Assignment 1 - Supervised Learning

Datasets

Source

Both datasets were taken from the UCI Machine Learning Repository. The HTRU2 dataset can be found at https://archive.ics.uci.edu/ml/datasets/HTRU2 (https://archive.ics.uci.edu/ml/datasets/HTRU2), and the Letter Recognition set can be found at https://archive.ics.uci.edu/ml/datasets/letter+recognition (https://archive.ics.uci.edu/ml/datasets/letter+recognition).

Data Dictionaries

HTRU2

COLUMN	FIELD	NOTES
1	Mean of the integrated profile.	
2	Standard deviation of the integrated profile.	
3	Excess kurtosis of the integrated profile.	
4	Skewness of the integrated profile.	
5	Mean of the DM-SNR curve.	
6	Standard deviation of the DM-SNR curve.	
7	Excess kurtosis of the DM-SNR curve.	
8	Skewness of the DM-SNR curve.	
9	Class	1 for legitimate pulsar. 0 otherwise.

Letter Recognition

COLUMN	FIELD	NOTES
1	Class	Capital Letter (A-Z)
2	x-box	horizontal position of box (integer)
3	y-box	vertical position of box (integer)
4	width	width of box (integer)
5	height	height of box (integer)
6	onpix	total # on pixels (integer)
7	x-bar	mean x of on pixels in box (integer)
8	y-bar	mean y of on pixels in box (integer)
9	x2bar	mean x variance (integer)
10	y2bar	mean y variance (integer)
11	xybar	mean x y correlation (integer)
12	x2ybr	mean of x x y (integer)
13	xy2br	mean of x y y (integer)
14	x-ege	mean edge count left to right (integer)
15	xegvy	correlation of x-ege with y (integer)
16	y-ege	mean edge count bottom to top (integer)
17	yegvx	correlation of y-ege with x (integer)

Data Pre-processing

No pre-processing was done on the letters dataset, as the attribute values had already been scaled. Each attribute in the HTRU2 dataset was scaled to be a value between 0 and 1. This was done using the formula:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Scaling is necessary so that some attributes aren't given more weight than others due to magnitude. For example, the k-nearest neighbors algorithm calculates the "closest" k points to a given record based on the attribute values. If one attribute has significantly more magnitude than others, that attribute will have much greater influence on the distance calculation than the others, and may impede the accuracy of the algorithm.

For each algorithm, the dataset was first separated into a training set (70% of data) and a test set (30%) of data. The test set was not used for any of the training, hyperparameter optimization, etc. It was only used to measure accuracy as a final step.

The Problems

HTRU2

The HTRU2 dataset posed the problem of classifying pulsar candidates as a positive case (legitimate pulsar) or a negative case (not a legitimate pulsar).

Letter Recognition

The Letter Recognition posed the problem of classifying records as one of the 26 capital letters in the English alphabet.

Why These Datasets?

I chose these datasets because while both were used for classification problems, the datasets were drastically different. The HTRU2 dataset contained a large number of records, and relatively low number of features. In contrast, the Letter Recognition dataset contained many more features. In addition, the problem for the HTRU2 dataset was binary classification (and was skewed with ~91% of records negative examples) while the problem for the Letter Recognition set was to group into 26 different classes. Finally the values for the letter recognition set had already been scaled, so I did not further scale the values. The values for the HTRU2 dataset were not scaled, so I scaled the values for each attribute to be between 0 and 1. Because both classification problems had different features, I believed they would be interesting to compare different algorithms.

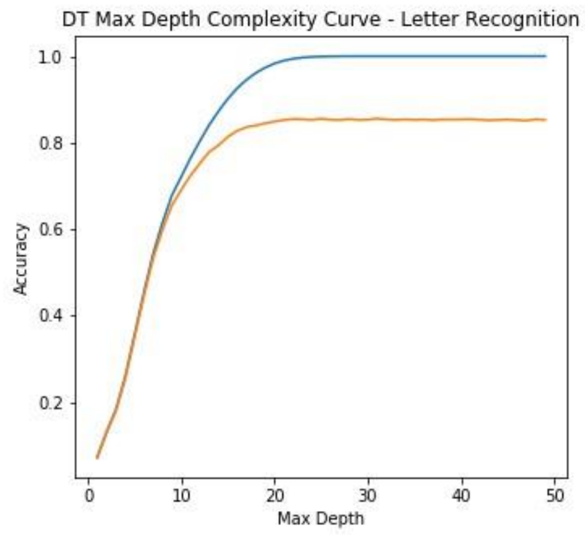
Code and Algorithms Used

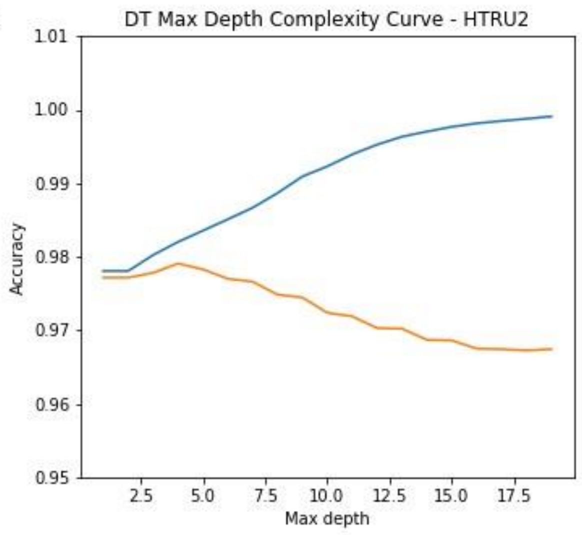
Each model was built using Python code, and the following models in the sklearn library:

tree.DecisionTreeClassifier, neural_network.MLPClassifier(), ensemble.AdaBoostClassifier(), svm.SVC(), neighbors.KNeighborsClassifier().

