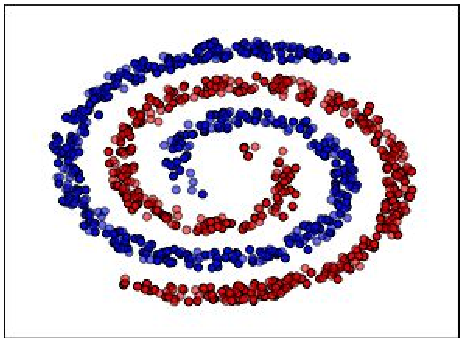
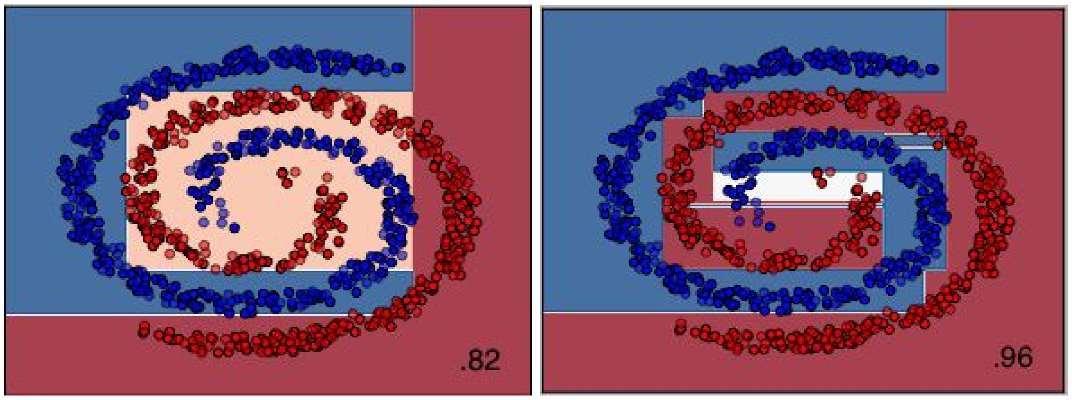
1. Two spirals dataset



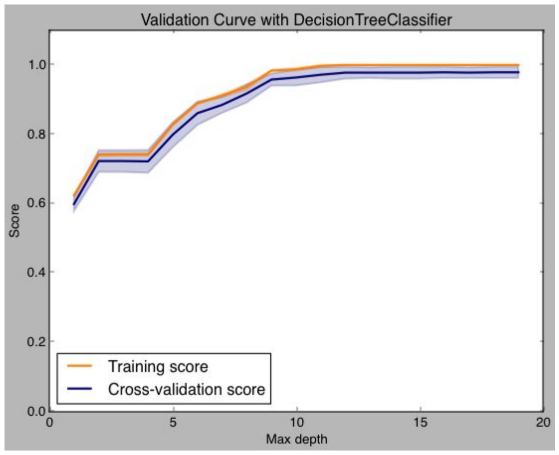
My first dataset is the two spirals dataset. This dataset caught my attention because it's extremely easy for humans to visually classify the points, but some of the machine learning approaches struggle, partly because it is non-linear.

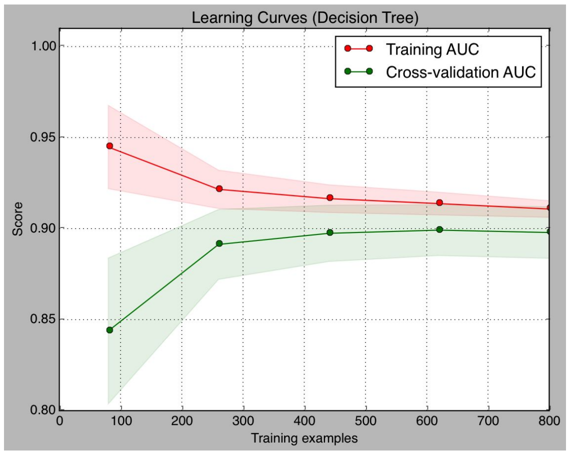
1.1 Decision Tree



1: *On left, decision tree using max_depth = 5. On right, decision tree using max_depth=10*

On initial tests, decision trees had an AUC of 0.82 using a maximum depth of 5. I tested the optimal value for maximum depth, which can be seen in the validation curve below. Using the validation curve, it looks like the score flattens around a max depth of 10. After increasing the maximum depth to 10, the decision tree performed much better with an AUC of 0.96. The visualization can be seen in figure 2.

