Supervised Learning

1. DATASETS:

White Wine quality dataset from the UCI ML repository: It has 11 features/attributes to decide the quality of the wine on a scale of 0 to 10. I have designed this as a classification problem, where a supervised learning algorithm would learn from the features and predict the "quality" class. I found this dataset interesting because it provides a complexity through a wide number of attributes (11) to evaluate the algorithms as well.

HTRU2 dataset from the UCI ML repository: It consists of samples of detected pulsars as well as samples which are just random noise. It has 8 attributes which classify the emission as a pulsar or not pulsar. I have designed it as a classification problem, where the model predicts the target class as 0(not a pulsar) or 1(pulsar). I have always envisioned machine learning in the field of physics and astronomy which has humungous amount of data.

3. TRAIN/TEST SPLIT

I split train/test set by 80-20. Here I have used the 80% data for training, validating, tuning the model and kept aside 20% of data for testing. Finally testing on this data will give us an actual idea of how our model will probably perform on the real-world unseen data.

4. METRICS

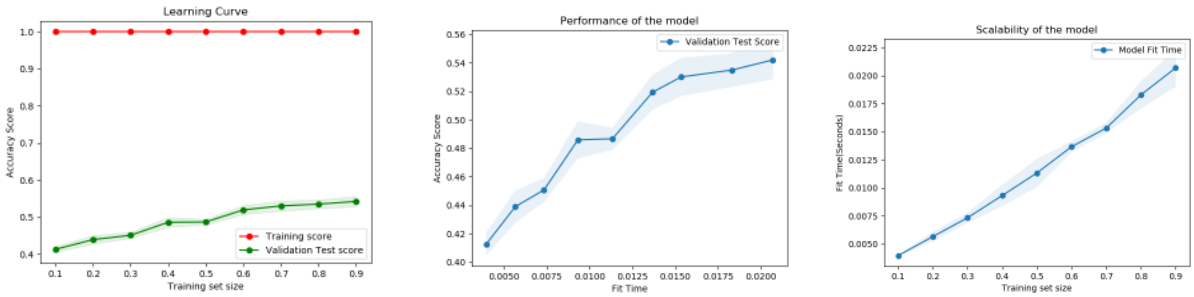
I have used the sklearn's "accuracy_score" metric to evaluate the algorithms. For the classification problem, the accuracy score metric computes if the predicted class matches the exact corresponding class expected.

5. SUPERVISED LEARNING ALGORITHMS

5.1. Decision Trees

I started out with the default parameters of the DecisionTreeClassifier and plotted the learning curves for various training sizes (from 0.1 to 0.9). I used K-fold cross-validation using 3 folds. This method is useful to evaluate our model on the bias-variance trade-off. It reduces bias as in, most of the data gets used to train the model, it also reduces variance as most of the data is used for testing as well, over iterations.

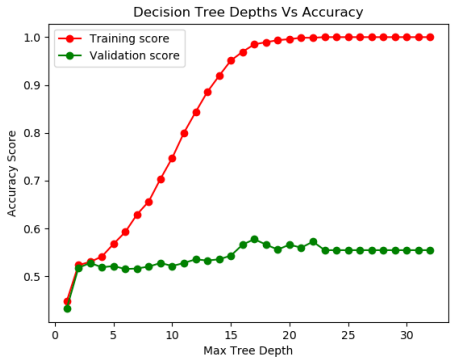
Wine Dataset Decision Trees before pruning:



From the above graphs, we can deduce that there is some form of overfitting going on. Overfitting is a phenomenon where the model is too closely fit to the data points, wherein the model learns even the random noise in the data. This leads to the model performing too well on the training data but is not able to perform as well on the validation set. Here, the training accuracy score is very high (around 1.0) compared to the validation test score (0.4 to 0.5).

Pruning the decision trees is a form of reducing the size of the decision trees. I decided to use the "max_depth" parameter of the decision trees. The depth of the tree is when the decision tree keeps splitting until it captures all the data points. By tuning this parameter, we can achieve reduce the tree size considerably.

I used "max_depth" = 1 to 32. The below graph beautifully illustrates how the training score increases as depth increases. This is the classic case of overfitting. But the validation score is very low at around 0 at max_depth = 1 then follows a significant climb after that, but the curve almost stabilizes around 0.5 to 0.6.

