Datasets

The datasets are downloaded from the link https://archive.ics.uci.edu/ml/index.php which was posted by a student in Ed. The two datasets chosen are Car Evaluation and Adult. The car evaluation dataset was chosen as it was interesting to see the effects of the various parameters across the general population. However, the data size was a little unsatisfactory. The adult dataset was chosen to look at the impact of race and sex on income earned for the same occupation. This dataset also had a relatively large number of records to play around with. For both datasets, the features were limited to be able to run effective experiments.

1) Car Evaluation [1]
   The dataset is derived from a simple hierarchical decision model originally developed for the demonstration of DEX, M. Bohanec, V. Rajkovic: Expert system for decision making. Sistemica 1(1), pp. 145-157, 1990.).
   This dataset will be referred to as dataset 1.
2) Adult [1]
   The dataset extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0)).
   This dataset will be referred to as dataset 2.

Dataset 1 does not have as good results in the experiments as dataset 2 due to the lesser source data available for processing.

Decision Tree

The following links were used as references to code for the decision tree model:

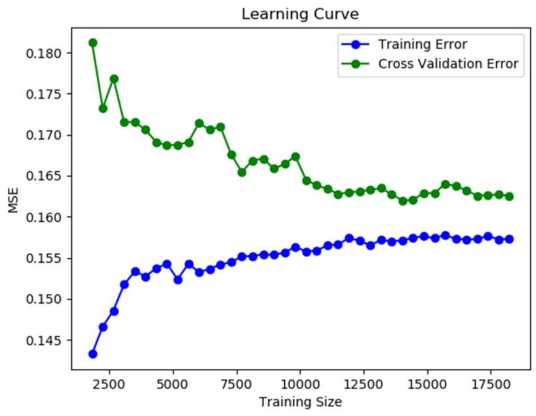
1) https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
2) https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.learning_curve.html
3) https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
4) https://www.projectpro.io/recipes/plot-learning-curve-in-python
5) https://www.dataquest.io/blog/learning-curves-machine-learning/
6) https://ai.plainenglish.io/hyperparameter-tuning-of-decision-tree-classifier-using-gridsearchcv-2a6ebcaffeda
7) https://towardsdatascience.com/how-to-find-decision-tree-depth-via-cross-validation-2bf143f0f3d6

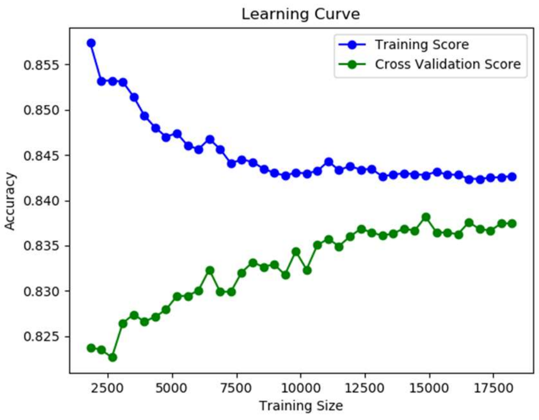
As the training sample increases, there is marked improvement in the accuracy of the model and a reduction in bias. This is captured in the graphs depicting the Learning Curves of the decision tree model. This corresponds to a lower mean squared error as captured in the middle two graphs with MSE in the y-axis. The impact of overfitting along with the quality of split by adding gini (gini impurity) and entropy (information gain) as the hyper parameter in the decision tree is captured in the Validation Curves.

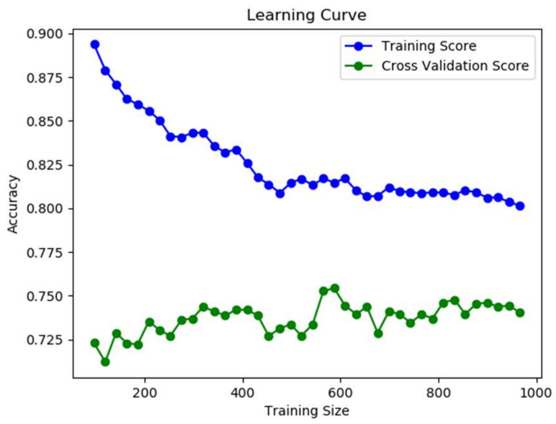
1) For dataset 1, the best two parameters are max depth of 7 and entropy as the criterion.
2) For dataset 2, the best two parameters are max depth of 7 and gini as the criterion.

