Eugene Oh
Assignment #1

Supervised Learning Report

Datasets

Census Income

This is a set of data taken from the census bureau database and was extracted in 1994. The task is to predict whether the person earns over $50K per year. Attributions for the dataset can be found in the adult.txt file.
The data is not complete. After removing rows with incomplete data we see there are 32,561 samples to use and each with 14 attributes. Additionally, there are 24,720 samples that are labeled as making less than or equal to 50K per year and 7,841 samples labeled as making greater than 50K per year. After transforming the text based features into binary columns we actually end up creating 107 features. To illustrate this transformation, for example the country feature is a string of the country that person is originally from. Transformed this becomes several features of the form “country_${COUNTRY_NAME}” where the value is either a 1 or 0 based on whether or not that person is from that country. One last thing to note is that 75.9% of the data is labeled as a person making less than $50K per year.

Wisconsin Diagnostic Breast Cancer

This dataset was originally used to help create models that could diagnose breast cancer based on measurements of breast masses based solely on Fine Needle Aspiration. The mean, standard error, and worst (mean of the worst 3) values of these measurements are calculated for this dataset. It is a complete dataset that includes 569 samples as well as 30 attributes (10 measurements and the 3 calculated values for each measurement mentioned above). The task is to take these attributes and diagnose whether or not the breast mass is malignant or benign.

Why are these interesting?

In a practical sense, the census income dataset is interesting because when supervised learning techniques are applied to it, the output could potentially give insight into the conditions that allow for relative financial stability within the United States. Additionally, before the techniques are even applied it’s interesting to see that a majority of people in the dataset earn less than $50K per year.
The diagnostic breast cancer data is interesting because the application of supervised learning techniques can lead to improved cancer detection. Early detection in cancer is considered to be one of the key factors in the success of treatment.

For each of the experiments, I set aside 30% of the total set to be my test set. The other 70% was used for training and cross validation.

Decision Trees

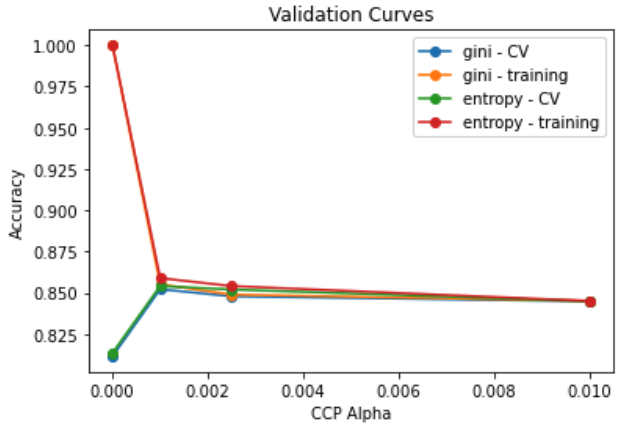
Census Income
Unpruned

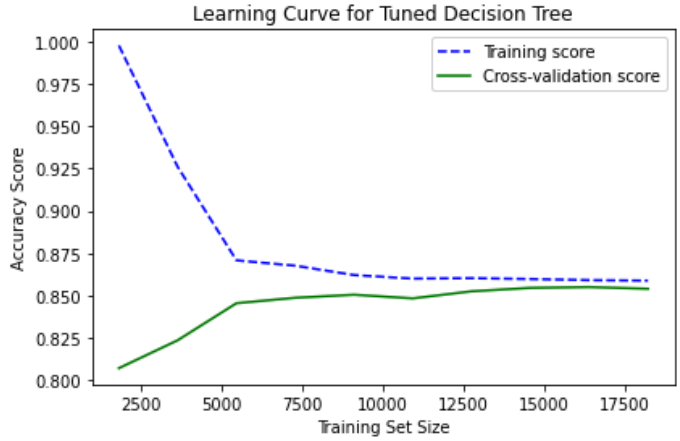
criterion	mean_fit_time	mean_score_time	mean_test_score	mean_train_score
gini	0.287317	0.006868	0.812565	1.0

The depth of this tree is 45, with 3224 leaves and 6447 total nodes within the tree.