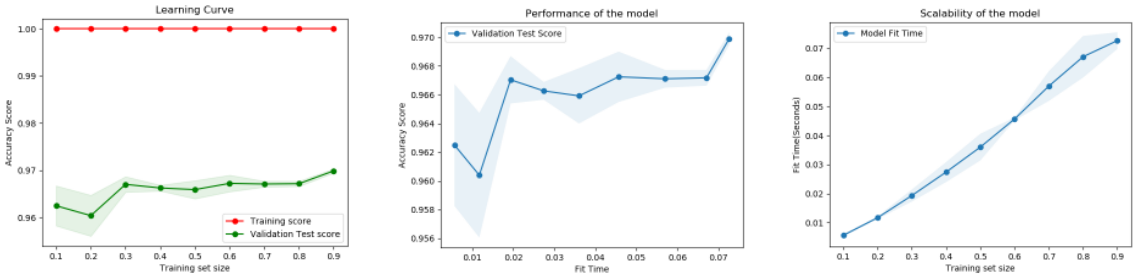


After max_depth = 1, the validation scores were similar, so I chose max_depth = 11, which had a validation score=0.528 and train score of 0.80.

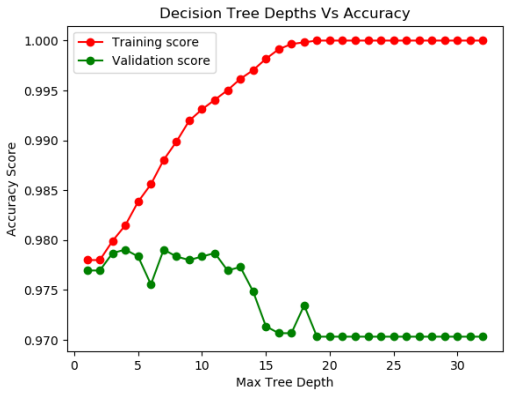
Wine Dataset Decision Tree after pruning:

After pruning, the learning curves of the training set and validation set approach each other which is a better model with better bias-variance trade-off than the one we started out with. Even the performance and scalability of the model is better now. The final accuracy score on the test data we kept aside was 0.6020408163265306, which is even better than the validation score.

HTRU2 Dataset Decision Tree before Pruning:

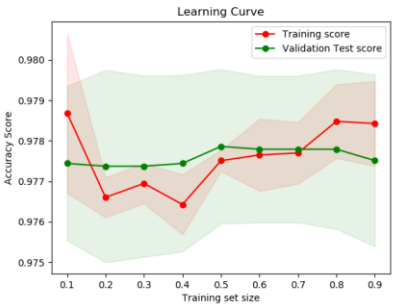


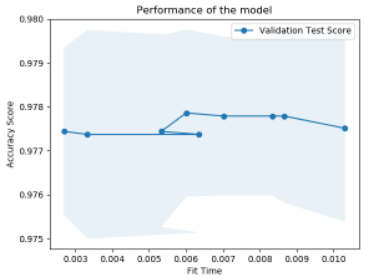
This dataset fascinated me in a way that there were very high validation scores around 0.97. This is because of the kind of data points in the set. I investigated to find that there were about 10% of data points for class 1(pulsar) and the remaining 90% where class 0(not pulsar). The not pulsar data points must be clearly distinguishable from the pulsar points and even if the pulsar is not classified correctly, that doesn't affect the accuracy score. This is one such example where an imbalanced dataset can easily mislead a ML algorithm.

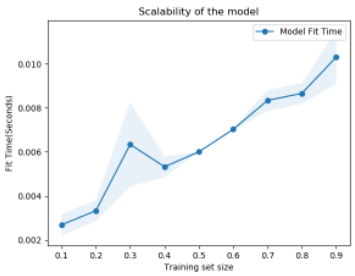


Here, I decided to choose the max_depth = 1, since a simple model is enough to capture the pattern in the data.