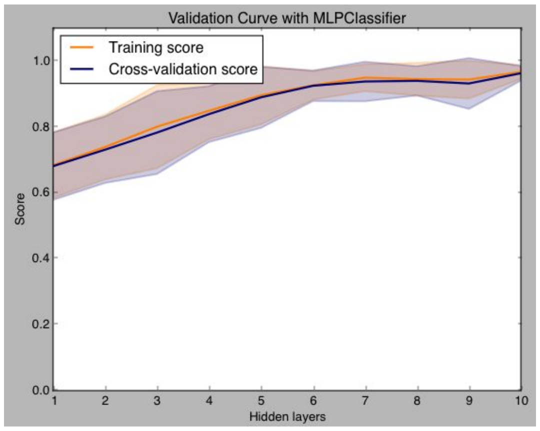


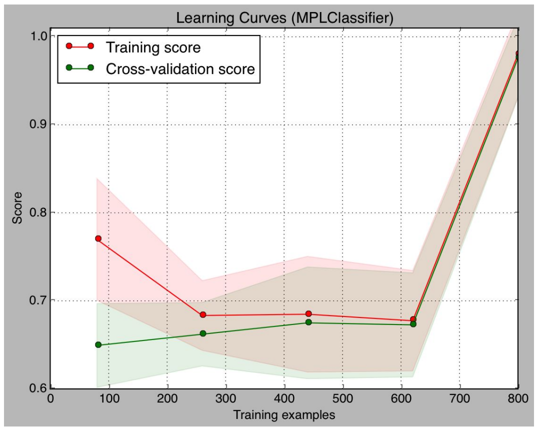


accuracy score as max depth increase. Right, learning curve

1.2 Neural Network

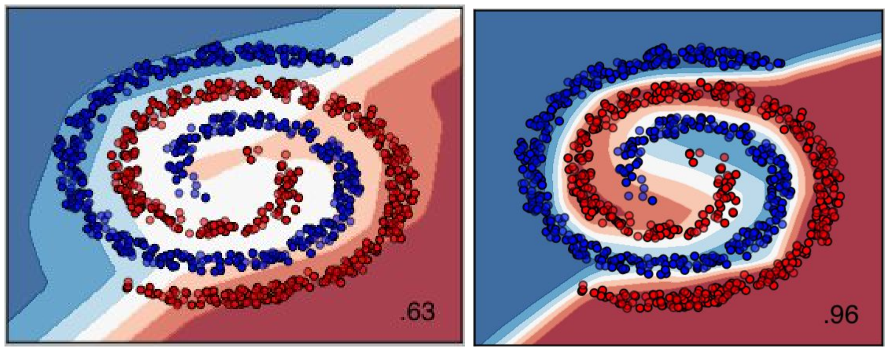
On initial tests using a multi-layer perceptron using default parameters, the neural network performed poorly achieving an AUC of 0.63. I investigated tuning the hyperparameters and while most parameters showed mostly noise, increasing the number of hidden layers looked promising.





ayers increases. Right: Learning curve showing accuracy
les increases

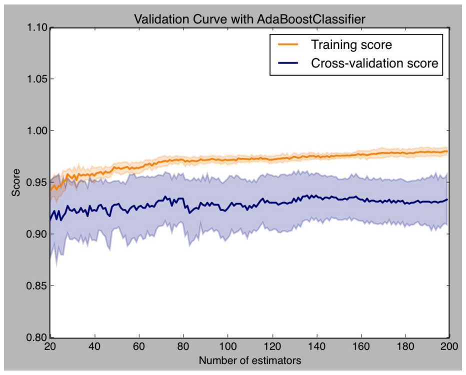
After performing grid search and finding that the best number of hidden layers is X, the performance was much better, achieving an AUC of 0.96.

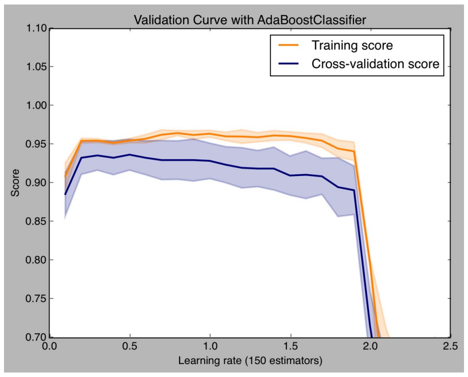


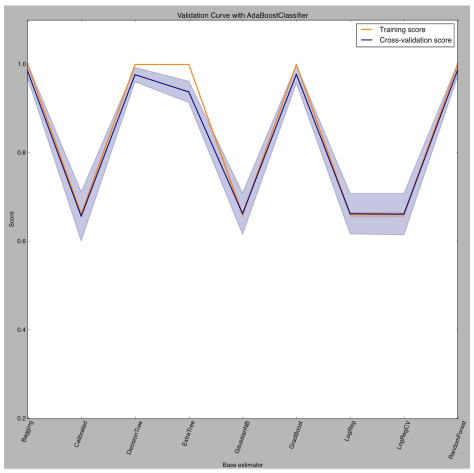
r tuning

Additionally, the learning curve for MLP turned out to be very interesting. Both the training score and cross-validation score rocketed up towards 1 after reaching a training sample size somewhere between 600-800.

