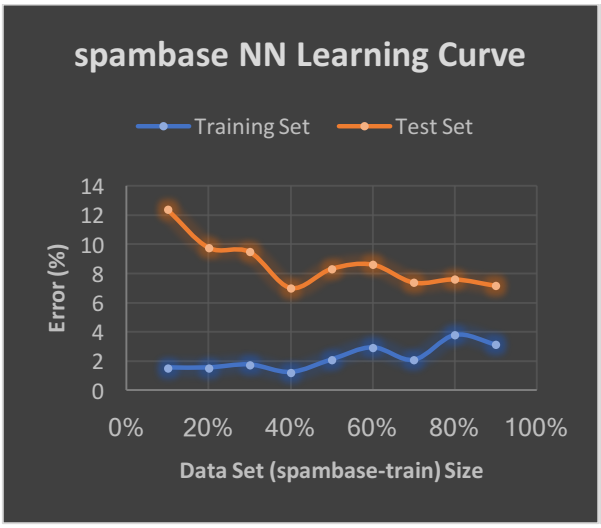
As shown in the model complexity graphs for the *car* dataset, the error rate decreases as confidence factor grows and suddenly increases. This is merely a noise caused by small size of your training and test sets, as well as the random nature of the random forest model. As we can see from the learning curve, the error for the test set is gradually decreasing as the size of the dataset increases. We can infer that as the size of the dataset increases, the decision tree makes better decision.

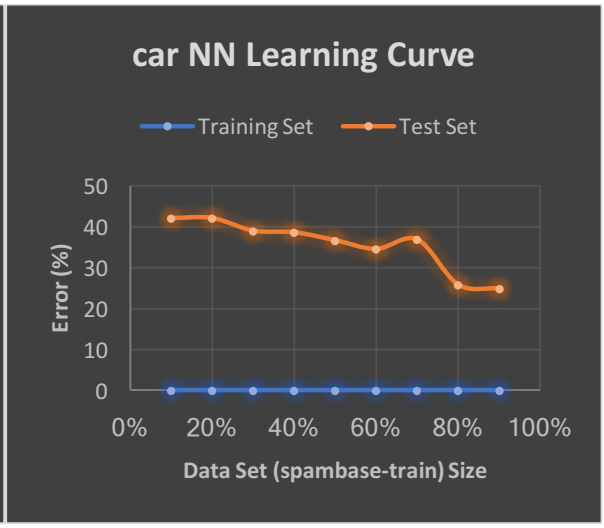
Neural Network:

For Neural Network, different learning rate and momentum have been used to see how it effects on the performance of the result.

spambase (58 attributes, Number of Instances: (Training: 3220, Testing: 1381))								
	L = 0.3, M = 0.2				L = 0.3, M = 0.4			
Number of Epochs	Error(%)	Build Time(s)	Test Error(%)	Build Time(s)	Error(%)	Build Time(s)	Test Error(%)	Build Time(s)
500	9.5933	41.71	6.8116	21.37	15.9267	47.3	16.8116	23.67
1000	13.3188	83.11	5.942	37.5	9.407	114.26	6.3768	48.91
3000	8.1031	389.52	4.7826	169.19	8.5377	357.18	4.9275	172.57
10000	5.8367	1113.12	4.2029	617	6.6439	1398.71	4.8551	515.98
	L = 0.1, M = 0.2				L = 0.1, M = 0.4			
Number of Epochs	Error(%)	Build Time(s)	Test Error(%)	Build Time(s)	Error(%)	Build Time(s)	Test Error(%)	Build Time(s)
500	9.7175	44.94	5.5072	18.36	9.2828	42.61	5.7246	19.16
1000	9.4691	120.85	4.2754	36.66	14.1881	87.33	4.4928	39.16
3000	4.9364	425.87	3.1159	124.22	5.9919	271.65	2.3188	126.14
10000	4.8432	1285.03	1.5942	479.93	5.0916	1179.23	2.029	535.08