HTRU2 Dataset Decision Tree after Pruning:





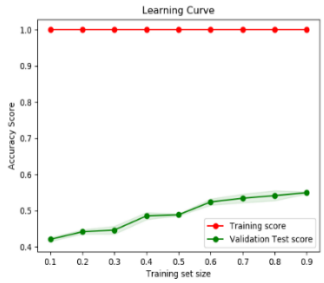


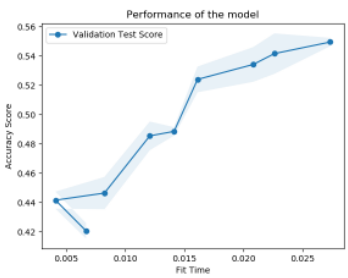
After pruning we can that the training and test scores are almost the same.

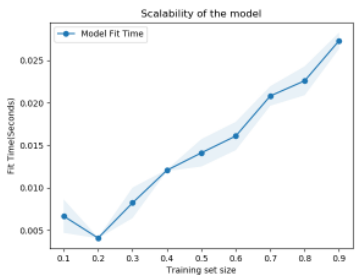
5.2. Boosting:

I used sklearn's AdaBoostClassifier (no estimators =10)boosting method on the DecisionTreeClassifier.

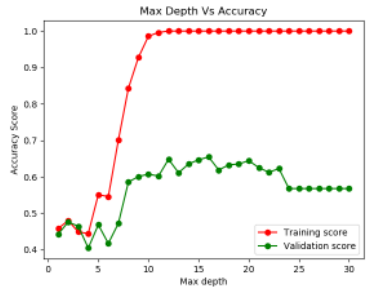
Wine Dataset Boosting:



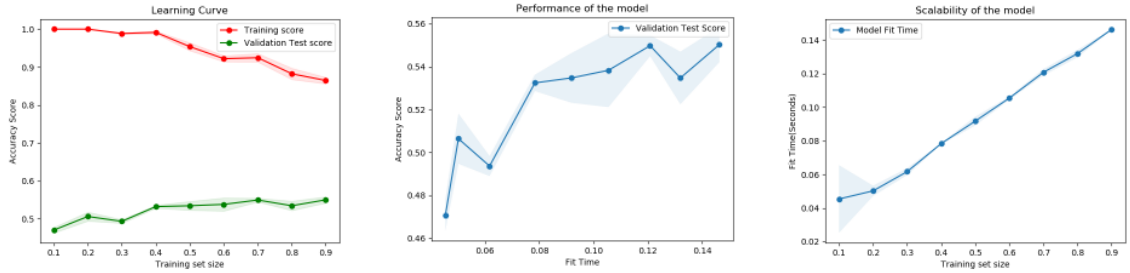




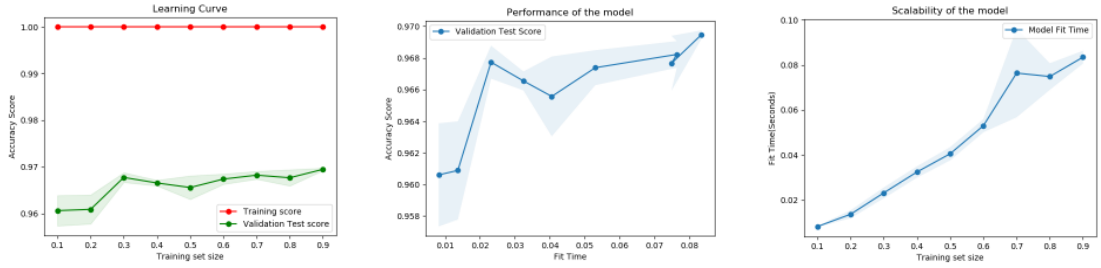
I used max_depths values 1 to 30. We can see the same pattern here, as the depth of the tree increases train score significantly increases. I chose max_depth=8, which gives good validation score, but after that it remains almost same. That is a good trade-off to choose.

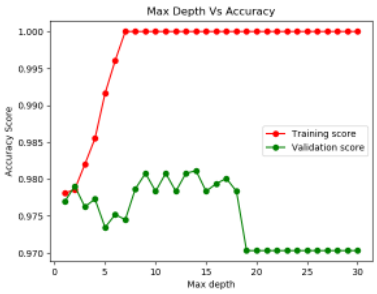


Wine Dataset Boosting after pruning: We can see train and validation scores approach each other after pruning.