Pruned
After running a grid search to try out some hyperparameters we see the following results:

ccp_alpha	criterion	mean_fit_time	mean_score_time	mean_test_score	mean_train_score
0.0010	**entropy**	0.432138	0.009034	0.854116	0.858909
	gini	0.441761	0.009544	0.852185	0.855168
0.0025	**entropy**	0.444617	0.009309	0.852009	0.854115
	gini	0.450227	0.009166	0.847885	0.848927
0.0100	**gini**	0.459293	0.009574	0.844902	0.845077



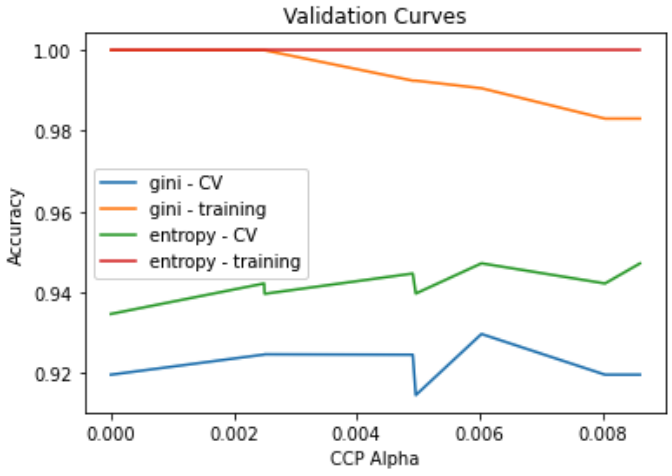


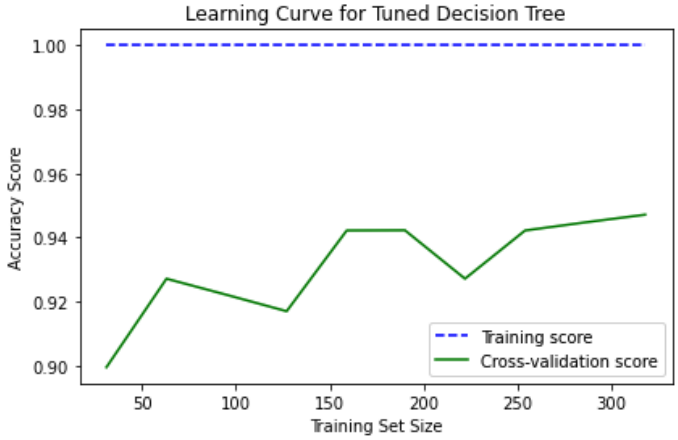
Wisconsin Diagnostic Breast Cancer
Unpruned

criterion	mean_fit_time	mean_score_time	mean_test_score	mean_train_score
gini	0.0072	0.0022	0.922089	1.0

Pruned

ccp_alpha	criterion	mean_fit_time	mean_score_time	mean_test_score	mean_train_score
0.000000	**gini**	0.010363	0.003448	0.919525	1.000000
	entropy	0.010050	0.003372	0.934589	1.000000
0.004961	**gini**	0.009985	0.003238	0.914462	0.992461
	entropy	0.010435	0.003260	0.939652	1.000000
0.006030	**entropy**	0.010171	0.003198	0.947152	1.000000
	gini	0.010129	0.003516	0.929620	0.990576





There are a couple of things to note here. First, any pruning of the tree improved accuracy on the test set. In the case of the census income dataset, the pruning was quite aggressive. The unpruned tree began with a max depth of 45, 3224 leaves, and 6447 total nodes. The pruned tree with the cost complexity parameter that gave the best result from our grid search resulted in a tree of max depth 10, 25 leaves, and 49 total nodes. With this it's very clear the tree was overfitting on the training set. This makes sense in that there are 107 features for this data set and with 32561 samples so the decision tree could be prone to overfitting. Thus pruning will be able to undo some of the overfitting.

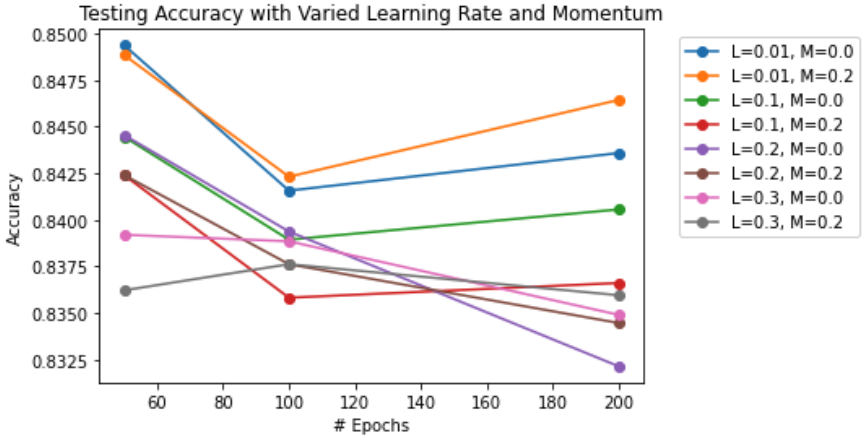
One other piece of data highlighting the overfitting is in the validation curves. The training and validation scores converge as the cost complexity parameter is increased. This means the tree was able to generalize better as it was pruned more aggressively.