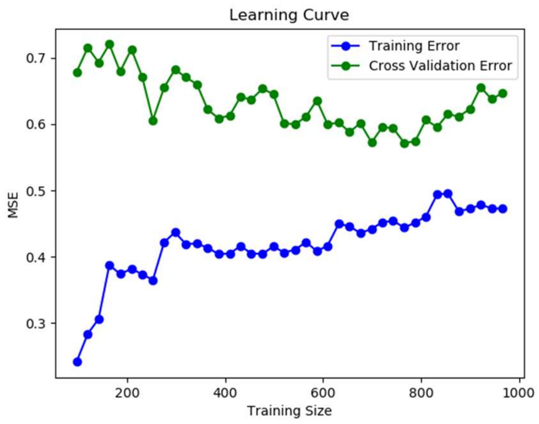
3) For tree depth < 7, the model underfits.
4) Tree depth > 7 it starts to overfit. The training accuracy increases a lot for greater tree depths.

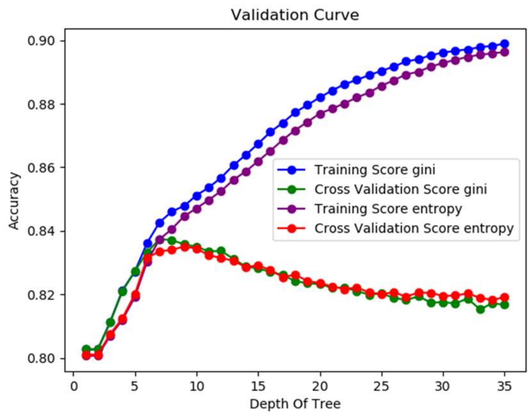
Dataset 2 **Dataset 1**

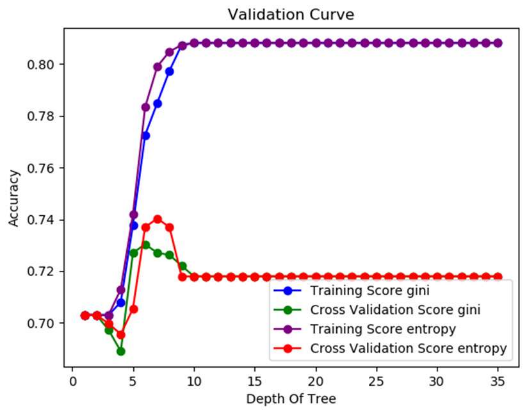












AdaBoost

The following links were used as references to code for the adaboost implementation:

1) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
2) https://www.datacamp.com/community/tutorials/adaboost-classifier-python

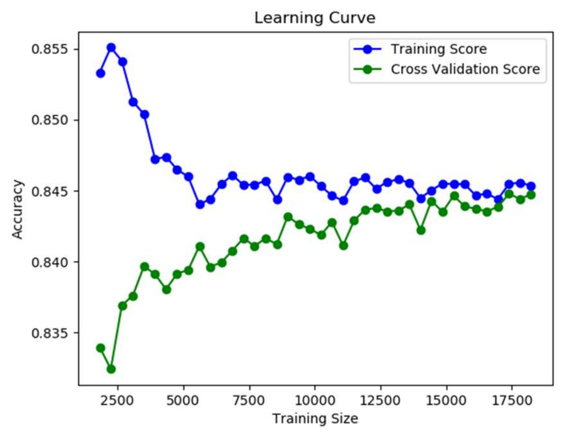
As the training sample increases, the accuracy of the training set decreases and that of the validation set increases along with a reduction in bias. Gradually, they converge. This is captured in the graphs depicting the Learning Curves of the model after being boosted via the Adaboost classifier. This corresponds to the lower mean squared error in validation and increasing mean squared error in training as captured in the middle two graphs with MSE in the y-axis.

The two parameters with respect to hyper tuning the model considered here are the number of estimators and the learning rate. The number of estimators determines the number of weak learners and the learning rate determines the contribution of each classifier. The base estimator is defaulted to the decision tree.

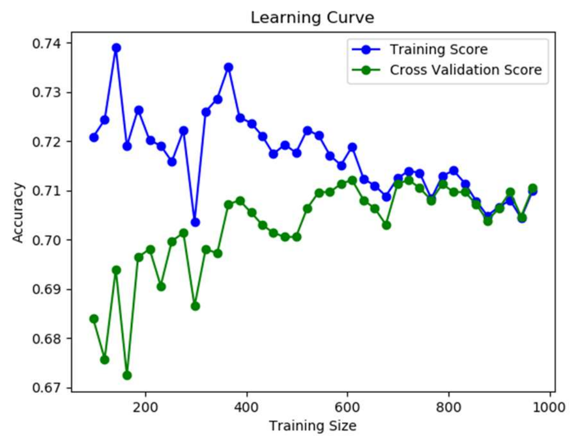
As the number of estimators increase, the model is prone to overfitting. This can be seen in the validation curve of dataset 1. It is interesting to note that in dataset 2, the accuracy starts to drop at estimators around 130. The graph does not reflect that of dataset 1 and am unable to plot the downward trend of accuracy for even higher number of estimators similar to dataset 1.