HTRU2 dataset Boosting: I don't think this dataset needs boosting at all, since it performed well on a simple decision tree itself. It yields same results, only with increased fit times due to the ensemble.

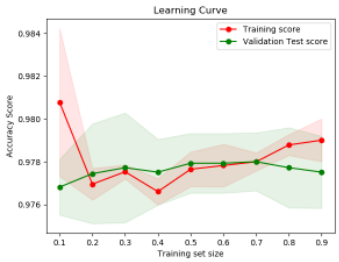


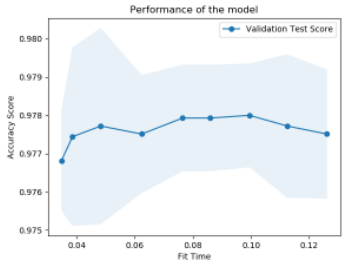


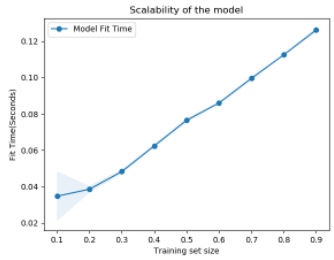
With increased depth the train scores increase rapidly, causing overfitting, which we don't want. I decided to go with, max_depth=1, which yields pretty good results.

We can see below how the performance of the model is almost stable across fit times. I achieved a final accuracy score of 0.974301675977653 on the test data.

HTRU dataset Boosting after pruning:



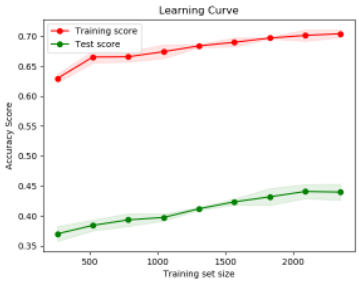


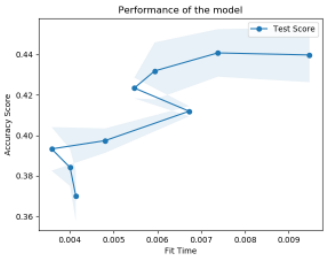


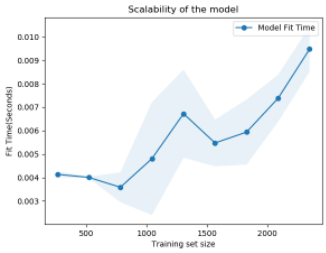
5.3. KNN

I have used the sklearn's KNeighborsClassifier, starting with k=3.

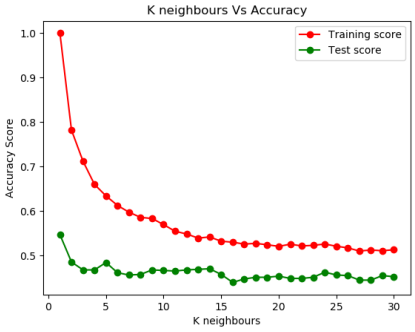
Wine Dataset KNN before tuning:





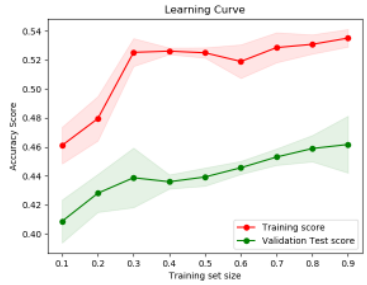


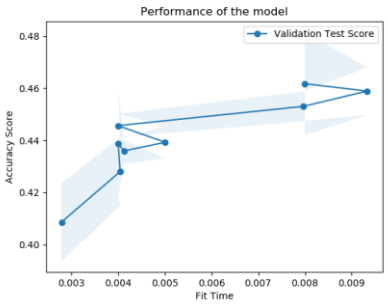
Unlike our previous algorithms, the train score here is around 0.7, because of low k value. This number of neighbours are not enough to predict correctly even on the train set. The validation score is also obviously on the lower end of 0.4. Also, for KNN the fit times are very low because there is no learning involved and the neighbour data points are referenced when predicting.