1.3 AdaBoost

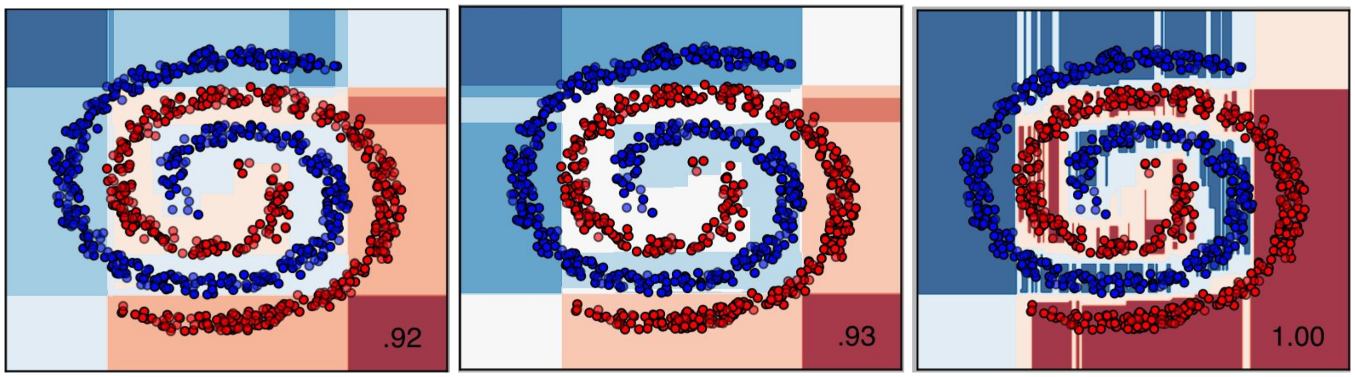




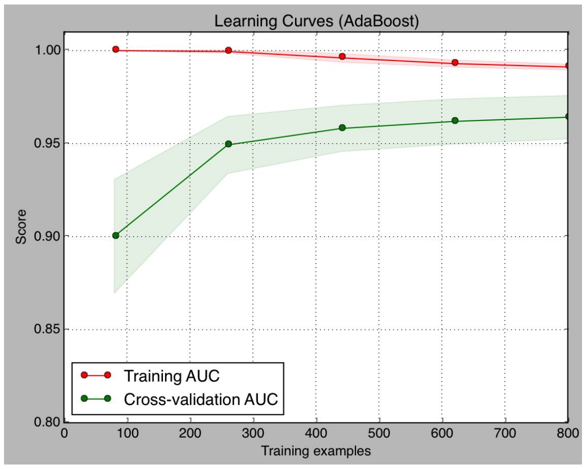


eases. Right,

Using default parameters, AdaBoost performed with an AUC of 0.92. After looking at the validation curves for learning rate and number of estimators, I thought that the best combination would be around 150 estimators and 0.5 learning rate. However, after using those parameters the performance was slightly worse at an AUC of 0.91. I then ran grid search and the best parameters came out to be 0.30 learning rate and 125 estimators.



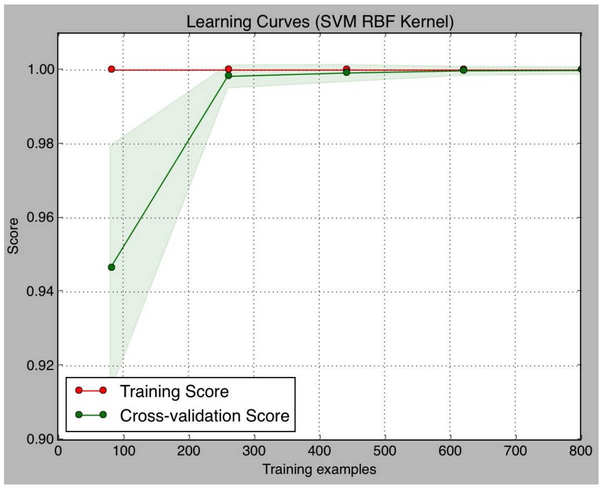
Unfortunately, even using the grid search parameters only had a minor improvement in AUC, moving from 0.92 to 0.93. However, I also measured ran a validation curve with different base estimators and found that Random Forest performed the best. Using Random Forest as the base estimator improved the AUC to a perfect 1.0.

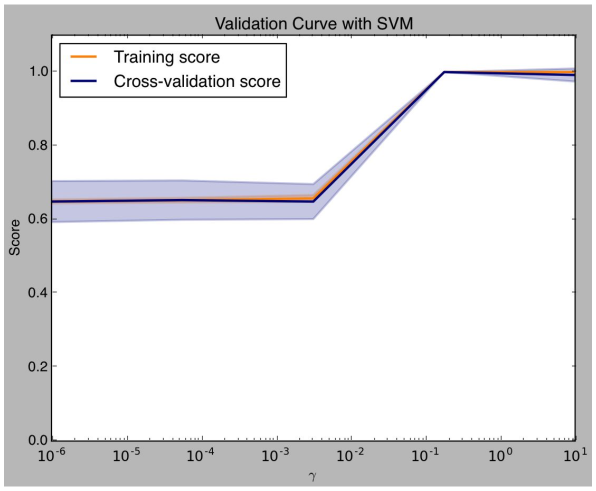


The learning curve indicates that AdaBoost sees a big jump in performance after about 300 samples, and then slowly converges as the number of training samples increases.

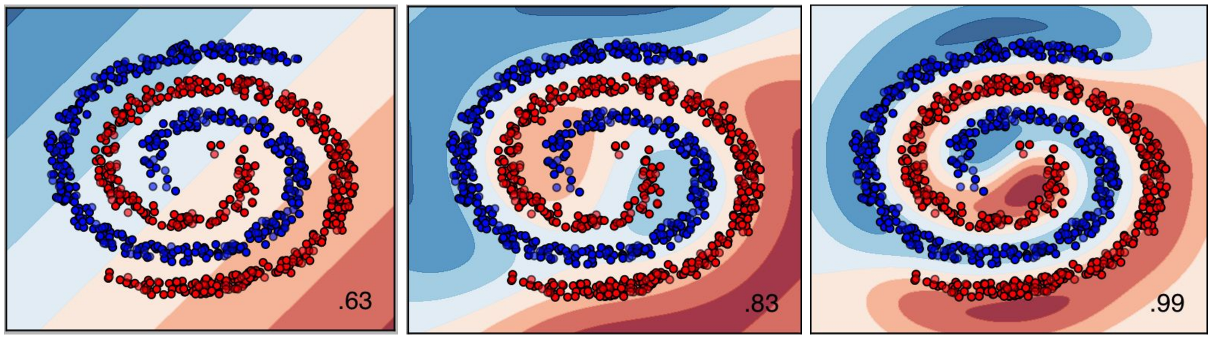
1.4 SVM

Linear SVM predictably performs very poorly with an AUC of 0.65. However, using the RBF (Radial Basis Function) kernel and default parameters it performed significantly better with an AUC of 0.83. For the rest of this section I'm going to ignore linear SVM because it's a lost cause and I need to save space. Looking at the RBF learning curve it shows a big jump in cross-validation score at around the 300 training samples range and becomes almost perfect.