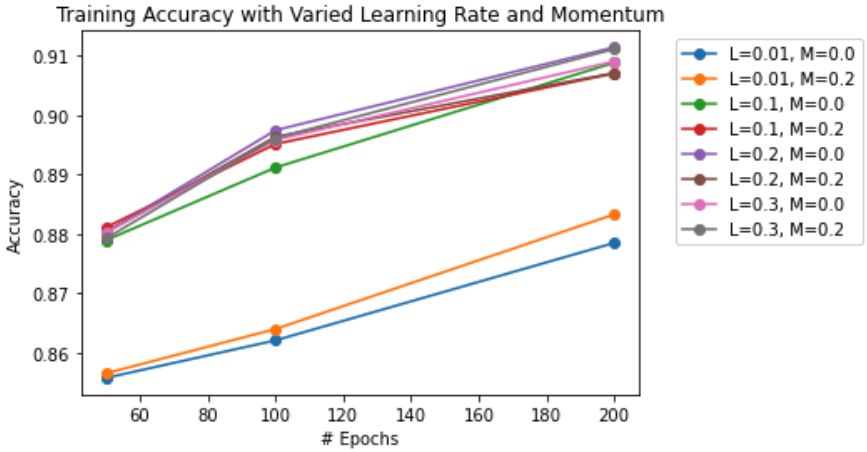
For the breast cancer dataset, it was interesting to see that the tree with no pruning only led to a max depth of 6. This is probably a result of this dataset having fewer features and way fewer samples. This is drastically different from that of the census income dataset. Additionally the tree performed much better with no pruning than with the census income dataset which makes sense as the less complex a tree is, the less likely it is to overfit the training data.

Another thing to note is the training and testing times of the tree. As an eager learner, it's expected for the training phase to take significantly more time and that is illustrated above.