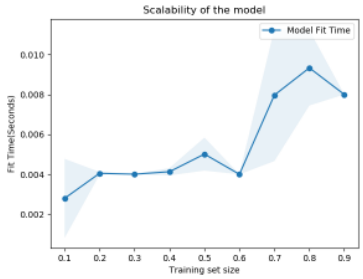


I used the k neighbours parameter to tune the model from 1 to 30. We can note here that at k=1, the train score is 1, which mostly means overfitting. As k increases, the train score drops, whereas validation score is almost stable. I chose to use k=14, at which I get a good tradeoff of train and validation scores.

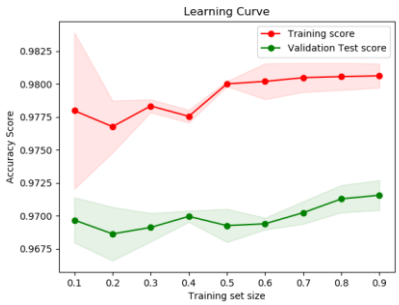
Wine dataset KNN after tuning: Compared to before tuning, we see the train and validation scores are almost similar. I achieved a final accuracy of 0.4571 on the test data which is similar to what we got from our model in training.

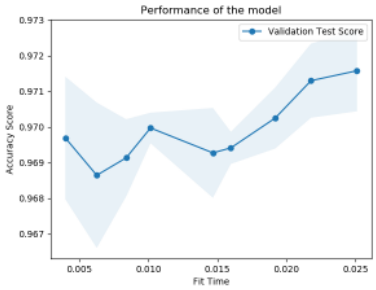


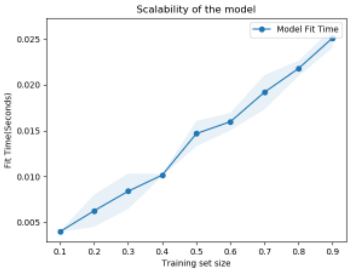




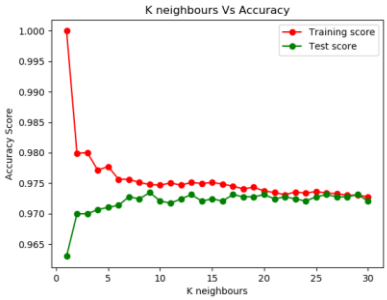
HTRU dataset KNN:



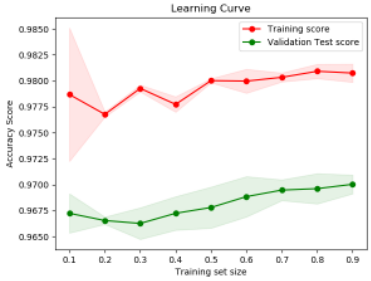


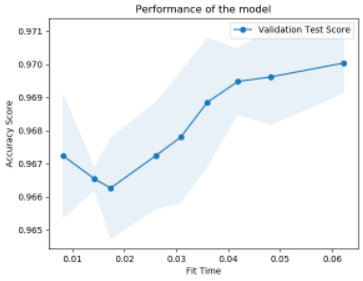


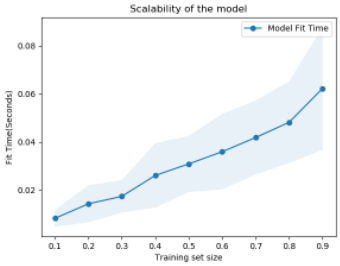
With k=3, the train and validation scores are high. This is again mainly due to the imbalance in the dataset.



As the neighbours increase train score drops and both curves overlap each other. I chose k=2.