Decision Tree

The first algorithm used to model the two datasets was a decision tree. For each dataset, 70% of the records were selected randomly as a training set, and the remaining 30% were held out as a test set to be only used once the models were finished. One interesting hyperparameter in a decision tree model is the max depth of the tree. As a decision tree is created to fit training data, the tree can be allowed to grow large enough to perfectly classify the data, unless there are some anomalies (for example two records with identical attributes, but different classification). As a result, the tree can become so dependent on the training records selected that its ability to model new data points decreases. This is a result of the high variance created by allowing a tree to grow too large. However, if a tree is too small, it will exhibit high bias as the classification model is restricted to a smaller hypothesis space. In an attempt to balance these two factors, I plotted model complexity curves below. This was done using 5-fold cross validation. Each training set was broken into 5 folds, then each set of 4 folds was used to train a model which is then tested against the 4 training folds, and the 5th hold out fold.





Complexity Curve Analysis

The learning curves showed both similarities and differences. The HTRU2 dataset had very high accuracy on both the training and the cross validation set even with a small max depth value. This was not a surprise as ~91% of records in the dataset are negative examples, even an algorithm that assigns every record to the negative class would be expected to have a 91% accuracy. This is a high baseline, and decision tree algorithms could improve on this baseline, even with a small depth limit. The letter recognition dataset showed very low accuracy with small max depth values, and drastically increased as the max depth was increased. There are many more target classes, so I would expect it to take more branches in the decision tree to create a satisfactory model to group into one of the 26 classes.

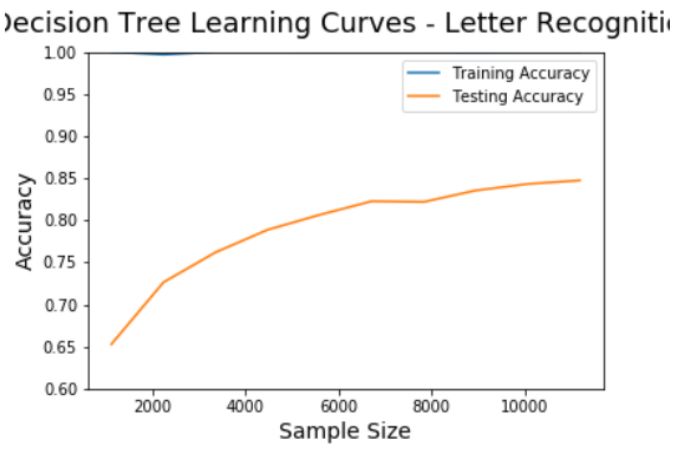
Despite the difference in initial behavior, the datasets showed similarities as the max depth increased. Accuracy on the training sets for each approached 1. As the decision trees are given more freedom to grow larger and larger, they can completely, or almost completely model the training data. While this seems like a good result, the complexity curves reflect that there is a cost. Although the bias is reduced, increasing the max depth hyperparameter increases variance. The tree becomes so dependent on the training set that it exhibits overfitting, and ability to predict classes in the hold out set is reduced. This is clearly seen in the HTRU2 complexity curve, as the accuracy against the validation set increases at first, but then steadily declines. While not as obvious in the letter recognition set, it is apparent that at some point adding to the max depth does not increase the accuracy on the held out validation set.

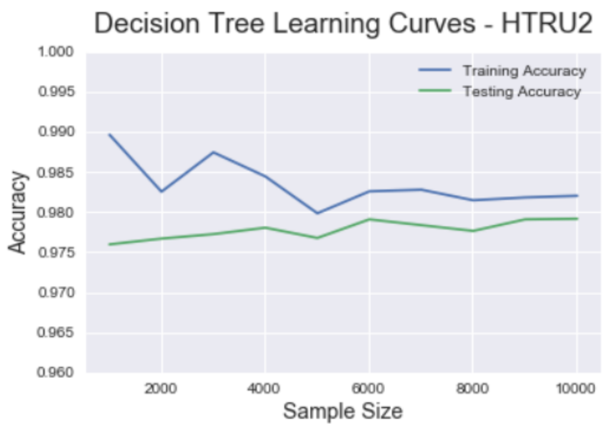
Optimizing Hyperparameters

After analysis of the complexity curves, the next step was to optimize the hyperparameter of max depth. This was done using a Grid Search method. This method is given a set of possible hyperparameter values, and for each possible value conducts kfold validation. The results of this analysis can be used to find the "best" values of the hyperparameter, and to train a model using the best value. After running each of my training sets through this grid search process (using 5-fold validation) I determined that the optimal max depth parameters for the HTRU2 and Letter Recognition datasets were 4 and 44, respectively. This reflected my analysis of the complexity curves.

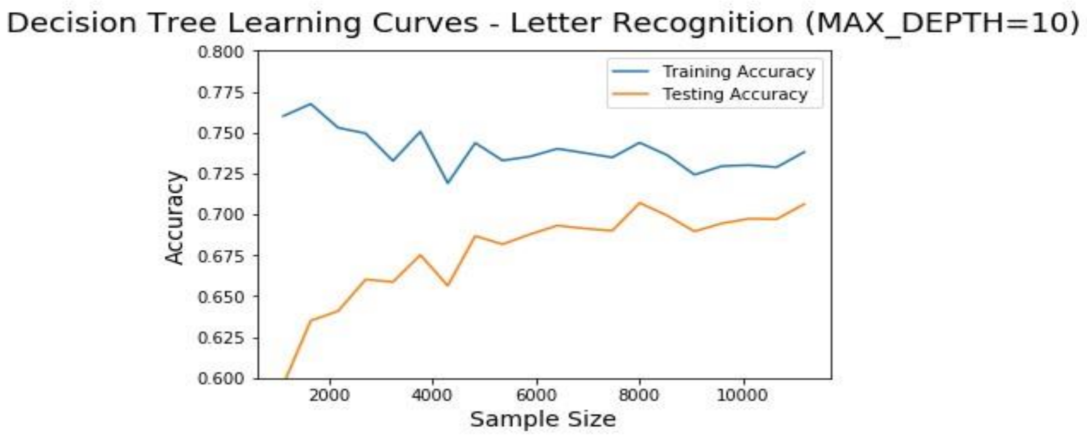
Learning Curves

The final step in Decision Tree analysis was to plot the learning curves. A learning curve shows how the size of the training set may affect the model's accuracy on both the training set and a validation set.