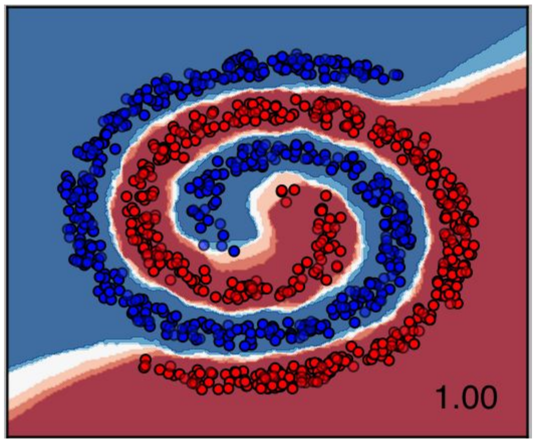




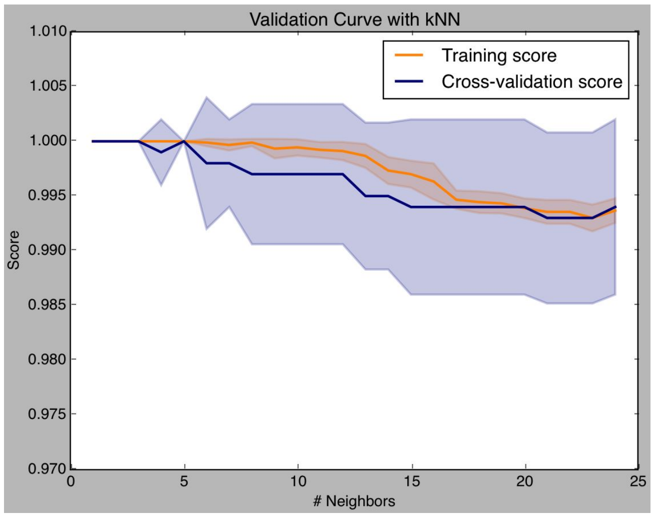
After looking at the learning curve, I wanted to tune the parameters so I could score above a 0.83 in AUC. The complexity curves show that parameters make a big difference in the RBF SVM as seen in the figure above. After running grid search, the best parameters for gamma and C turned out to be 0.3 and 0.1, respectively. Using those parameters, SVM achieved an AUC score of 0.99

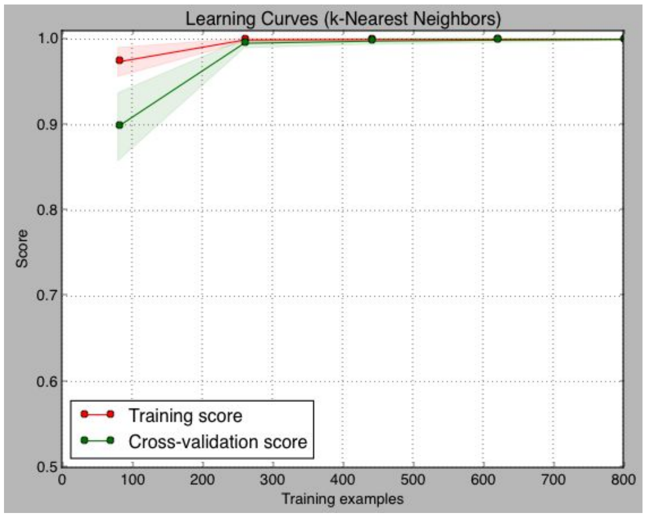


1.5 kNN



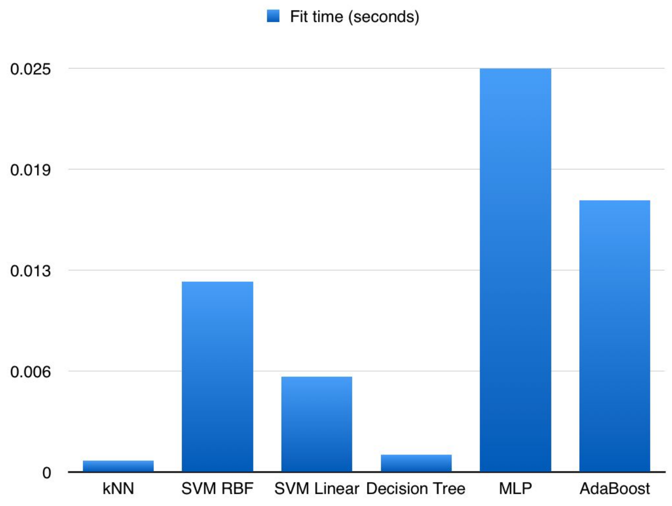
Using default parameters, kNN performed perfectly on the dataset achieving an AUC of 1.0. So there isn't much to tune here as far as classification goes, but if we can get away with a smaller k value that could potentially help with performance. Looking at the validation curve for kNN, it looks like we could go slightly lower than 5 and want to avoid anything higher.