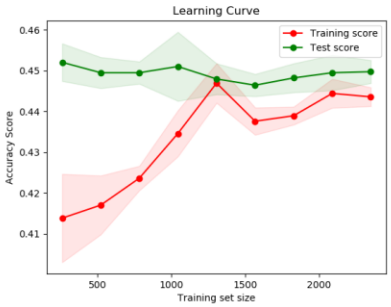


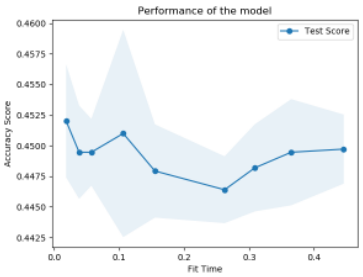


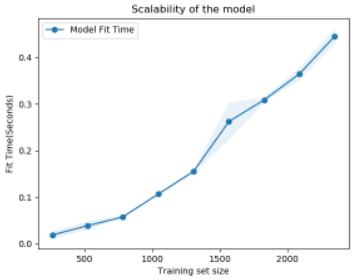


5.4. SVM

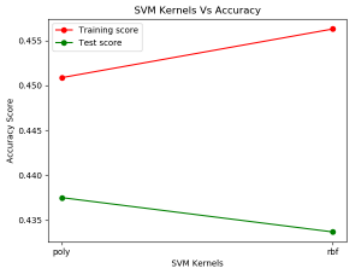
Wine dataset SVM (poly) kernel: For the poly kernel, validation scores are higher than train scores, maybe because the validation sets contained easier samples.



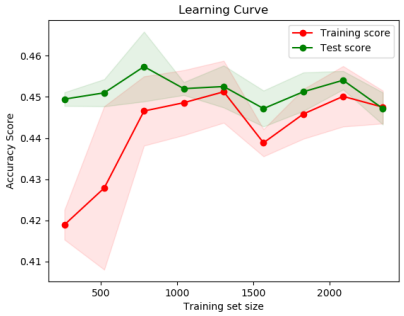


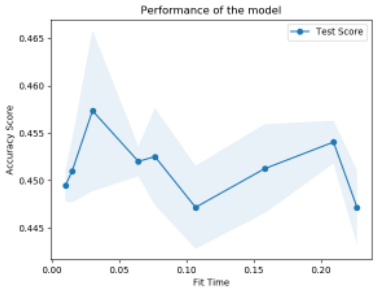


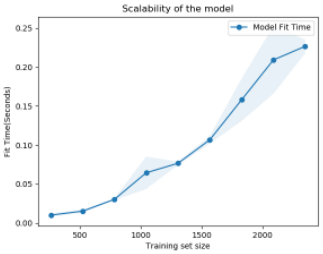
Kernels (poly vs rbf):



Wine dataset SVM(rbf) kernel:



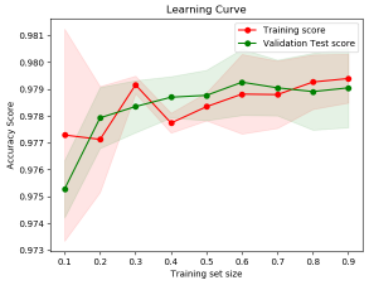


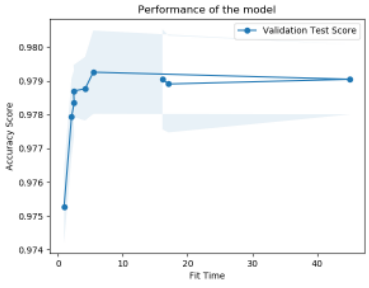


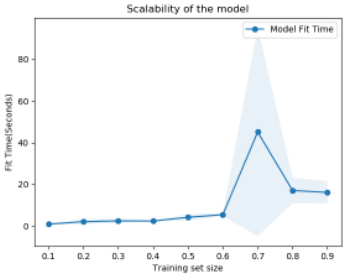
I chose the rbf kernel over poly, since it has better fit time, so this model will be easily scalable real-time.

HTRU dataset SVM (linear) kernel:

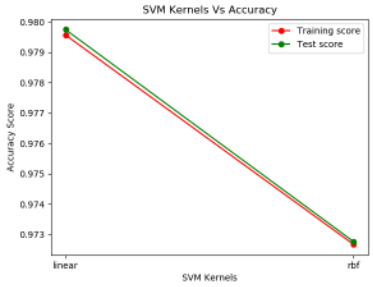
I started with linear kernel of the assumption that, since it is 2 classes, the data might be linearly separable by the model.