car (7 attributes, Number of Instances:(Training: 1210 Testing: 518))								
	L = 0.3, M = 0.2				L = 0.3, M = 0.4			
Number of Epochs	Error(%)	Build Time(s)	Test Error(%)	Build Time(s)	Error(%)	Build Time(s)	Test Error(%)	Build Time(s)
500	0.9091	3.51	0	1.53	0.6612	3.57	0	2.16
1000	0.8264	7.05	0	3.26	0.7438	7.89	0	4.83
3000	0.7438	25.86	0	10.39	0.6612	23.08	0	13.22
10000	0.7438	79.36	0	32.27	0.6612	81.01	0	33.13
	L = 0.1, M = 0.2				L = 0.1, M = 0.4			
Number of Epochs	Error(%)	Build Time(s)	Test Error(%)	Build Time(s)	Error(%)	Build Time(s)	Test Error(%)	Build Time(s)
500	1.0744	3.82	0	2.17	0.9091	3.69	0	2.24
1000	1.157	7.92	0	3.28	0.9091	7.59	0	4.56
3000	1.2397	30.43	0	10.09	0.8264	29.77	0	11.09
10000	1.2397	73.91	0	44.3	0.8264	90.14	0	47.45





Neural Network is known as flexible in describing any function, so it is easy to get overfitting as well. Overfitting commonly happens in Neural network and it is a major problem. As we can see in (L = 0.3, M = 0.2), (L = 0.3, M = 0.4), and (L = 0.1, M = 0.4) on the *spambase* table shown above, the error rate when epoch is 500 is higher when the epoch is 1,000.

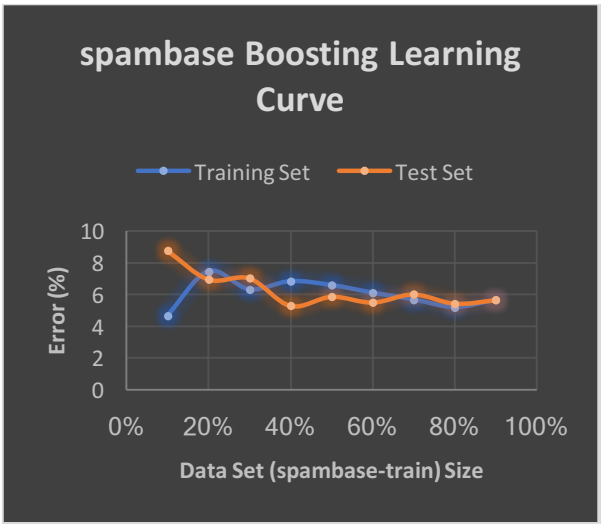
Also, we can see the learning rate has significant effects on the result. As we can see from the table above, we can observe that the error rate is lower when the learning rate is lower. However, the lowering the learning rate can cause overfitting when the number of training epochs is not large enough for the smaller learning rate which could result in the system failing to reach the optimal solution.

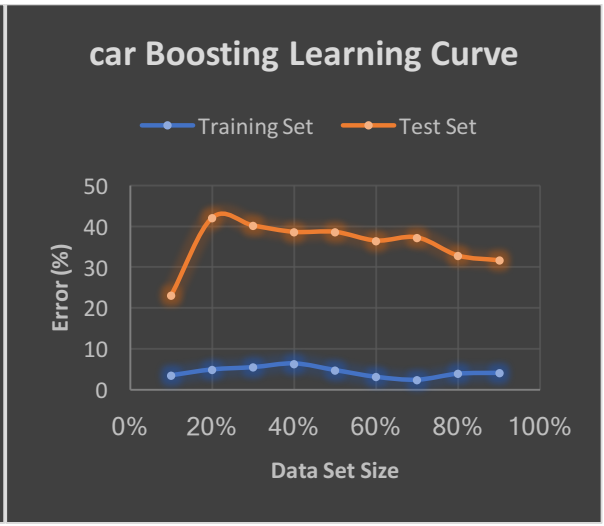
We also can observe that the momentum has significant effects on the result. Comparing (L = 0.1, M = 0.2) and (L = 0.1, M = 0.4) tables above, we can see the test error differs almost three times. Since momentum is used to help the network out of local minima, high momentum will result in overshooting the absolute minimum, and low momentum will result in converging at local minimum which will result in high error rates.

Also, as we can see from the *car* learning curve, the training set error rate is extremely low while test error is high with a small size of dataset. This could happen when there are overfitting on the training samples when the dataset is small with high variance. As the dataset size increases, the classifier starts to generalize better, thus raising the success rate in the test dataset.