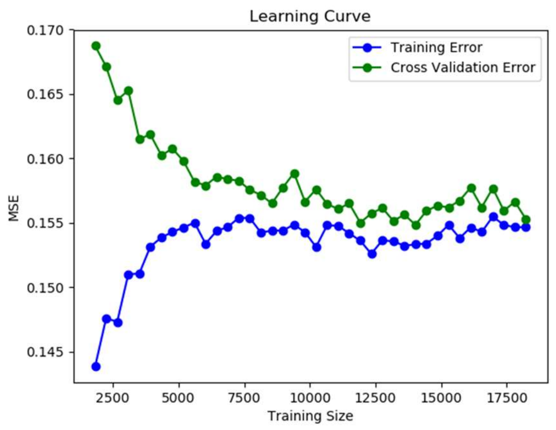
1) For dataset 1, the best two parameters are learning rate = 0.25 and number of estimators = 11.
2) For dataset 2, the best two parameters are learning rate = 1.5 and number of estimators = 128.

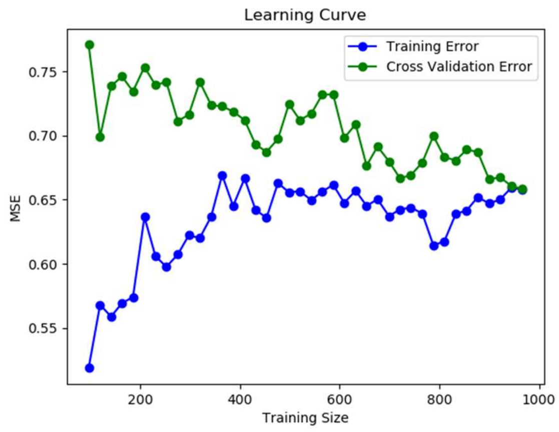
Dataset 2

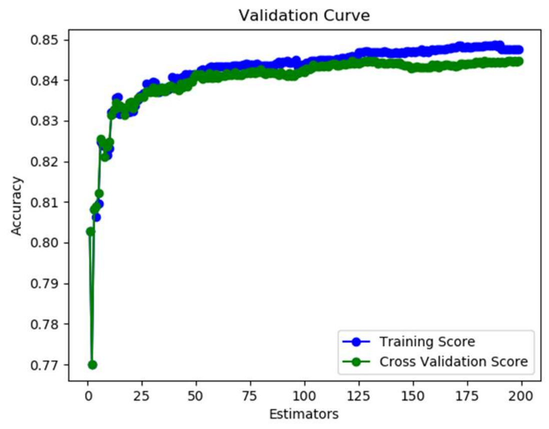


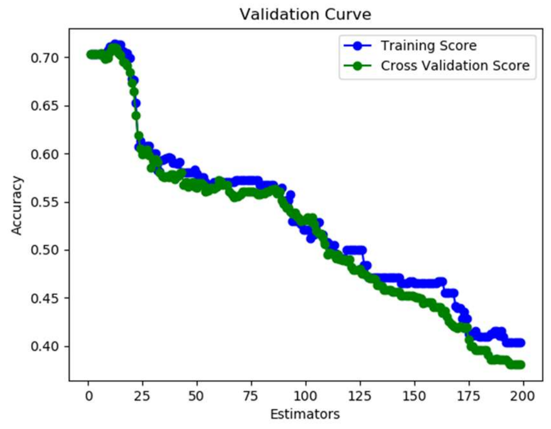
Dataset 1











Neural Network

The following links were used as references to code for the neural network MLP model:

1) https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
2) https://www.tensorflow.org/api_docs/python/tf/keras/wrappers/scikit_learn
3) https://www.kaggle.com/residentmario/using-keras-models-with-scikit-learn-pipelines
4) https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/
5) https://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/

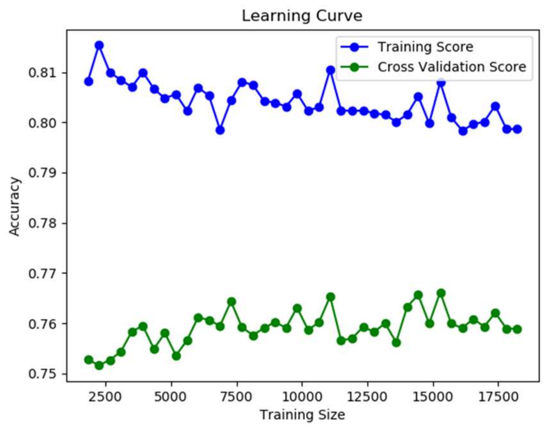
The MLP model is defined using the Keras wrapper for scikit learn. The number of input layers is set to the number of data points. The hidden layer size is defaulted to 100. The input and hidden layers are defaulted with relu activation and the output layer is defaulted to sigmoid activation. The output units are also set to 1.