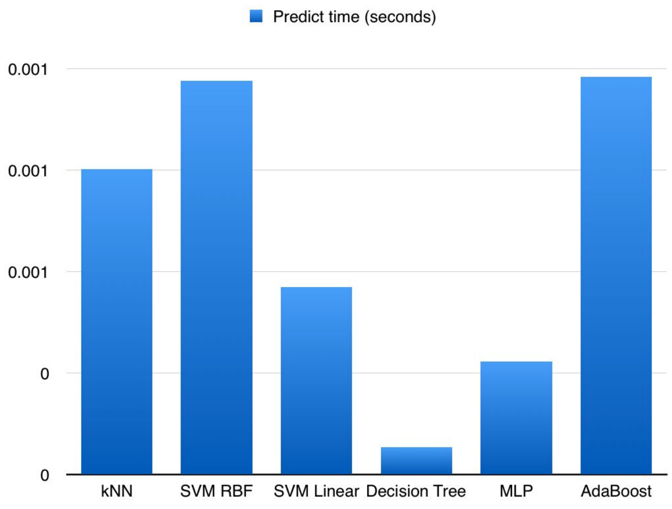




Additionally, the learning curve shows that cross-validation score matches the training score quickly, somewhere around 250 training examples.

1.6 Performance





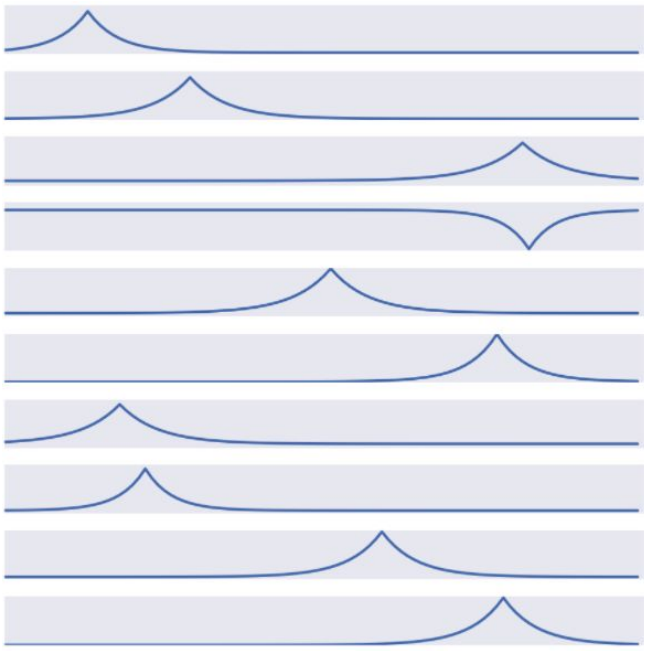
I measured fit time using 80% of the dataset and predict times using 20%. Predictably, kNN performed the best in fit time, with Decision Tree just behind. MLP was the slowest in fit time by a large margin. In prediction time, Decision Tree was the best, and kNN, SVM RBF, and AdaBoost all performed the worst.

1.7 Conclusions

Algorithm	AUC before tuning	AUC after tuning	Fit time	Predict time
Decision Tree	0.82	0.96	0.001065	0.00008108
MLP	0.63	0.96	0.6524	0.0003341
AdaBoost	0.92	1.00	0.01680	0.001176
SVM Linear	0.65	0.65	0.005885	0.0005532
SVM RBF	0.83	0.99	0.01179	0.001164
kNN	1.0	1.0	0.0006935	0.0009020

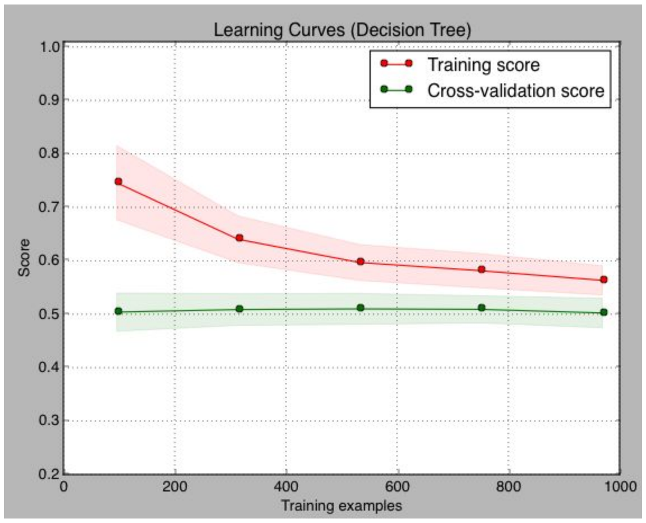
It’s hard to pick a “best” algorithm here because there isn’t a real life application for this dataset. If the predictions had serious consequences, like medical decisions, then the obvious choice would be kNN or AdaBoost because of it’s perfect AUC score. However, both kNN and AdaBoost are not the most performant, so if it were a situation where a 0.96 AUC score is good then I would pick Decision Tree as the “best”. 0.96 is still a very high AUC score and Decision Tree had the best overall performance, with the second best fit time and the best predict time.