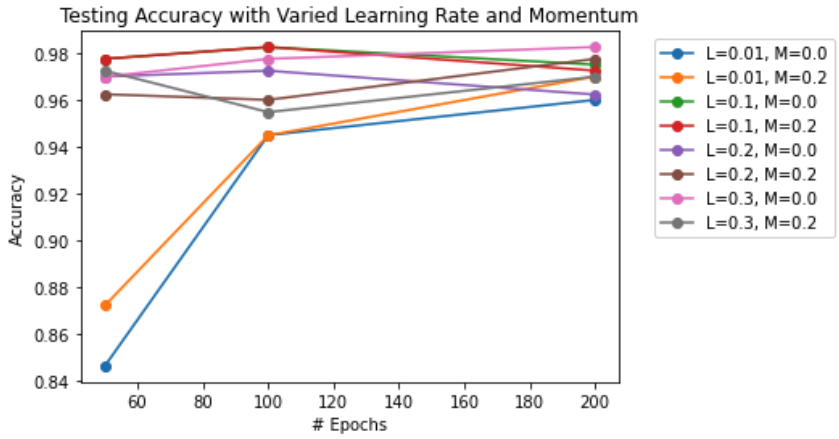
Neural Networks

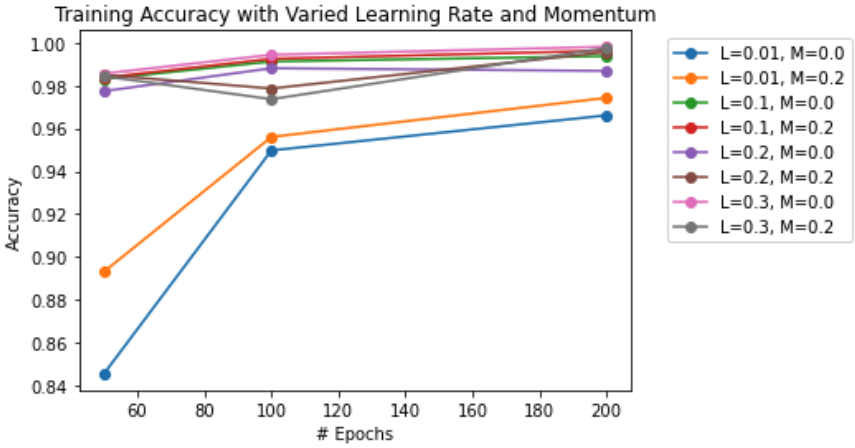
Census Income





Breast Cancer

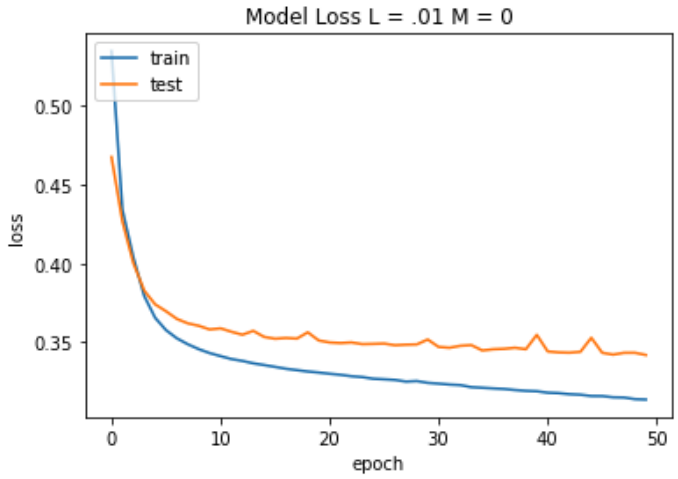




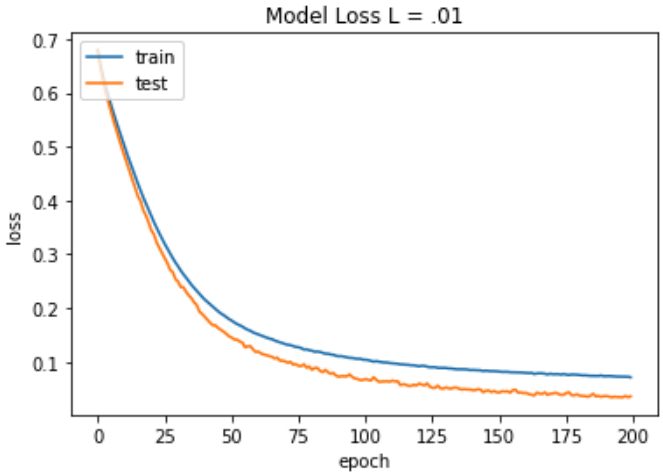
With neural networks the main parameters I tried varying were the learning rate and momentum. After performing a grid search to run the several combinations possible from the different values, the results were not varied much. All of the resulting testing accuracies are within a few percent of each other. One thing to note however is that the accuracies at the lowest number of epochs were much lower for the models with the smallest learning rates which makes sense as the weights are adjusted more slowly. Another main thing to note is that neural networks are absolutely eager learners as the training phase for all iterations took much longer than that of any other learning algorithm explored in this report. The mean training time for even the least number of epochs was significantly larger than that of any other model.

The plots below show the loss as the number of epochs increases:

Census Income ANN with Learning Rate 0.1 Momentum 0



Breast Cancer ANN with Learning Rate 0.01



One interesting thing to note here is that with the breast cancer dataset, the test loss actually becomes smaller than the training loss as the number of epochs increases. I'm not quite sure why this behavior occurs.

At around the 7th epoch mark on the census income dataset we can already see the overfitting taking place. This is probably a result of there being so many features due to the transformation that we performed on the census income set. The curse of dimensionality shows clearly with both datasets, but with over 100 attributes and only around 30k samples to train and validate on in the census income dataset the problem is magnified. One thing I'd like to explore in the future is to transform the census income data differently to reduce the number of attributes. For example, I'd like to simply represent the country with a single number and make that a single feature (eg: US: 0, Cambodia: 1, UK: 2 etc).

One last thing to note is that neural networks performed among the worst of the learning algorithms on my datasets. This is again probably a result of the curse of dimensionality and there not being enough samples for the network to generalize to the held out test set.

Boosting

Census Income

		mean_fit_time	mean_score_time	mean_test_score	mean_train_score
base_estimator__ccp_alpha	**n_estimators**				
0.0025	**25**	17.502893	0.088497	0.860609	0.865326
	50	35.520115	0.178132	0.860609	0.863790
	10	7.193724	0.045559	0.860345	0.863790
0.0010	**10**	8.568175	0.049728	0.856046	0.892539
0.0100	**10**	5.970859	0.043118	0.852229	0.853315
	25	14.617631	0.092372	0.852229	0.853315
	50	29.844664	0.192391	0.852229	0.853315

Breast Cancer

		mean_fit_time	mean_score_time	mean_test_score	mean_train_score
base_estimator__ccp_alpha	**n_estimators**				
0.029287	**25**	0.358019	0.011244	0.964842	1.000000
0.009484	**100**	0.573894	0.017842	0.962184	1.000000
0.030721	**50**	0.713692	0.021772	0.959842	1.000000
0.029287	**100**	1.427723	0.041147	0.959747	1.000000
0.030721	**100**	1.443515	0.040778	0.959747	1.000000
0.029287	**50**	0.711700	0.021334	0.957215	1.000000

Adaboost was used on the same decision tree code that was used for the Decision Tree section above. A grid search was performed using the same cost complexity parameters used on the decision trees above for pruning as well as a varied number of weak learners to use. As shown in the data above, which is sorted by test scores, unpruned trees did not make it into the top testing scores for either dataset. Additionally the boosting improved test accuracy for both datasets as well. This is the expected trend given the usage of boosting.