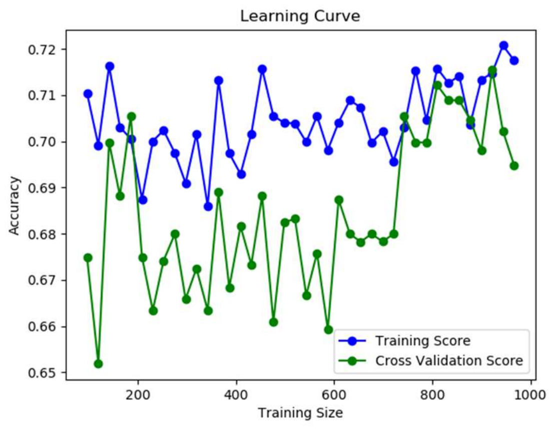
The accuracy of the validation set increases as the training sample increases. As the epoch increases, the losses are reduced for the training set. However, for the validation set, the loss decreases and then increases. This is captured in the graphs depicting the Loss Curves of the MLP model. The impact of increase in epochs vs accuracy is captured in the Validation Curve plots.

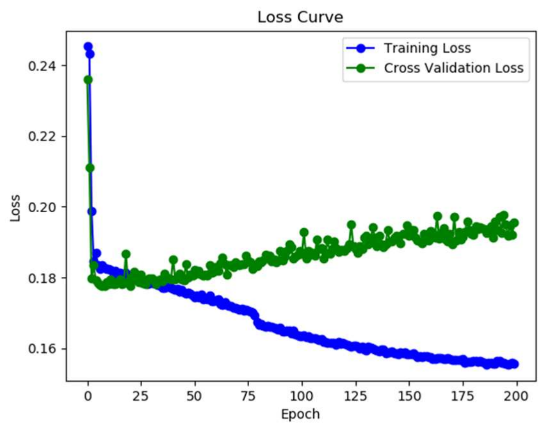
1) For dataset 1, the best two parameters are epochs = 20 and batch size = 80.
2) For dataset 2, the best two parameters are epochs = 30 and batch size = 100.

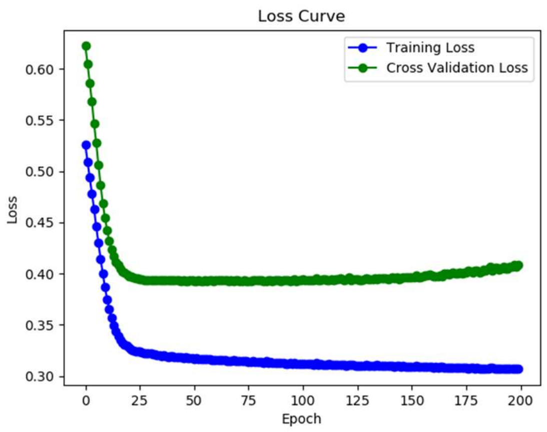
Dataset 2

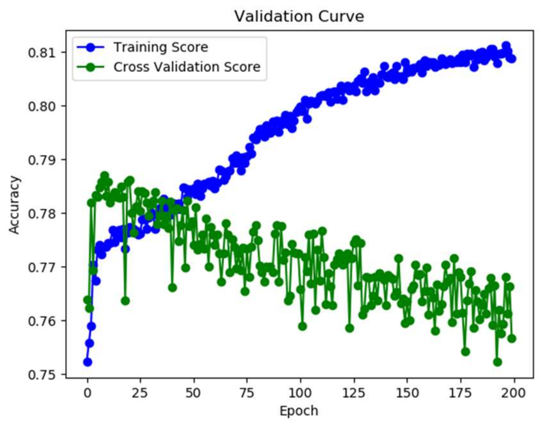
Dataset 1

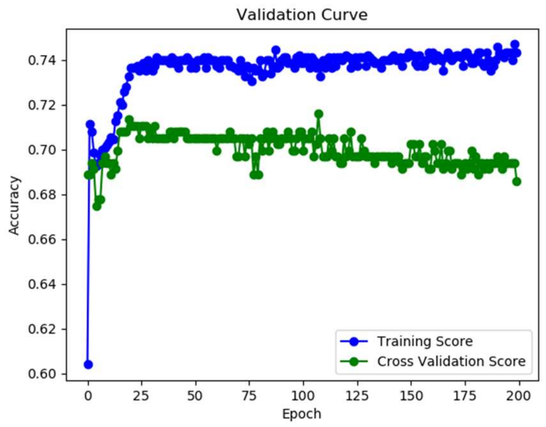












KNN

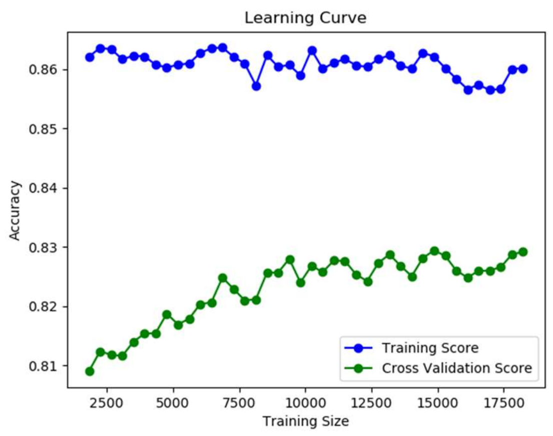
The following links were used as references to implement the KNN model:

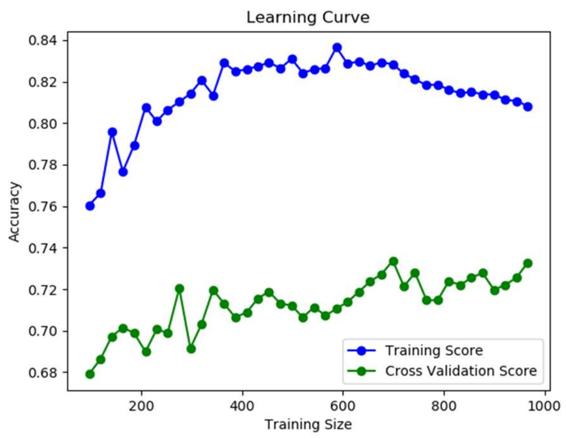
1) https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
2) https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn

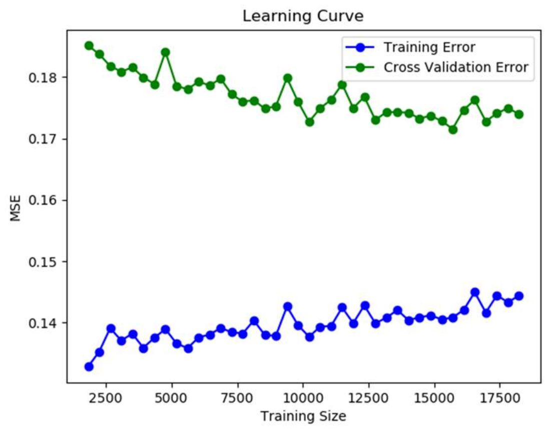
As the training sample increases, the accuracy score increases. This is shown in the Learning curves of both datasets. The means squared error reduces as well as the training sample increases and is shown in the middle two plots. The effect of increasing k is shown in the validation curves. The grid search function returned the best k as 1 for dataset 1 which doesn't seem to be right as it indicates overfitting.

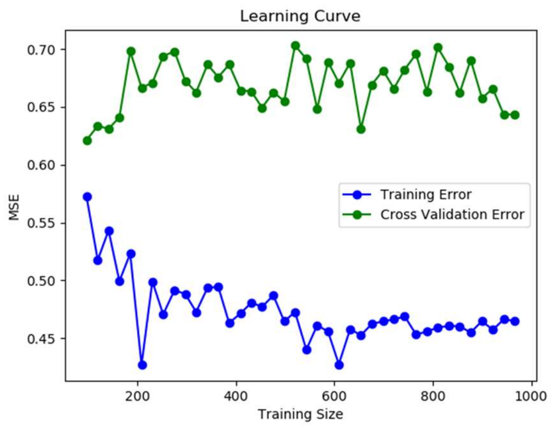
1) For dataset 1, the best two hyper parameters are k = 1 and p = 1
2) For dataset 2, the best two hyper parameters are k = 16 and p = 2

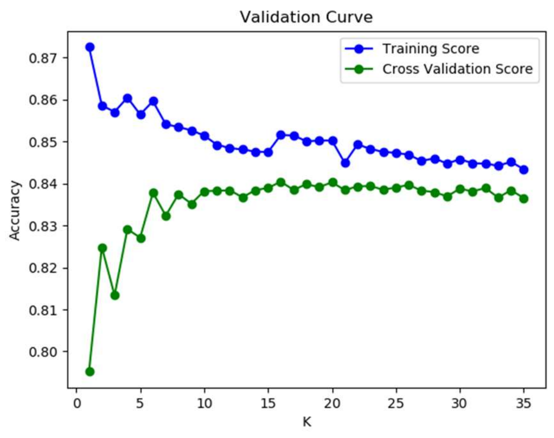
Dataset 2 **Dataset 1**

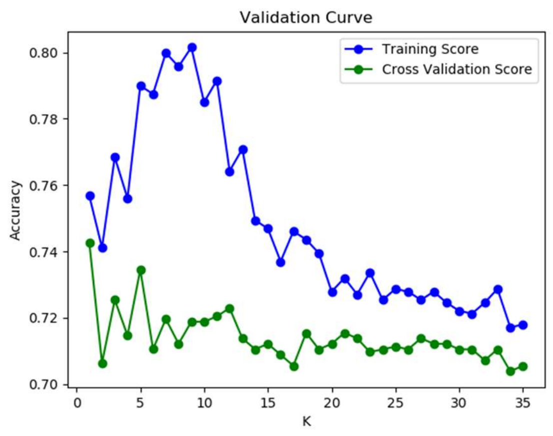












SVM

The following links were used as references to implement SVM model:

1) https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
2) https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python

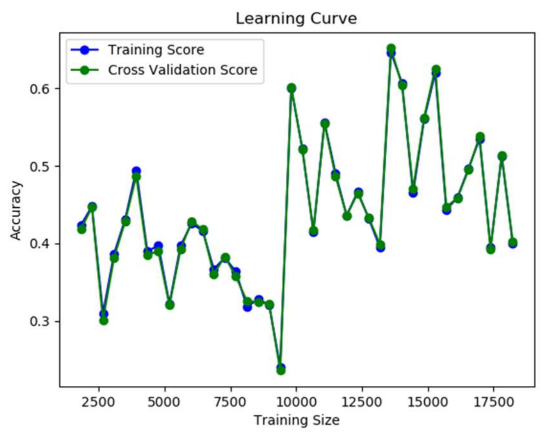
Kernel = Linear

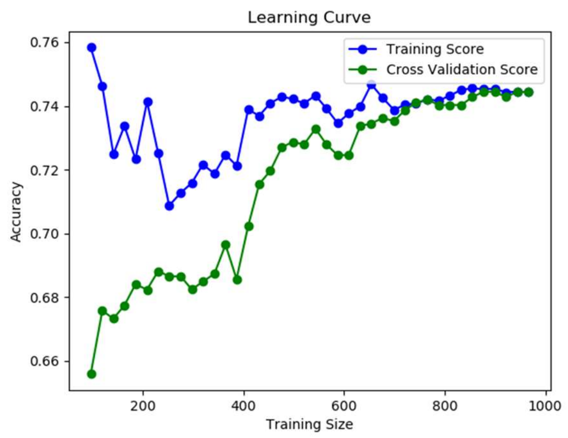
As the training sample increases, there is marked improvement in the accuracy of the model and a reduction in bias for dataset 1. For dataset 2, the accuracy score is the almost the same for both the training and validation set and seems to be all over the place. This is captured in the graphs depicting the Learning Curves of the model. This corresponds to a lower mean squared error as captured in the middle two graphs with MSE in the y-axis.

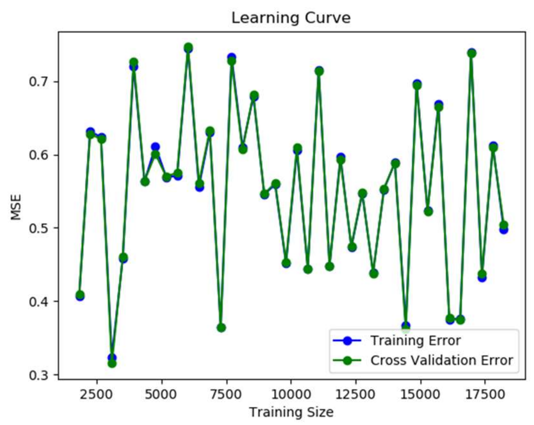
There are two validation curves for dataset 2. One with a max iteration of 5000 and another with a max iteration as derived using Grid Search. To generate the 5000 plot, the value of iter will need to be set manually in the runCurves function in SVMLinear.py

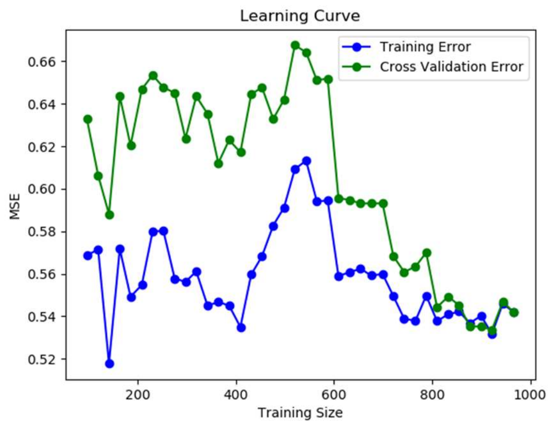
1) For dataset 1, the best two parameters are max iteration = 1000 and C = 0.75.
2) For dataset 2, the best two parameters are max iteration = 10 and C = 0.25.
3) For both datasets, the best class weight option is balanced.