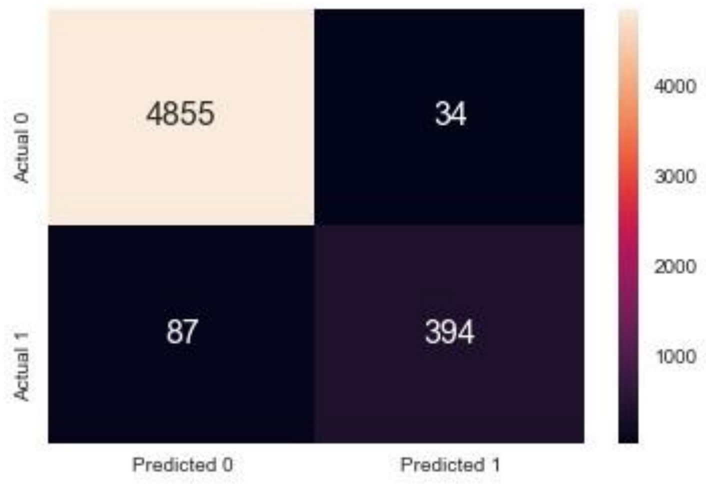


Both graphs showed an increase in accuracy on the testing sample as the size of the training sample increased. This is expected; the more data the model can see and learn from, the more it adjust its model to better predict future classes. The HTRU2 graph shows a slight decrease in accuracy on the training set as the size of the training set increases. A model can easily fit to classify a small number of training examples correctly. However, as the training set increases in size, the model can no longer classify every training example correctly (due to restrictions on the depth of the tree). Despite this fact, the Letter Recognition data model did not seem to decrease in accuracy on the training data even when training size got very large. I attributed this to the larger max depth parameter. By allowing the model to grow to a depth of 44, it has a greater ability to model training data - even as the training set grows large. To test this theory, I plotted a second learning curve for a decision tree, this time setting the max depth to 10. The graph below shows that in fact if the tree model is further restricted, its ability to model all the training data does decrease as the training set grows larger.



Accuracy

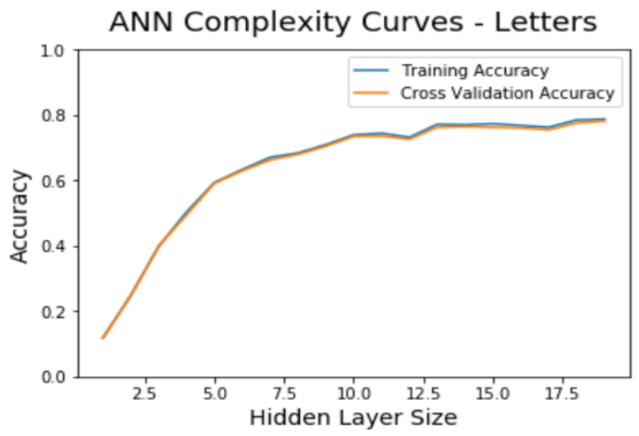
The final step was to use my models to predict the class of the test sets, and compare the predicted classes to the actual classes. The HTRU2 model was able to predict the correct class in 97.7% of the test cases. The Letter Recognition model was able to predict the correct class for 86.8% of the test cases. While this might indicate that the HTRU2 model was better, the nature of the datasets prevents direct comparison. As mentioned previously, a simple algorithm that predicts a negative class for any attribute values would in fact be expected to predict 91% of test cases accurately. To visualize this, I created a confusion matrix for the HTRU2 model, below:

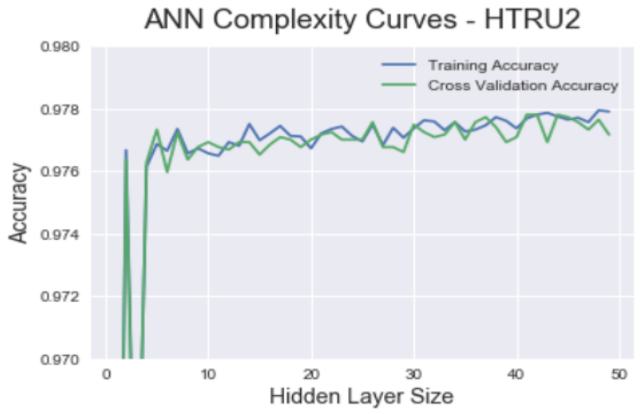


While the model was successful in >99% of negative test cases, it was not as successful predicting positive cases at around 82%.

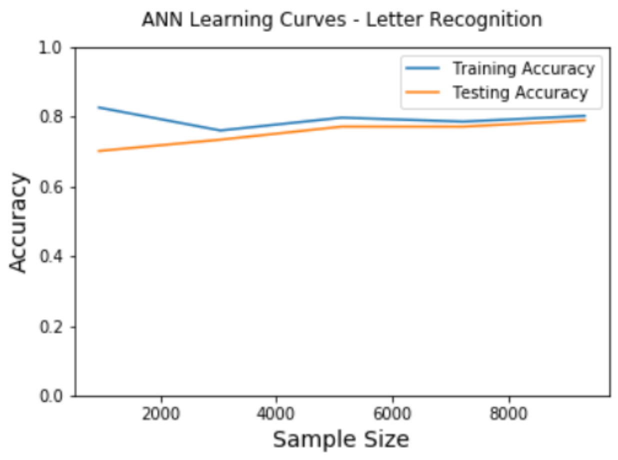
Artificial Neural Networks

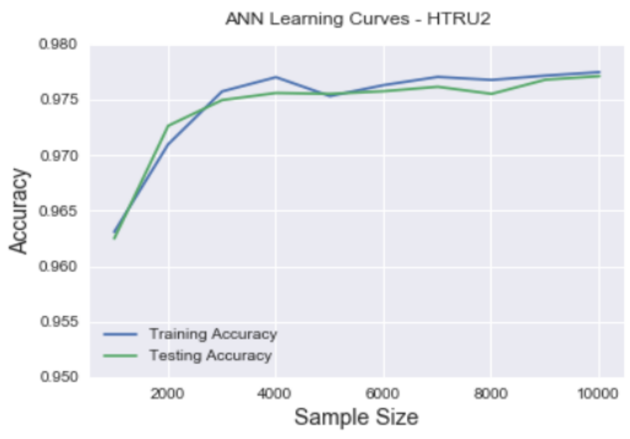
The next algorithm used to model the data was an artificial neural network, using sklearn MLPClassifier. First I graphed model complexity curves for models with 1 hidden layer, but different numbers of neurons in the hidden layer.