Again the training phase takes much longer for both datasets which is also expected because this is simply composed of many decision trees which are all eager learners. One

other thing to note about the training times is that it seems to grow close to linearly with the number of weak learners used as shown in the figure below.

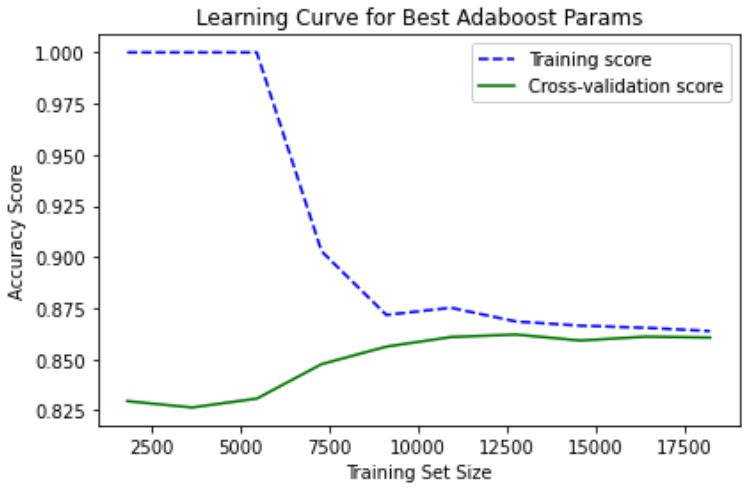
Census Income Adaboost with Cost Complexity Parameter 0.001

n_estimators	mean_fit_time	mean_score_time	mean_test_score	mean_train_score
10	8.568175	0.049728	0.856046	0.892539
25	22.696713	0.106779	0.832046	0.974476
50	44.675716	0.195658	0.841610	0.960139

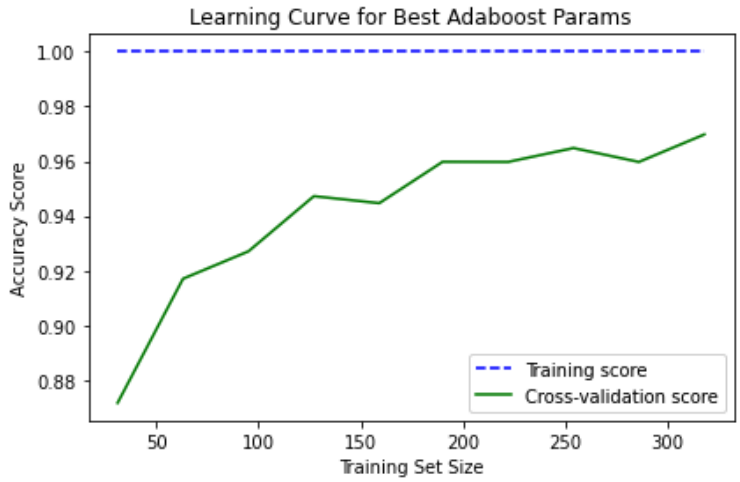
Breast Cancer Adaboost with Cost Complexity Parameter 0.030721

n_estimators	mean_fit_time	mean_score_time	mean_test_score	mean_train_score
10	0.147207	0.008324	0.929652	1.0
25	0.371019	0.014259	0.954715	1.0
50	0.745428	0.028089	0.964778	1.0
100	1.485498	0.047332	0.969810	1.0