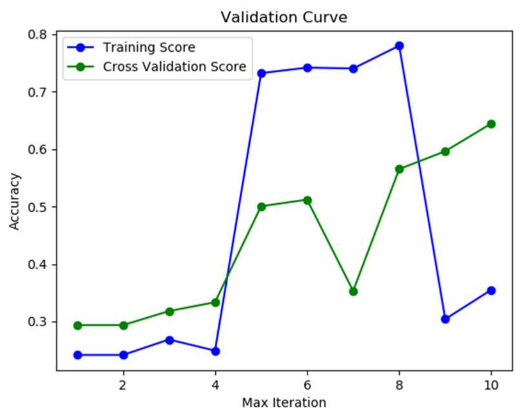
Dataset 2 **Dataset 1**

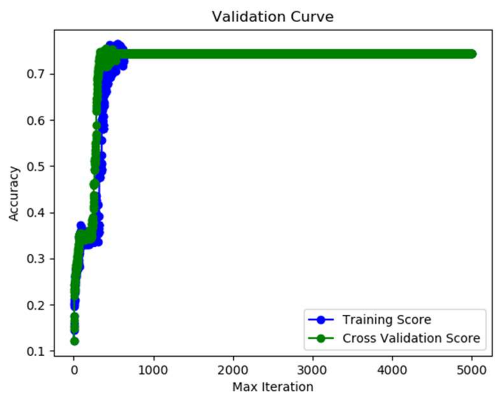

Learning Curve
Training Score
Cross Validation Score
Accuracy
Training Size

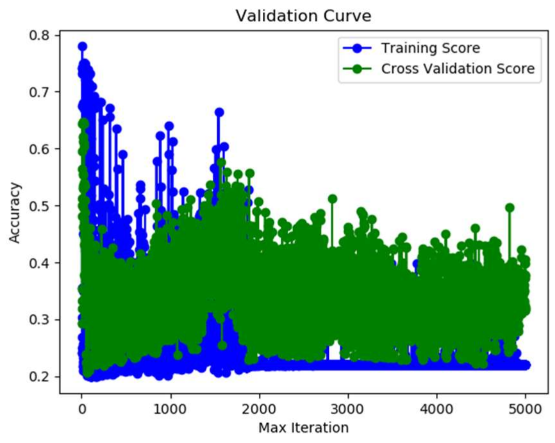

Learning Curve
Training Score
Cross Validation Score
Accuracy
Training Size


Learning Curve
Training Error
Cross Validation Error
MSE
Training Size


Learning Curve
Training Error
Cross Validation Error
MSE
Training Size


Validation Curve
Training Score
Cross Validation Score
Accuracy
Max Iteration


Validation Curve
Training Score
Cross Validation Score
Accuracy
Max Iteration



Kernel = rbf

As the training sample increases, the accuracy of the model increases and a reduction in bias for dataset 1. For dataset 2, the accuracy score is the almost the same for both the training and validation set and seems to be all over the place. This is captured in the graphs depicting the Learning Curves of the model. This corresponds to a lower mean squared error as captured in the middle two graphs with MSE in the y-axis.

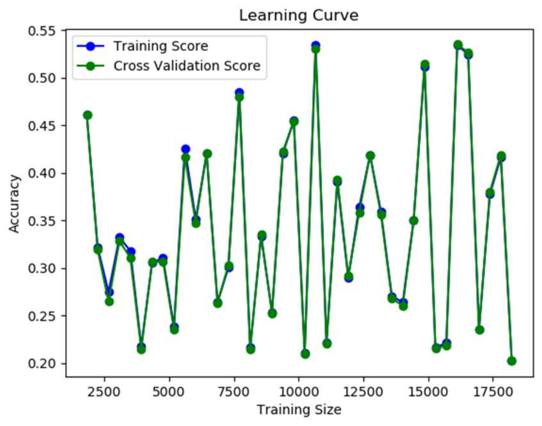
There are two validation curves for dataset 1. One with a max iteration of 100 and another with a max iteration as derived using Grid Search of 1000. To generate the 1000 plot, the value of max_iter will need to be set manually in SVMrbf.py

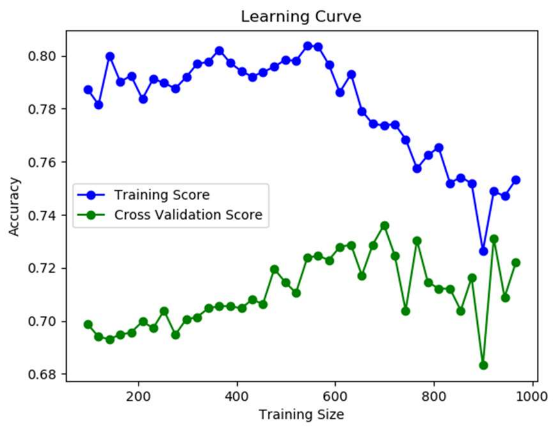
The max iter value is around 1000, however to be able to run both datasets quickly, the parameter has been set to 100 as the upper limit. This is why the plot for dataset 1 shows an increasing trend.

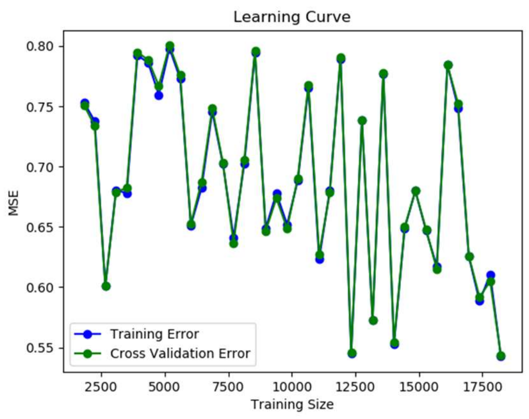
1) For dataset 1, the best two parameters are max iteration = 100 and C = 0.75.
2) For dataset 2, the best two parameters are max iteration = 10 and C = 0.25.