To optimize the parameters for the ANN model, I used a random search. In addition to different hidden layer sizes, I also searched different activation functions and learning rates. Since a greater number of parameter combinations was used, random search was able to tune the hyperparameters much quicker than an exhaustive grid search. For the HTRU2 data, the optimal parameters were: hidden layer size=13, activation function=tanh, learning rate=constant.

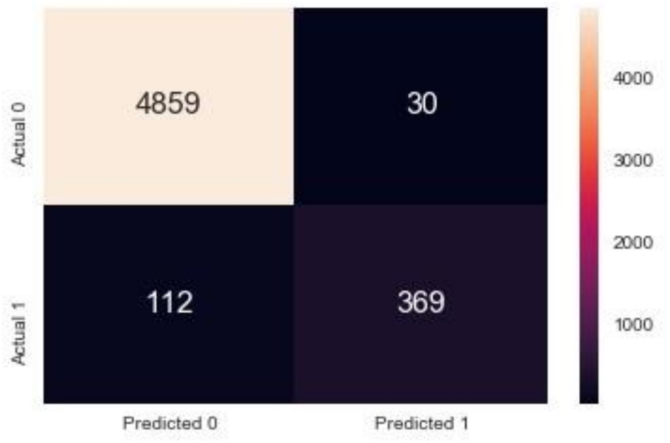




Like many of the other algorithms, the Letter Recognition ANN model did not appear to lose accuracy on predicting the training set as training size increased. Again, I attribute this to little noise in the data, so that even as more training examples are added the model can still predict the training examples accurately. As more training data was used, the testing accuracy did increase, and approached the accuracy of the model on the training set.

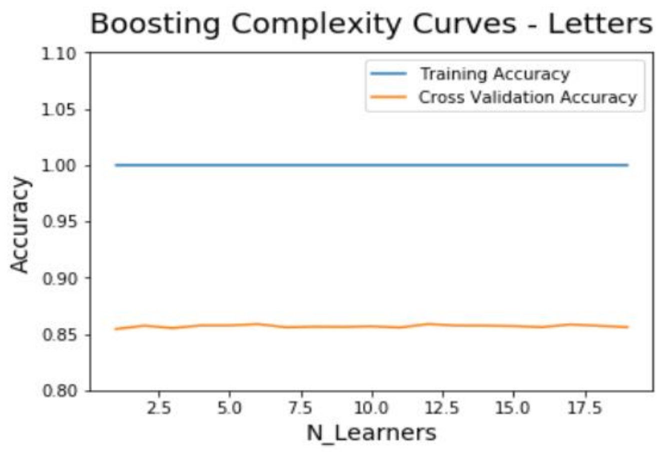
Accuracy

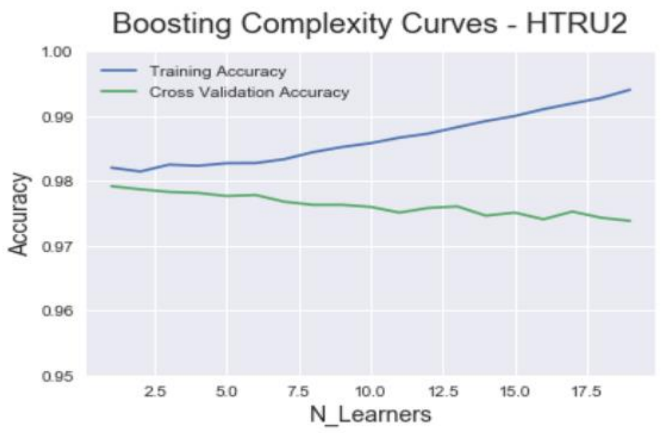
The Letter Recognition ANN model correctly classified 79.5% of the test records. The ANN model for the HTRU2 dataset correctly classified ~97.4% of the test records. Again, I plotted a confusion matrix for the HTRU2 dataset to see the false positive and false negative rates. The ANN model was only able to predict 76.7% of the legitimate pulsars correctly.



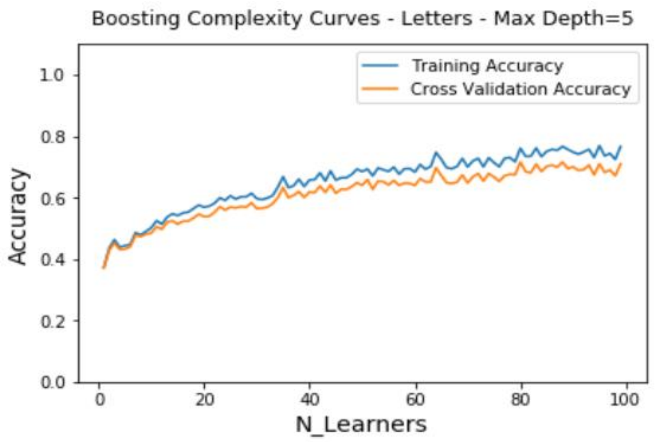
Boosting

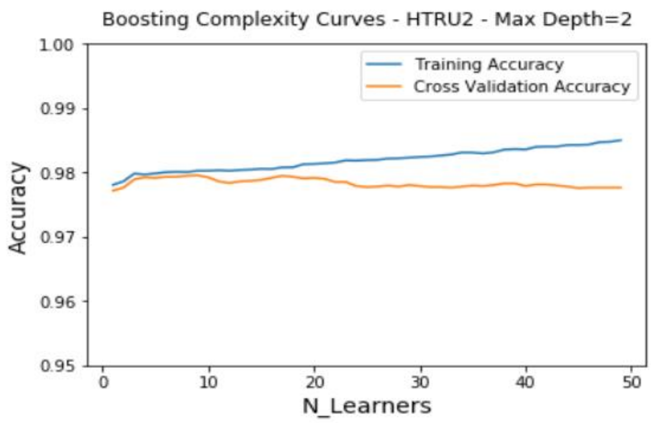
The next algorithm used, was adding boosting to the decision tree model. The goal of boosting, is to take several weak learners (in this case the decision trees) and combine them to form a more accurate model.