Census Income Learning Curve



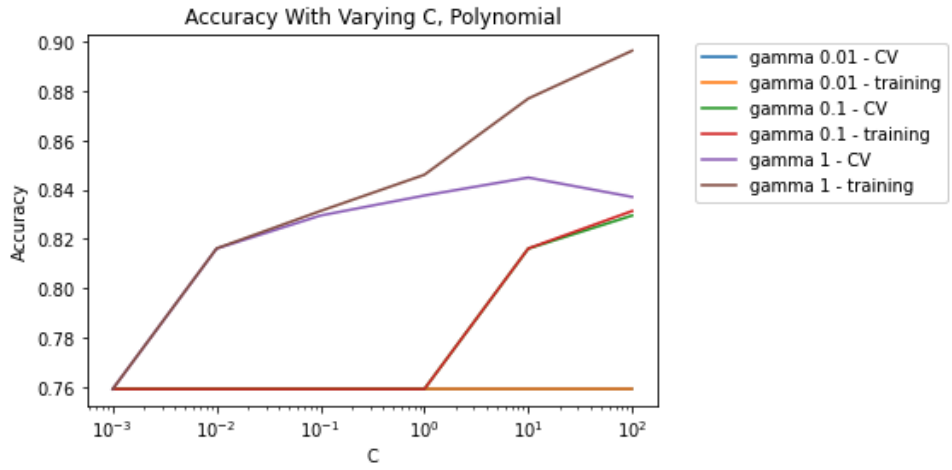
Breast Cancer Learning Curve

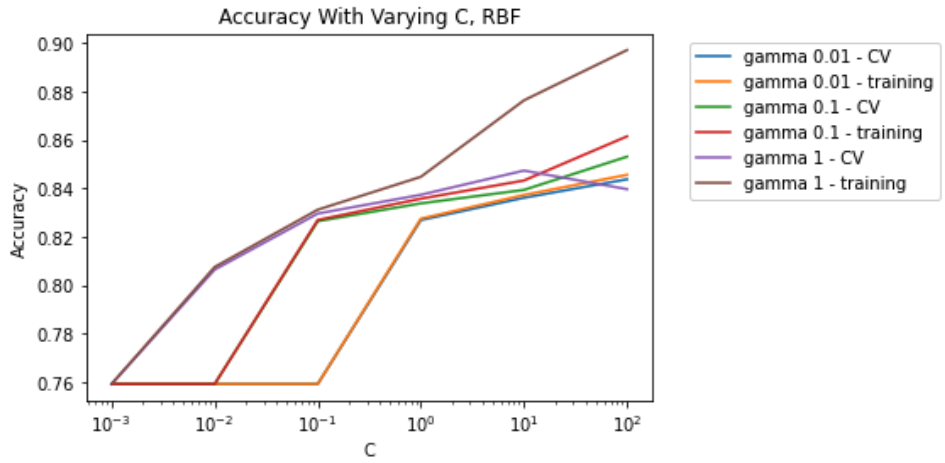


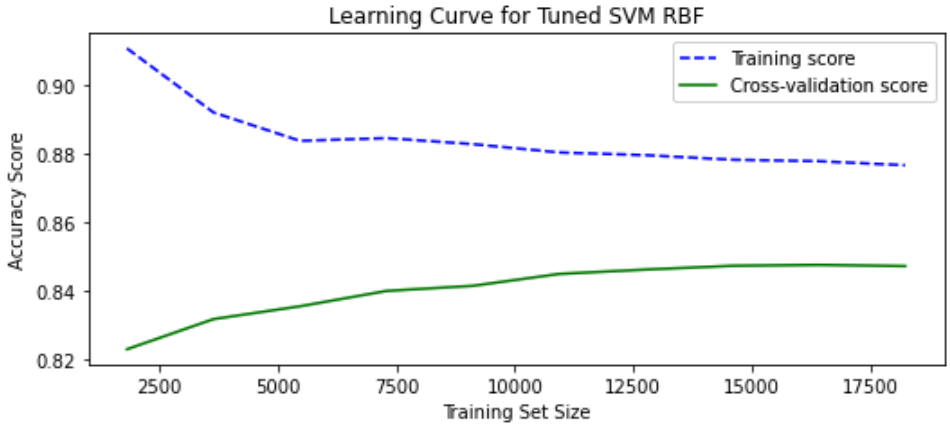
We can see from the above learning curves that for both datasets the training and validation scores move towards converging which is a behavior we want for a model that generalizes reasonably well. One thing to note is that the training accuracy for the breast cancer dataset never drops below 100%. This is most likely due to there not being very many samples and that there is a 1:1 mapping of inputs to outputs, meaning there are not 2 samples where they have the same attributes but different outputs. One final thing to note is that adaboost performed the best out of all algorithms for the census income dataset. My guess as to why this is true is because the dataset does not have a lot of continuous attributes and decision trees perform better with those kinds of datasets. The Adaboost decision trees also performed the best on the cancer dataset. I defined best as taking relatively less time to train and accuracy on the test set.