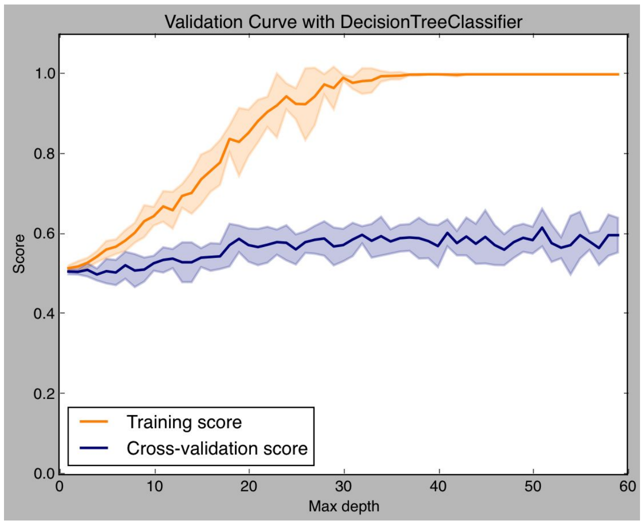
2. Hill Valley Dataset



This dataset was interesting to me because similar to the two spirals, it is extremely easy for humans to classify whether the signals have a hill or a valley. However, many of the machine learning algorithms struggle greatly. The hill valley dataset is time series data, so each sample consists of a series of columns that represent ‘ticks’ of time, and the Y value is the height at that time.

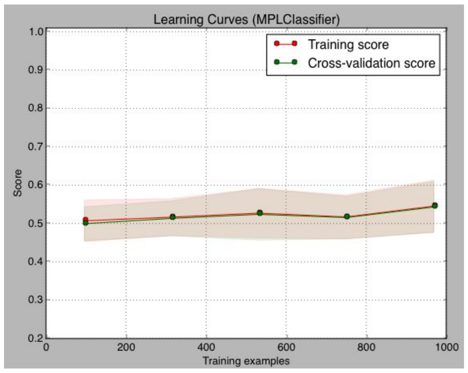
2.1 Decision Tree

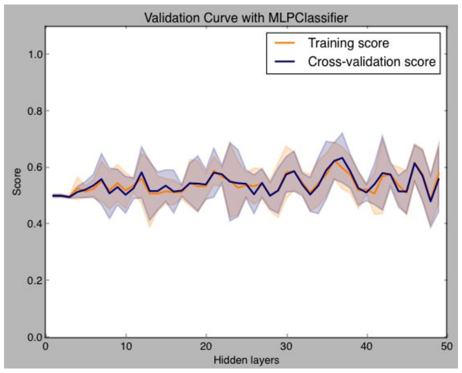


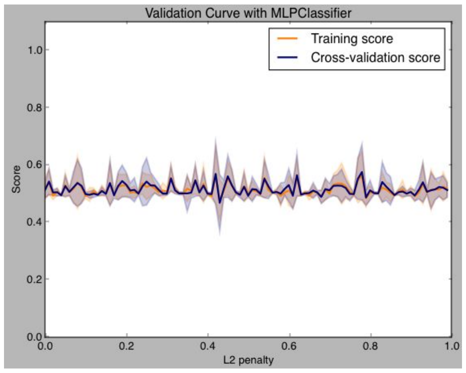


As seen in the figures above, decision tree performed poorly at classifying the hills and valleys with a .50 accuracy. Additionally, increasing the training set size sees no improvement, and increasing the complexity by max depth only sees marginal improvements. After increasing max depth to 40, the decision tree achieved an AUC of .5144.

2.2 MLP

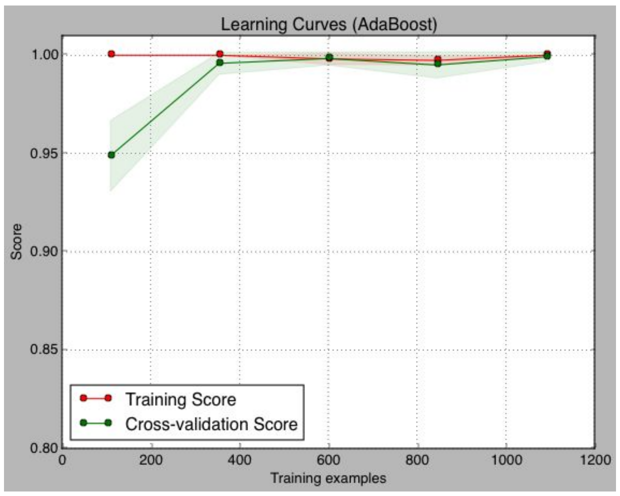


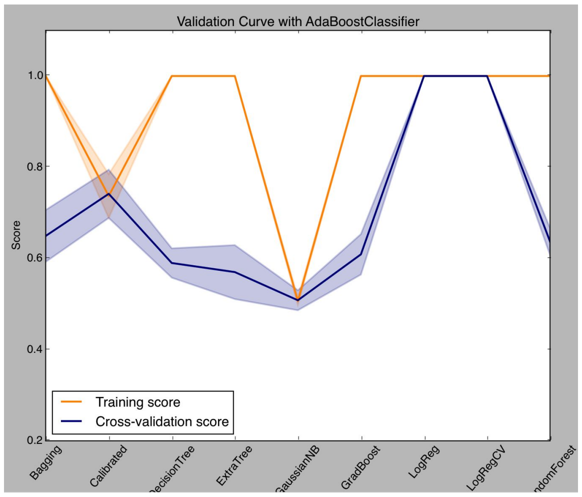




Referring to the learning curve figure above, the MLP unfortunately does not perform better after training on a higher number of training samples like it did for the two spirals dataset. In addition, all of the hyperparameter tuning I tried did not improve the accuracy at all. I tried tuning the number of hidden layers, increasing number of neurons, L2 penalty, momentum, and initial learning rate but none of those increased the accuracy.