For both datasets, boosting did not appear to significantly increase performance. This may be due to the fact that the decision tree models for both datasets already appeared to be fairly accurate models. Boosting is usually more helpful when the models used are not already by themselves able to accurately classify data with a high degree of accuracy. One interesting observation, was that the boosted decision tree for the Letters dataset had a 100% accuracy for all the different number of learner values. The letters decision tree was allowed a max depth of up to 44, which allows it to conform closely to training data, and in this case it appears to perfectly model the training data. This dataset appeared to be resilient to overfitting in the decision tree model, and the boosted version showed similar behavior. To further investigate the affects of boosting, I ran a second model complexity curve on the letters dataset, but using boosting with decision trees limited to max depth=5. Based on the decision tree complexity curve, this parameter would greatly reduce the accuracy of the decision tree model, so I was curious if using boosting with this model would have a more noticeable effect. After running a grid search to optimize the number of learners, I found that the HTRU2 data did best with 1 learner, while the Letter Recognition set did best with 15 learners.

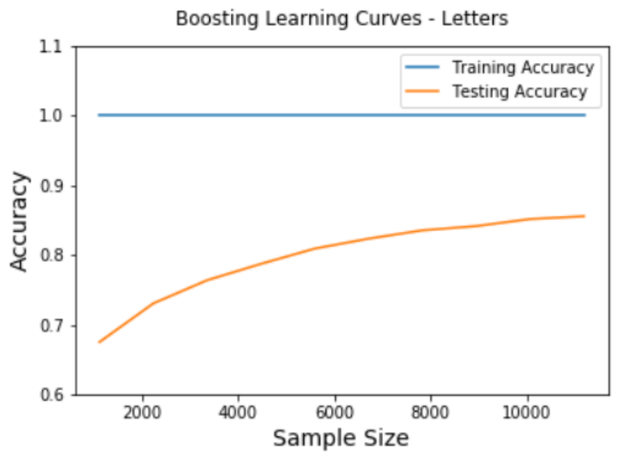


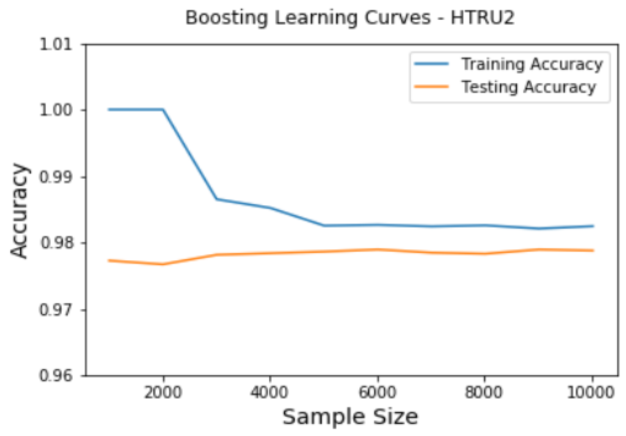


For the HTRU2 dataset with a lower max depth, the boosted version of the model had more typical complexity curves. As the number of learners increases, the model can grow more complex allowing it to more accurately fit to training data. However, by expanding the hypothesis space and reducing the bias, variance is introduced into the model. This means that a model can grow so complex as to model nuances and noise in training data causing overfitting, and decreased accuracy on the cross validation set. The Letter Recognition dataset, did not seem to suffer from decreased accuracy as number of learners was increased. Again, I attributed this to the fact that the data was not very susceptible to overfitting using decision tree models.

Learning Curves

The learning curves for each dataset using boosting with decision trees are below:

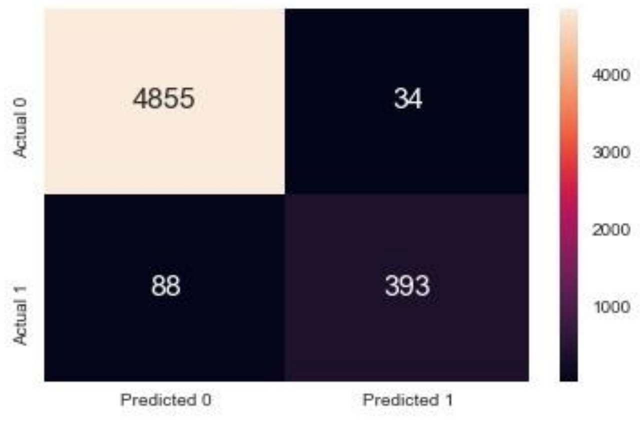




As expected, the training accuracy was very high (100%) for both datasets while the sample size was small. The decision tree with boosting model could correctly classify all the training examples while the training size stayed relatively small. For HTRU2, tt was interesting that the boosted model appeared to have better accuracy for these small sample sizes than the decision tree model. The boosted version had 100% accuracy for a training sample of 2000, while the original decision tree had very high accuracy, but not 100%. The boosted version of the Letter recognition dataset had 100% accuracy on the training set even with large training sets. The decision trees used in the boosting had a max depth of 44, which allows the model to grow very complex and model training data perfectly. It appears that this did not result in any overfitting however, as testing accuracy continued to increase.