SVM

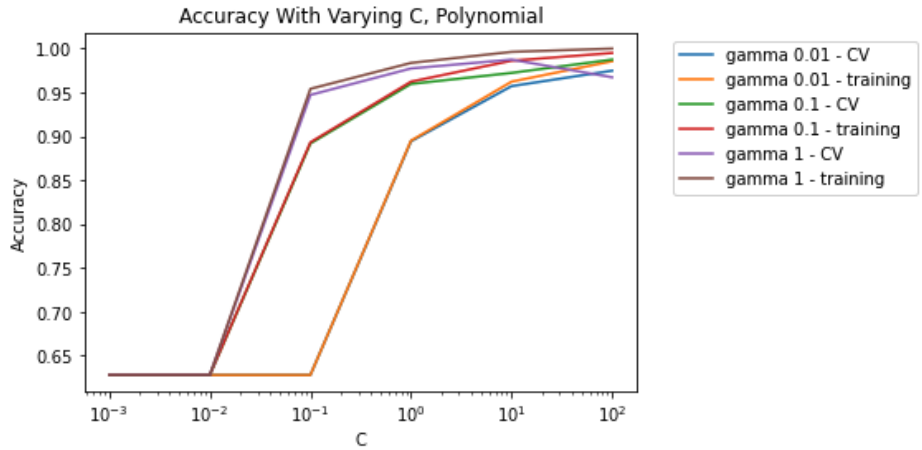
Census Income Max Iterations 10000

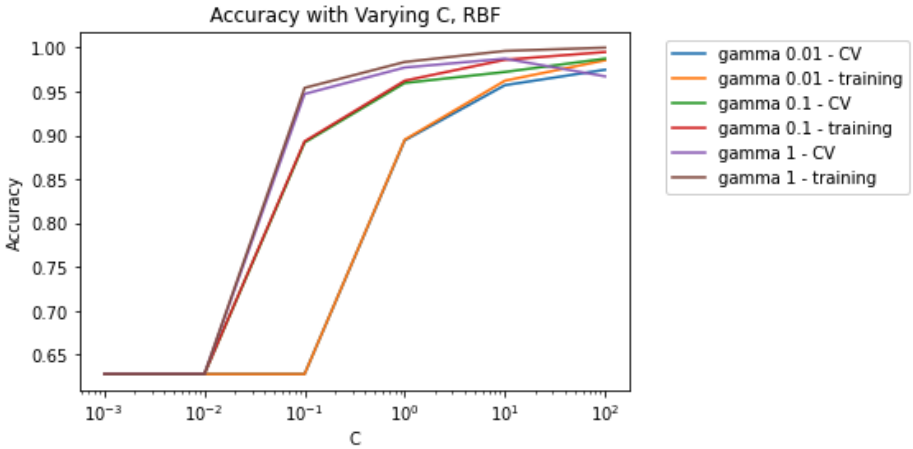


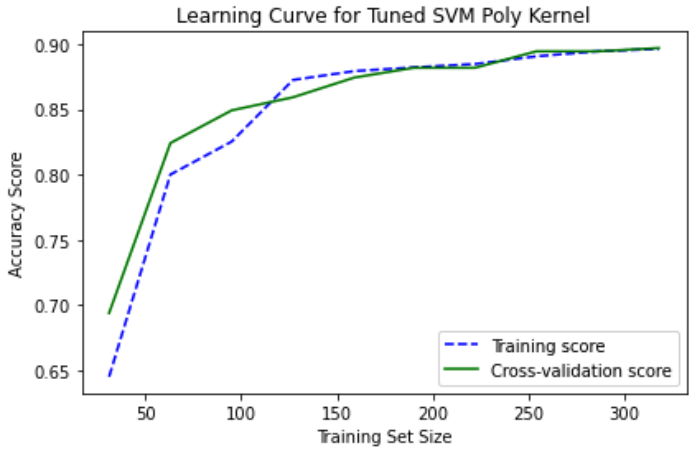




Breast Cancer Max Iterations 10000





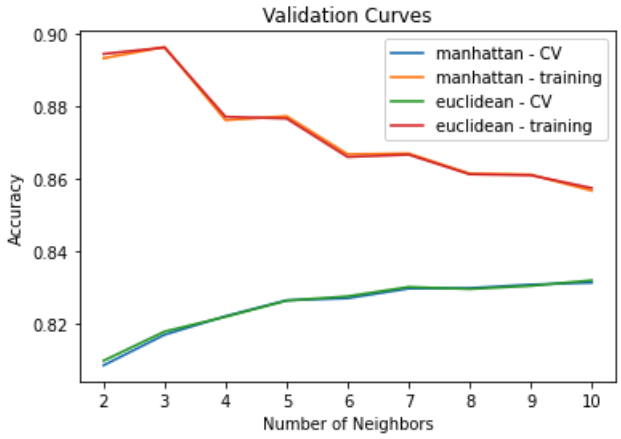


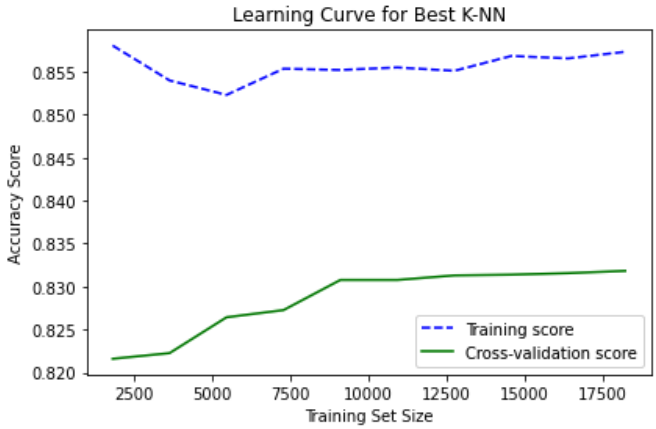
In the experiments with support vector machines, I chose to use a polynomial kernel and a radial basis function kernel. For the experiments I used varying values for gamma and C. As shown above, a larger gamma led to the training and validation scores increasing more quickly as well as diverging more quickly as C was increased. This makes sense as the gamma parameter affects how much influence a single training example has. Additionally, with a larger C, the model makes a greater effort to correctly label training examples. Thus with larger values of gamma and C the model is more prone to overfitting and this can be shown in the plots above as the cross validation and training scores diverge. One thing to note is that for the polynomial kernel on the census income dataset, for the smallest value of gamma the accuracy never improves.

k-NN

Census Income

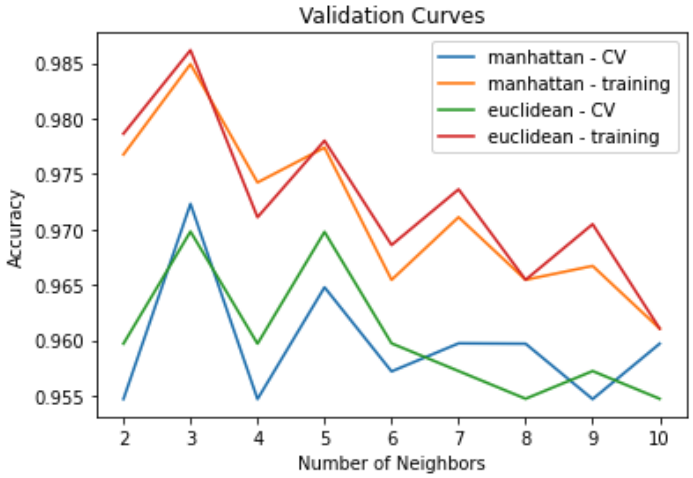
		mean_fit_time	mean_score_time	mean_test_score	mean_train_score
metric	**n_neighbors**				