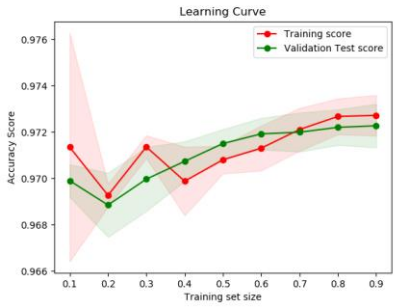


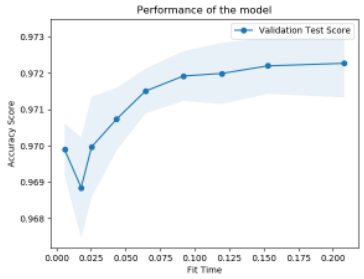


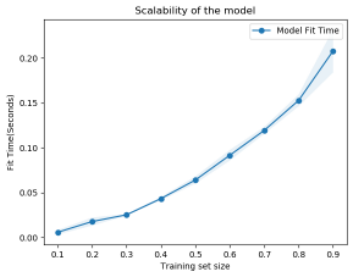
Kernels (linear vs rbf):



HTRU dataset SVM (rbf) kernel:





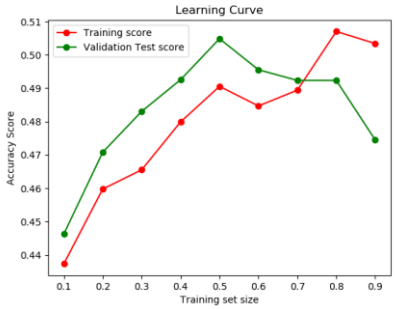


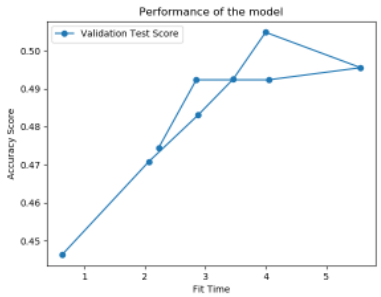
I chose rbf kernel over linear,they had same accuracy, but rbf had better fit time of 0.2seconds compared with linear 40 seconds.

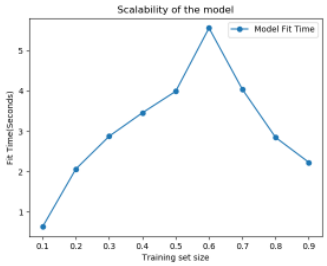
5.5. Neural Networks

I used the sklearn's MLPClassifier with 3 hidden layers of size 10.

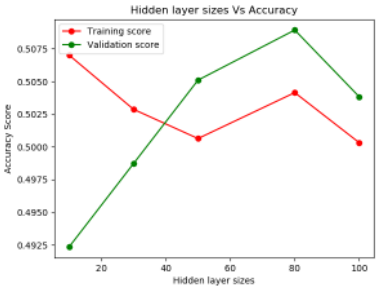
Wine Dataset NN:



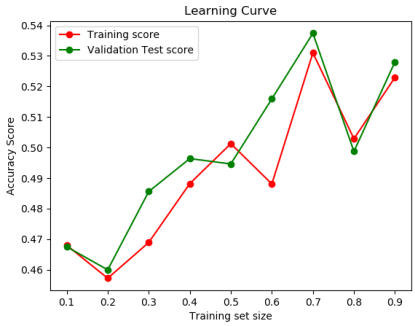


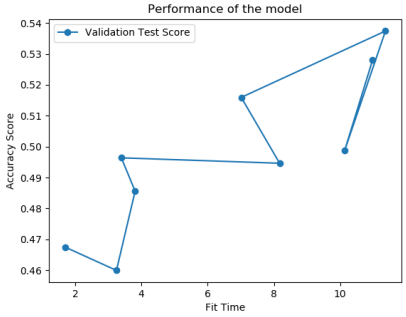


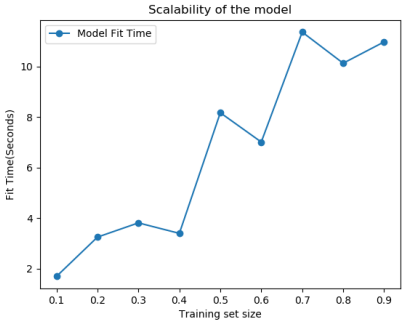
I think NN is not a good model for this dataset, we get very low train and validation scores with high fit times as well. I decided to tune the layer size with values 10, 30, 50, 80, 100. But even with 100, there is not much difference in the score.