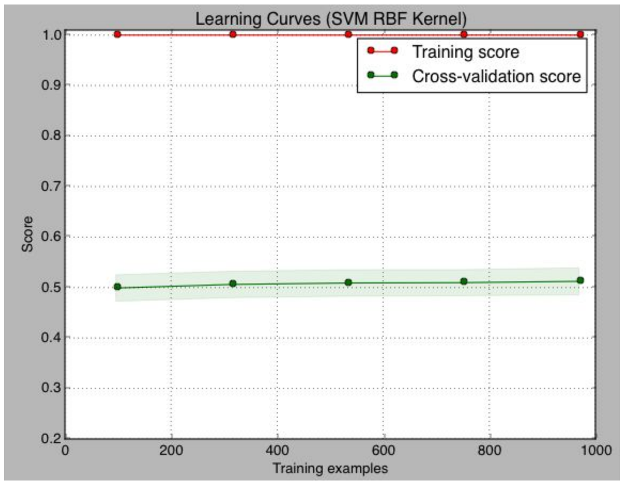
2.3 AdaBoost

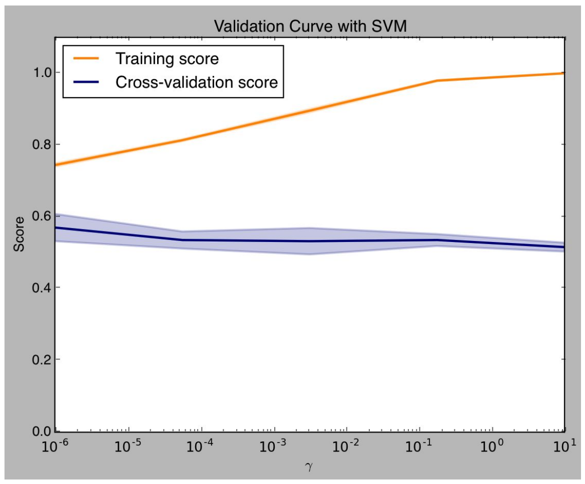




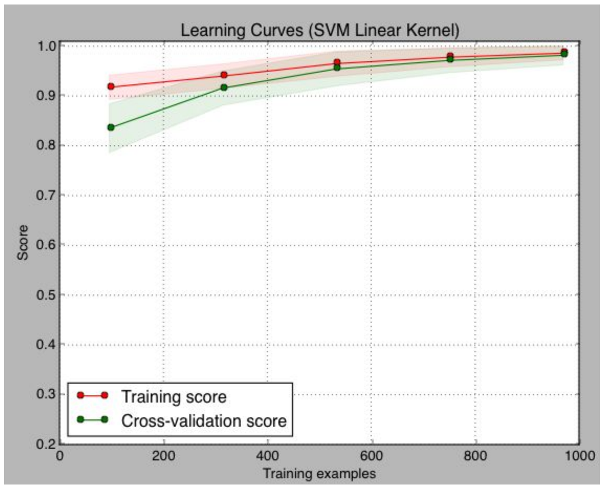
The base classifier used in AdaBoost made a huge difference as seen in the figure above. Most of them hovered around the 0.6 accuracy range, but using Logistic Regression as the base estimator brought the cross-validation score all the way up to a perfect 1.0

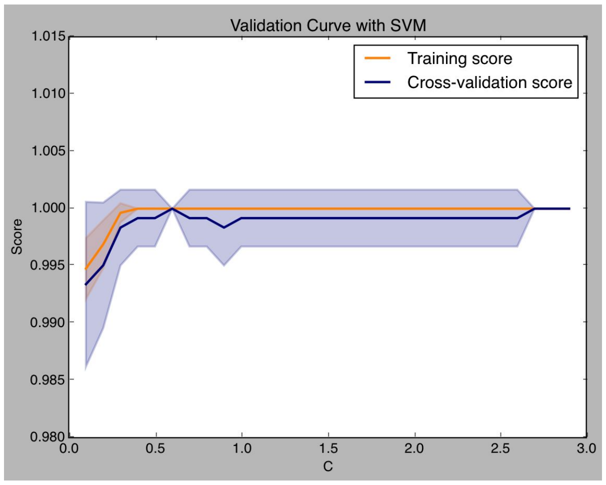
2.4 SVM





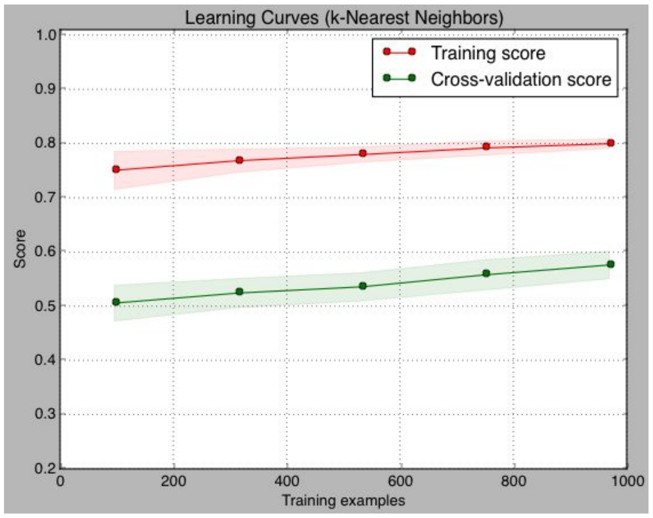
As seen above, SVM with the RBF kernel performed poorly, achieving around a .50 accuracy score. I tried tuning hyperparameters but saw no improvement. However, when using a linear kernel SVM performed very well.