Boosting(Using J-48):

spambase (58 attributes, Number of Instances: (Training: 3220, Testing: 1381))								
Number of Instances: (Training: 3220, Testing: 1381)				Using training set		10-Cross-Validation		
Pruning?	Confidence Factor	Num Interation	Performed Iteration	Error(%)	Build Time(s)	Error(%)	Build Time(s)	Testing Error(%)
No	-	10	10	0.031	2.34	5.8057	2.25	
No	-	20	18	0.031	2.74	5.5573	3.17	
No	-	30	18	0.031	2.75	5.4952	3.09	
Yes	0.125	10	10	0.031	3.42	5.2779	3.23	
Yes	0.125	20	20	0.031	3.25	5.1537	4.67	
Yes	0.125	30	24	0.031	3.53	5.0295	4.04	
Yes	0.25	10	10	0.031	2.79	4.7811	2.85	4.6377
Yes	0.25	20	20	0.031	3.22	4.7811	4.91	
Yes	0.25	30	30	0.031	3.66	4.7811	3.82	
Yes	0.5	10	10	0.031	2.91	5.34	2.91	
Yes	0.5	20	16	0.031	4.07	5.34	4.07	
Yes	0.5	30	16	0.031	4.09	5.34	4.09	



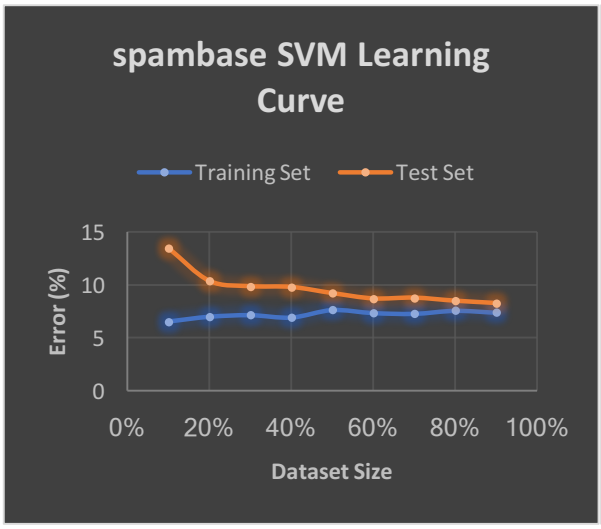


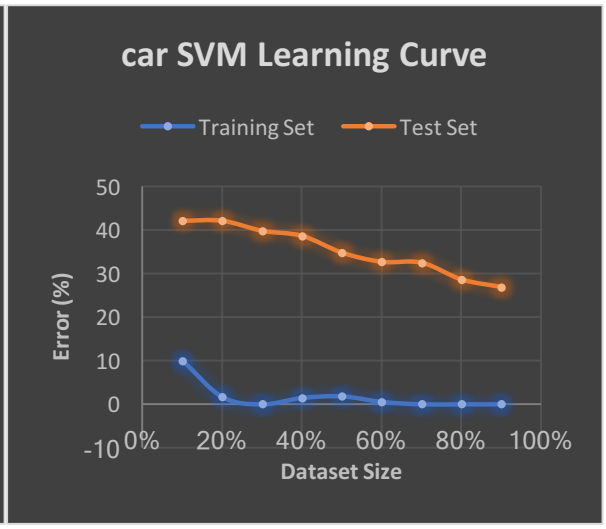
As we can see from the both learning curves, we can see the error of training set in *spambase* and the error of test set in *car* dramatically increase from 0% – 20%. It is because a noise caused by small size of the training and test set, as well as the random nature of the random forest model. When there are small samples it is easy to fit a model that has good accuracy for the training data, however, that fitted model is not going to generalize well for test data. When dataset size is increased, in it is going to be harder to fit the training data, but the results generalize better for the test data. And from the *car* dataset, we can observe overfitting since a model learning the detail and noise in the training data to the extent that it is negatively impacting the performance of the model on new data.

Also, another thing we can obtain from the table above, we can see the actual performed iteration are sometimes different as given number of iteration. This happens because of *AdaBoost* algorithm which ends iteration before it reaches to the end when the absolute value of [0.5 – weight error rate] is less than threshold. We can make it iterate more by lowering the threshold. This would make the error lower, but it will make run-time higher.

Support Vector Machine:

For Support Vector Machine, a polynomial kernel and Radial Basis Function were used.