I settled with layer size 30, where train and validation scores were close.

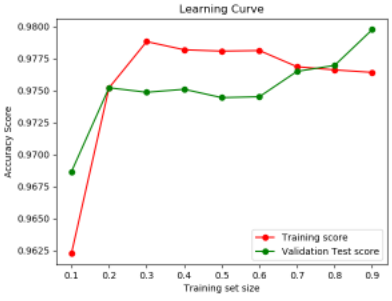


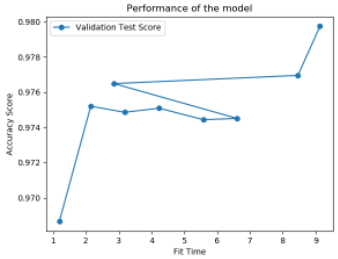


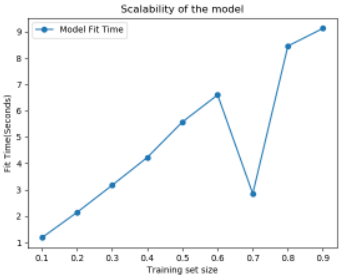


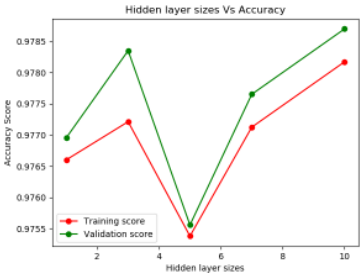
HTRU dataset NN:

I started with 3 hidden layers of size 1, since this dataset needs a basic model.









There was not much changes increasing the layer size, so I settled with size 3, which yielded more or less similar performance and accuracy.

6. COMPARITIVE ANALYSIS

For the wine dataset, Decision trees seem to be the best algorithm comparitively. It has the highest accuracy score with good performance and scalability with low fit time as well. Though we can see from the table that the accuracy scores vary by small numbers, the fit time plays a key role here. Neural networks have the highest fit times of 11 seconds, which is definitely not optimal for the problems here.

Algorithm	Test Accuracy Score (Wine data)	Fit time (seconds)
Decision tree	0.60204	0.017s
Boosting	0.55306	0.13s
KNN	0.4571	0.009s
SVM	0.463265	0.20s
NN	0.515306	11s

For the HTRU dataset, all the algorithms yield similar accuracy, but decision trees yield the lowest fit time. This can also be because of the imbalance in the dataset. But the problem in this case was to identify pulsar and non-pulsar emissions. The intent might be mostly to flag the noise, and rest of the portion of data might be re-analysed to confirm the presence of a pulsar. What we can understand here is that such complex datasets and problems do exist in the real world and we need to choose model that best fits not only the dataset but the problem we are tring to solve.

Algorithm	Test Accuracy Score (HTRU data)	Fit time (seconds)
Decision tree	0.974301	0.009s
Boosting	0.976256	0.13s
KNN	0.968156	0.05s
SVM	0.972625	0.16s
NN	0.975139	11s

7. CITATIONS

OlteanuAlex, Alex, et al. “Tutorial: Learning Curves for Machine Learning in Python for Data Science.” Dataquest, 30 July 2019, www.dataquest.io/blog/learning-curves-machine-learning/.

“Plotting Learning Curves¶.” *Scikit*, scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html#sphx-glr-auto-examples-model-selection-plot-learning-curve-py.

Ray, Sunil, and Business Analytics and Intelligence. “Understanding Support Vector Machines Algorithm (along with Code).” *Analytics Vidhya*, 3 Sept. 2019, www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

Robinson, Scott. “Introduction to Neural Networks with Scikit-Learn.” *Stack Abuse*, Stack Abuse, 6 Feb. 2018, stackabuse.com/introduction-to-neural-networks-with-scikit-learn/.

Harrison, Onel. “Machine Learning Basics with the K-Nearest Neighbors Algorithm.” *Medium*, Towards Data Science, 14 July 2019, towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.