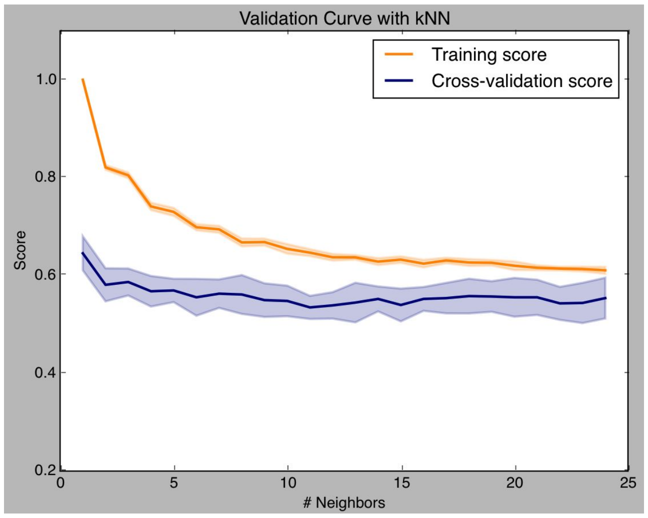




As seen in the learning curve, the accuracy starts off high but does have modest improvements as training size increases, and the cross-validation score converges towards 1.0 as the training size increases. Interestingly, when tuning hyperparameters a perfect score of 1.0 AUC score can be achieved with a C value of 0.6 or 2.7.

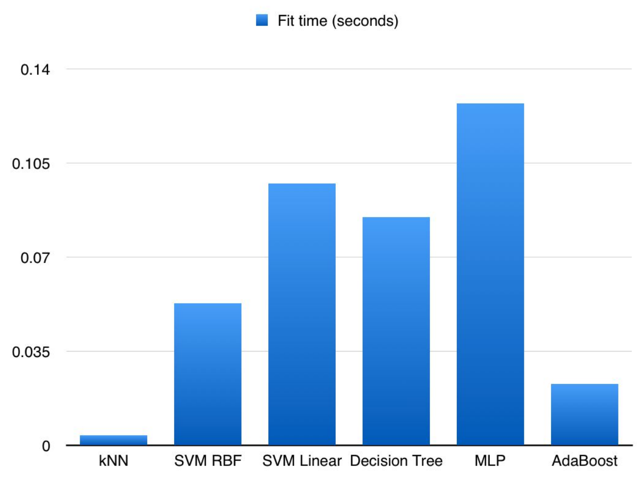
2.5 k-Nearest Neighbors

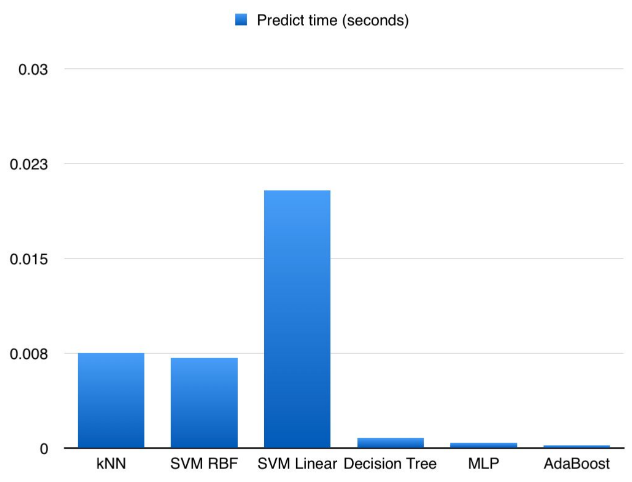




Using default parameters using a k-NN classifier, performance is poor with an AUC of 0.55. Additionally, as seen in the validation curve, increasing the complexity does not improve performance as the training score and cross-validation score converge when increasing number of neighbors.

2.6 Performance





I measured fit time using 80% of the dataset and predict times using 20%. Comparing fit times, kNN and AdaBoost were the best. In prediction time, Decision Tree, MLP, and AdaBoost were all very fast while SVM Linear was the slowest.

2.7 Conclusions

Algorithm	AUC before tuning	AUC after tuning	Fit time	Predict time
Decision Tree	0.6164	0.6085	0.08482	0.000796
MLP	0.5116	0.590.	0.12720	0.127203
AdaBoost	0.5183	1.0000	0.02272	0.022724
SVM Linear	1.0000	1.0000	0.09733	0.097336
SVM RBF	0.5289	0.5289	0.05281	0.052806
kNN	0.5597	0.6159	0.00090	0.003722

For this dataset, there is a clear best algorithm. After tuning, AdaBoost achieved a perfect 1.0 AUC score. Additionally, it is the best in terms of performance being second fastest (only behind kNN) in fit time and fastest in predict time.