Support Vector Machine						
spambase **(58 attributes, Number of Instances: (Training: 3220, Testing: 1381))**						
		Using training set		10-Cross-Validation		
Kernel	Exponent/Gamma	Error(%)	Build Time(s)	Error(%)	Build Time(s)	Testing Error(%)
Polynomial	1	10.1832	0.09	10.6799	0.39s	
Polynomial	2	18.131	13.88	19.0314	9.96s	
Polynomial	3	30.0217	16.36	30.6737	17.28	
Polynomial	4	36.1378	11.41	36.1689	10.15	
RBF	0.01	29.1524	34.81	31.2325	39.91	
RBF	0.1	13.1326	9.43	13.412	8.4	
RBF	0.5	8.724	7.52	9.407	7.26	
RBF	1	7.4511	5.41	8.4756	5.03	
car (7 attributes, Number of Instances:(Training: 1210 Testing: 518))						
		Using training set		10-Cross-Validation		
Kernel	Exponent/Gamma	Error(%)	Build Time(s)	Error(%)	Build Time(s)	Testing Error(%)
Polynomial	1	5.6198	0.08	5.7851	0.12	
Polynomial	2	0.0826	0.43	0.7438	0.44	
Polynomial	3	0	1.04	0	0.93	
Polynomial	4	0	1.54	0.0826	1.6	
RBF	0.01	18.843	0.55	18.84	0.51	
RBF	0.1	4.9587	0.42	8.0165	0.41	
RBF	0.5	0	1.13	2.2314	1.17	
RBF	1	0	1.67	9.0909	1.63	

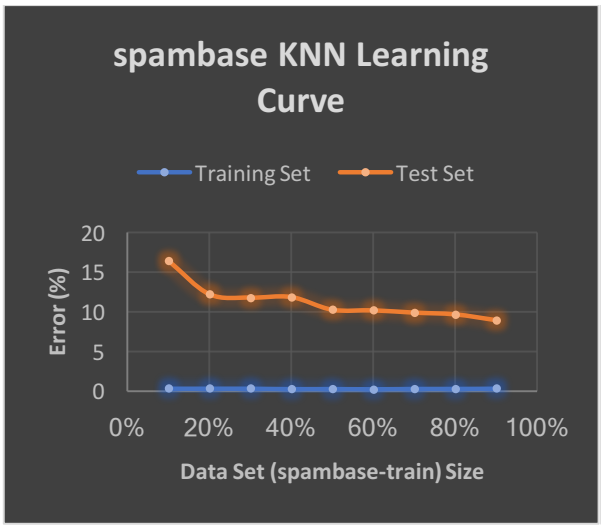
Support Vector Machine is known as strong resistance to overfitting. It is true in most cases, however we can observe overfitting as we can see from the table for the *car* dataset above. As it is shown, the error is higher when the polynomial 4 than when the polynomial is 3. We can infer from the result that when the kernel does not match the function described in the dataset, overfitting can happen.

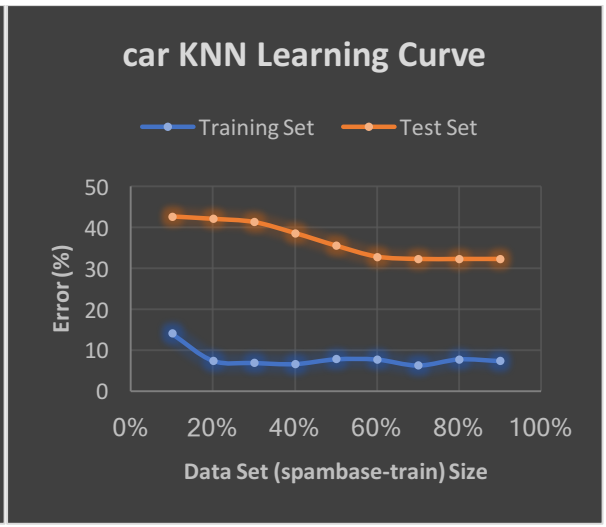
Also, as we can see from the *car* learning curve, the training set error rate is extremely low while test error is high with a small size of dataset. This could happen when there are overfitting on

the training samples when the dataset is small with high variance. As the dataset size increases, the classifier starts to generalize better, thus raising the success rate in the test dataset.

k-Nearest Neighbors:

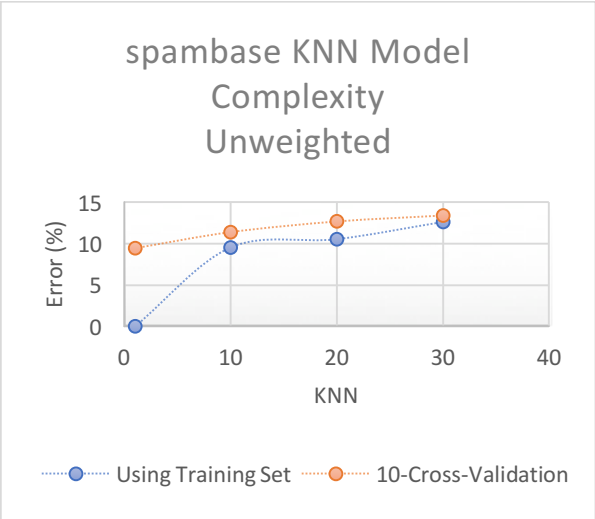
k-nearest neighbors						
spambase **(58 attributes, Number of Instances: (Training: 3220, Testing: 1381))**						
		Using training set		10-Cross-Validation		
Distance Weight	KNN	Error(%)	Build Time(s)	Error(%)	Build Time(s)	Testing Error(%)
Unweighted	1	0.031	0	9.4691	0.01	
Weighted	1	0.031	0	9.4691	0	
Unweighted	10	9.5312	0	11.394	0	
Weighted	10	0.2794	0	9.0345	0	
Unweighted	20	10.5247	0	12.6669	0	
Weighted	20	0.6209	0	9.2828	0	
Unweighted	30	12.6048	0	13.3809	0	
Weighted	30	1.2419	0	10.1521	0	
car (7 attributes, Number of Instances:(Training: 1210 Testing: 518))						
		Using training set		10-Cross-Validation		
Distance Weight	KNN	Error(%)	Build Time(s)	Error(%)	Build Time(s)	Testing Error(%)
Unweighted	1	0	0	7.6033	0	
Weighted	1	0	0	7.6033	0	
Unweighted	10	3.8843	0	7.7686	0	
Weighted	10	0	0	7.8512	0	
Unweighted	20	16.1157	0	17.1901	0	
Weighted	20	0	0	15.7025	0	
Unweighted	30	16.1157	0	17.1901	0	
Weighted	30	0	0	15.7025	0	

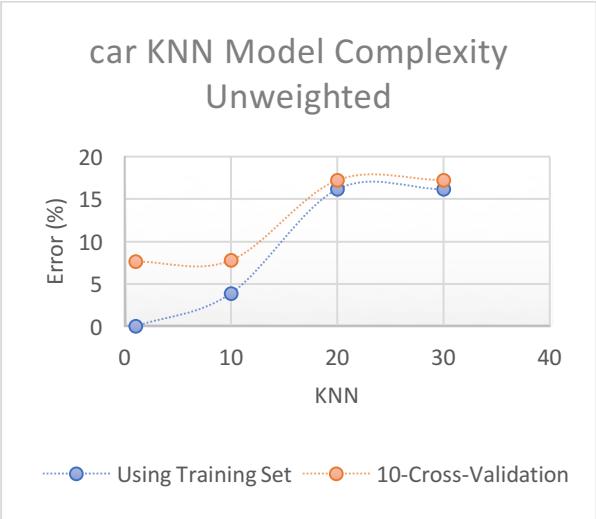




K-Nearest Neighbor is known as strong dealing with noise, and it is working for the *spambase* dataset as we can see from the learning curve above. As we observed in Decision Tree, the *spambase* dataset has a lot of irrelevant attributes that does not give a huge impact on the result. When we compare the learning curve from Decision Tree and the learning curve from KNN, we can see the error rate on the KNN is extremely lower than the learning curve from Decision Tree.

Also, as we can see from the learning curve above, both datasets show high variance, which means that the model hasn't figured out a representation of the data that also fits new data. In order to fix this problem, we can increase size of the datasets or give smaller set of features.





From the model complexity graphs for the both datasets above, we can see the training and testing error increase as the number of neighbors increases. Smoothing is typically a desirable property for generalization. While it technically depends on the characteristics of the dataset, increasing k should reduce over-fitting but once k is too large the smoothing effect you intuited results in decreased variance which will affect overall performance negatively. The increase of the errors can be lowered by using distance weighing as we can see the dramatic decrease of the errors from the table above.