euclidean	**10**	0.010173	2.707861	0.831827	0.857351
manhattan	**10**	0.010496	17.122805	0.831169	0.856616
	9	0.009841	16.995984	0.830642	0.861081
euclidean	**9**	0.010610	2.862879	0.830291	0.860905
	7	0.010365	2.892548	0.829984	0.866598

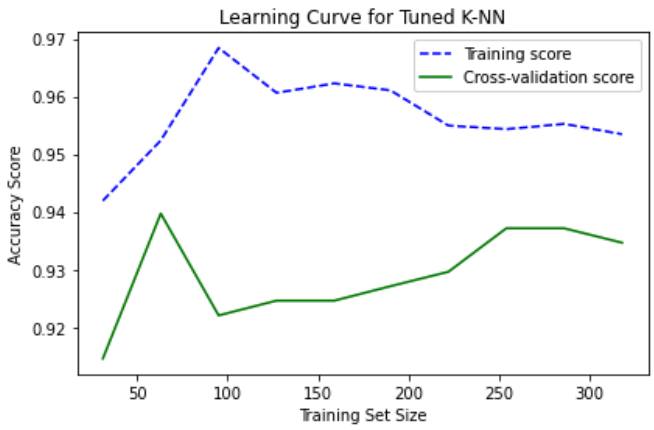




Breast Cancer

metric	n_neighbors	mean_fit_time	mean_score_time	mean_test_score	mean_train_score
manhattan	**3**	0.000770	0.004346	0.972310	0.984921
euclidean	**3**	0.001287	0.006580	0.969810	0.986179
	5	0.001066	0.005529	0.969778	0.978011
manhattan	**5**	0.000778	0.004635	0.964778	0.977384
	7	0.000759	0.004410	0.959715	0.971105





With both of the datasets, the fact that k-NN is a lazy learner is clearly highlighted. In both cases, the training phase takes significantly less time than the testing phase. Another interesting thing to note is that the distance metric used does not seem to really affect the accuracy of the models much for these specific datasets.

Additionally from the experimentation, the way the points in a neighborhood were weighted also did not affect the accuracy much. For example making the weights uniform or weighing the points by the inverse of their distance did not make much of a difference. This probably just means within the neighborhoods that are found the distance between neighbors is fairly uniform.