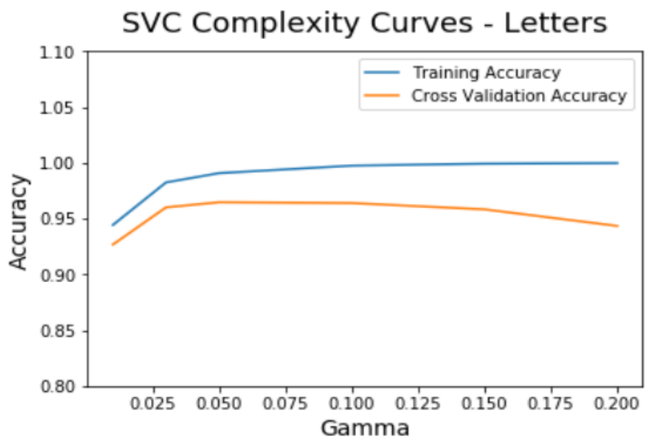
Accuracy

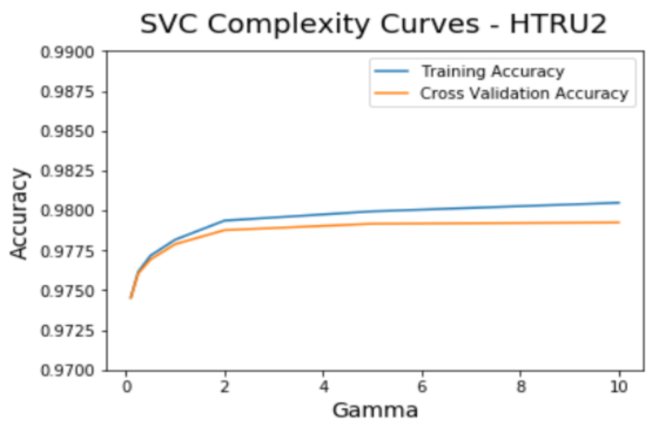
The boosted decision tree model for letter recognition set accurately classified 87.2% of test cases. This was slightly higher than the basic decision tree model, but was not a major increase in accuracy. The boosted decision tree model for the HTRU2 dataset accurately classified 97.7% of test cases. Again, I created a confusion matrix to compare the false negative rate:



The false negative rate was ~18%, almost exactly the same as it was for the decision tree model.

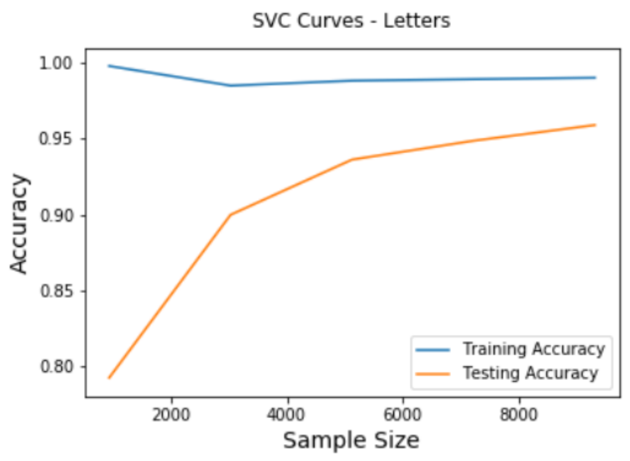
<u>Support Vecor Machines</u>

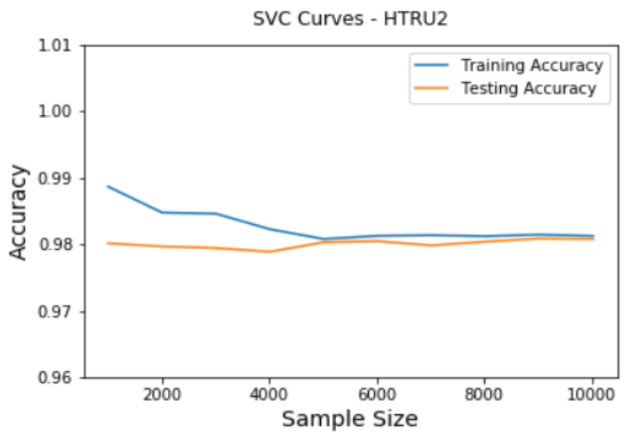




Both complexity curves showed an increase in training accuracy as gamma increased. When gamma was very small, both models had relatively low training and cross-validation accuracy. Low values of gamma constrain the complexity of the model, leading to bias and an inability to capture patterns in the data. Increasing gamma allows the model to be more complex and accurately model the training data, but this comes at a cost of increased variance. The increase variance means that the model becomes dependent on the training examples and can lead to overfitting a model that can no longer accurately classify new unseen examples. This is clear in the Letter Recognition complexity curve; cross-validation accuracy increases until gamma reaches a certain point that the model begins to overfit to the training data. This was the first model that seemed to show a risk of overfitting for the Letter Recognition data set.

For the SVM, I used a grid search to optimize hyperparameters. The accuracy of the HTRU2 model was optimized with gamma=5, and a polynomial kernel. The Letter Recognition model was optimized with gamma=.05 and a rbf kernal. It was interesting that as Gamma increased the Letter recognition accuracy showed a pretty clear decrease.

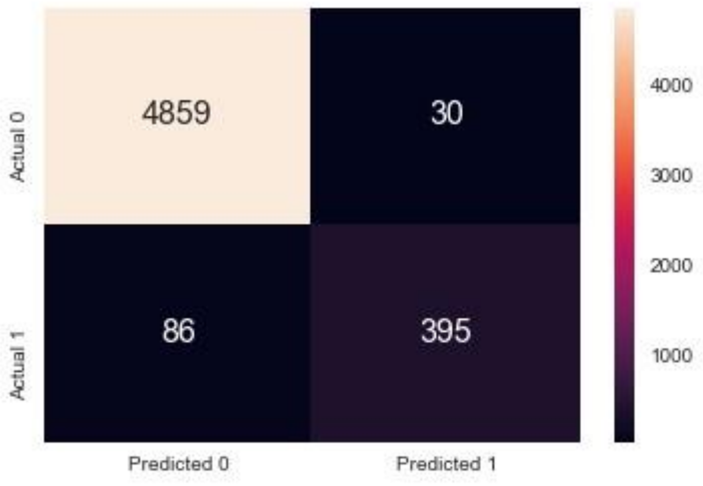




The HTRU2 learning curve was pretty standard - a decreasing training accuracy curve and an increasing testing accuracy curve that appeared to converge. This convergence, along with a high level of accuracy indicated that the SVC was a good model for this data. Similarly, the letter recognition model increased accuracy as it was able to use more training data, and the testing accuracy looked to approach the training accuracy.

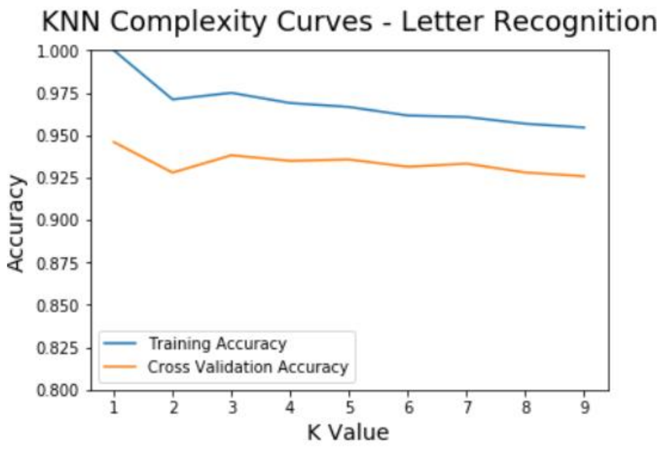
Accuracy

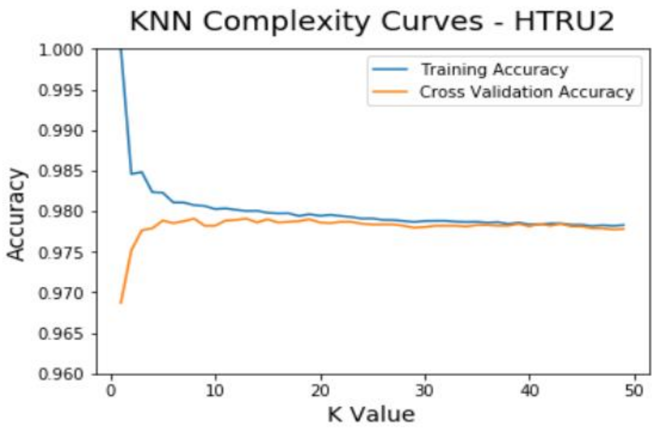
The SVC model classified 97.1% of the letter recognition test set correctly. The SVC model was able to correctly classify 97.8% of the test records for the HTRU2 dataset. However, the false negative error remained at 17.9%.



K-Nearest Neighbors

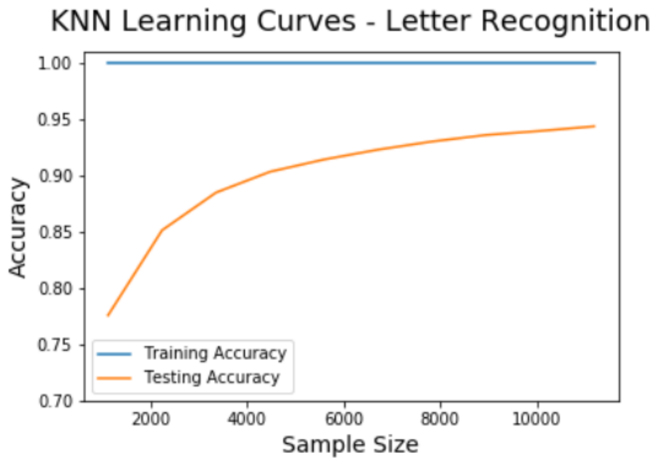
The final model used was a K-Nearest neighbors (KNN) model. The KNN model attempts to classify a vector by finding the "closest" k data points and observing the distribution of classes among those points. In this algorithm k is a hyperparameter which can be changed. If k is very small (for example 1) then the algorithm can model training data exactly, or almost exactly. This results in a high bias model that overfits the training data, and may be a poor model on new unseen test records. Below are complexity curves, showing how accuracy on training and validation sets is affected by value of k.

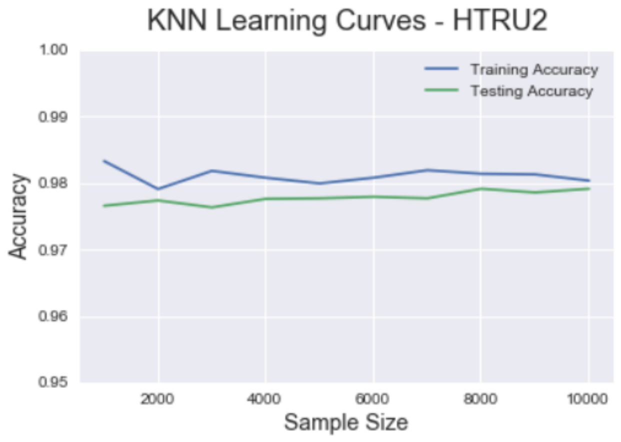




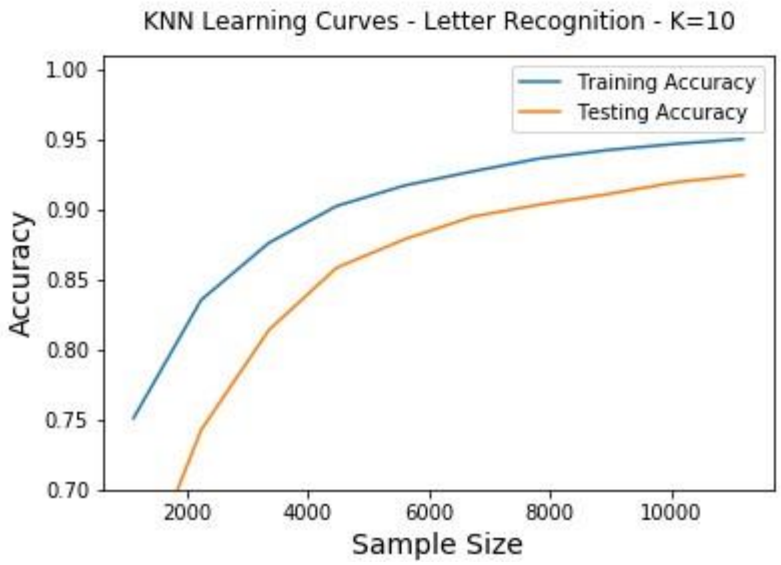
As expected, both models could perfectly model the training data when k=1. In this model, each training point is matched to the closest point which is in fact itself. Unless there is noise in the data (for example two records with identical attributes but different classes), k=1 will perfectly classify training data. The cross validation accuracy scores showed differences; the letter recognition accuracy appeared to be at its highest when k=1, and then decrease for larger values of k. The HTRU2 curve showed an increase in accuracy as k increased, but at some point began to decrease (or at least no noticeable increase). Using a grid search method with 5-fold cross validation, I determined that the optimal values of k for the HTRU2 and Letter Recognition set were 8 and 1, respectively.

Learning Curves





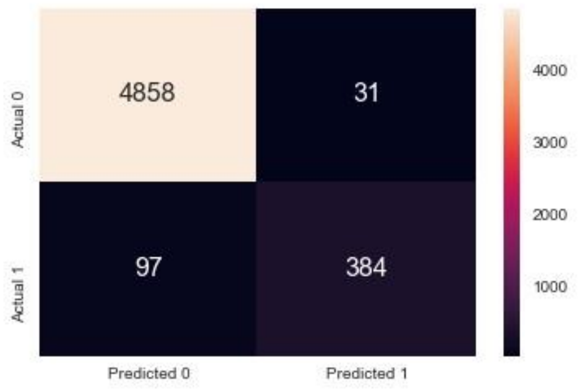
The HTRU2 dataset exhibited an increase in accuracy as more samples were added. The training accuracy was highest with a small sample, an expected result since a model can easily fit to classify a small set of training examples correctly. As training sample size increased, accuracy fluctuated but there appears to be a downward trend. The learning curve for the letter recognition model showed some differences. The accuracy on the test set increased until it appeared to level off around 95% accuracy. However, the accuracy on the training set remained at 100%, no matter how large the training set became. While at first this looked like an unusual result, it was in fact due to the hyperparameter of k=1. Since the model classifies each training point by observing the class of the closest training point (the point itself), 100% accuracy on the training data is expected. To observe this result further, I plotted a second learning curve, this time setting k=10.



The letter recognition learning curve with k=10 did not produce the results I was expecting. I had expected the training accuracy to decrease as sample size increased, because in general it is harder for a model to accurately classify a larger set of training examples once the algorithm is limited by a larger k value. I attributed this to the fact that examples of a class must have very similar attribute values. For example, if the training size was 10, then each record in the training set would be classified as whichever class (letter) appeared most frequently in that 10 record set - clearly not an accurate way to classify those 10 records. The model cannot begin to be accurate until it has seen some records of each class. Supposing the training size was increase to 260 (and assuming there were an equal number in each class) then I would expect the algorithm would be very accurate since of the 10 records closest to a given training example, I would expect the plurality of the records to be the same class as the training example. The learning curve indicates that even if the training size is greatly expanded, this result still appears to hold true. I concluded from this that there is little "noise" in the data, and that members of each class had attribute values that were very "close" to each other. This is likely one reason why the KNN model was very accurate, even when tested against unseen examples (results below).

Accuracy

The letter recognition KNN model was able to predict ~95.5% of test records correctly. The HTRU2 KNN model classified ~97.6% of test records correctly. The confusion matrix again showed a much lower rate in predicting legitimate pulsars, predicting ~79.8% of positive test cases correctly.



Conclusion

The final accuracy results for each algorithm when classifying unseen test data are below:

HTRU2

MODEL	ACCURACY	TIME TO TRAIN (S)
Decision Tree	.977	37.6
Artificial Neural Network	.974	60.7
Boosting	.977	35.8
Support Vector Machine	.978	63.4
K-Nearest Neighbors	.976	88.1

Letter Recognition

MODEL	ACCURACY	TIME TO TRAIN (S)
Decision Tree	.868	16.0
Artificial Neural Network	.795	206.5
Boosting	.872	8.3
Support Vector Machine	.971	1094.1
K-Nearest Neighbors	.955	59.2

**Note that for ANN, a random search was used rather than an exhaustive grid search.*

Every model built to identify legitimate pulsars had a very high accuracy when tested against unseen test data in the HTRU2 dataset. One reason for this was that this was binary classification, where the target classes were very skewed with only ~9% of records positive. This resulted in a high baseline for models - an algorithm that simply predicted that any data point was not a legitimate pulsar would be expected to be correct ~91% of the time. Because of this, I also compared the false negative rate of each model. The SVC, Decision Tree, and Boosted Decision tree models each had a false negative error around 18%. This means that even using the best algorithm would result in nearly one fifth of legitimate pulsars being missed. It would be interesting to see if new or further tuned models could reduce this error.

One conclusion that I drew from my analysis was that the letter recognition dataset was far less susceptible to overfitting than the HTRU2 dataset. For the letter recognition set, my decision tree model was optimized with a max depth of 44, and the KNN model was maximized with k=1. Typically a large max depth and a low k mean that a model has lots of freedom to conform to a specific set of training examples. This can possibly result in high variance, and poor accuracy when predicting new examples. In this case, I did not observe this high variance and concluded that each capital letter is so unique in terms of the 16 attributes that models do not easily overfit. The only model that showed overfitting as complexity increased was the SVC. In contrast, the HTRU2 models overfit the data once complexity increases too much using the decision tree, KNN, boosting, and SVC algorithms. The only algorithm that did not show a tendency to overfit for the HTRU2 model was the ANN, but perhaps if more hyperparameters were tested (for example adding more hidden layers) then overfitting would be more apparent.