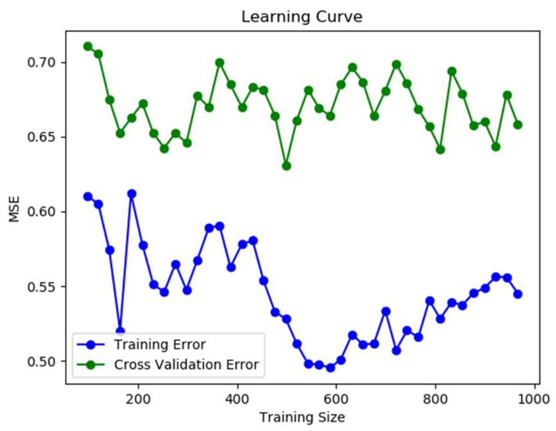
For both datasets, the best class weight option is balanced.

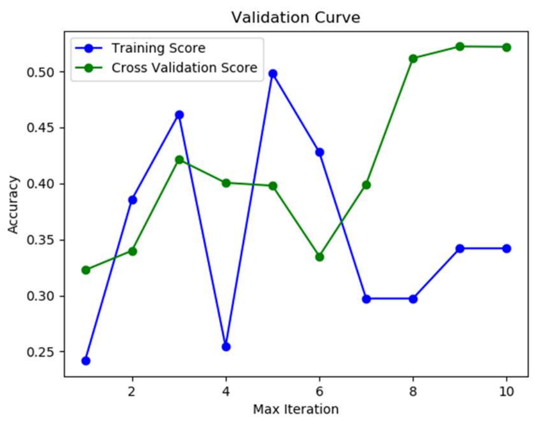
Dataset 2 **Dataset 1**

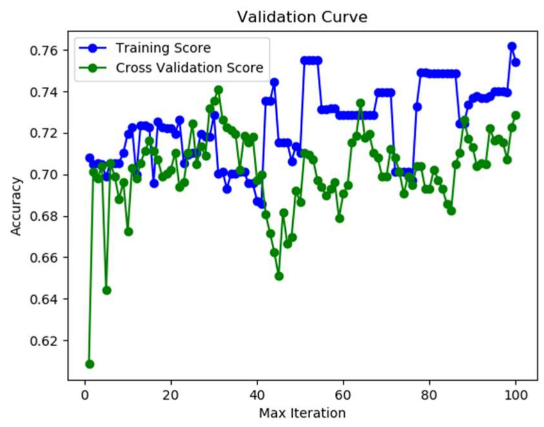


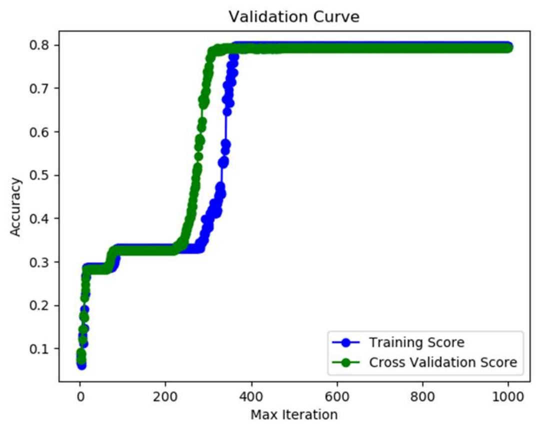



Learning Curve
MSE
Training Size
Training Error
Cross Validation Error


Learning Curve
MSE
Training Size
Training Error
Cross Validation Error


Validation Curve
Accuracy
Max Iteration
Training Score
Cross Validation Score


Validation Curve
Accuracy
Max Iteration
Training Score
Cross Validation Score


Validation Curve
Accuracy
Max Iteration
Training Score
Cross Validation Score

Model Evaluation

Below are the accuracy scores and model times for the experiments.

Dataset 1

Model	Accuracy Score Without Hypertuning		Accuracy Score With Hypertuning	
	In Sample	Out Sample	In Sample	Out Sample
Decision Tree	0.808105873	0.76300578	0.799007444	0.784200385
AdaBoost	0.277915633	0.298651252	0.711331679	0.701348748
KNN	0.789909016	0.757225434	0.756823821	0.755298651
Neural Network - MLP	0.730355666	0.736030829	0.730355666	0.741811175
SVM - Linear	0.713813069	0.697495183	0.744416873	0.753371869
SVM - rbf	0.756823821	0.716763006	0.754342432	0.710982659

Dataset 2

Model	Accuracy Score Without Hypertuning		Accuracy Score With Hypertuning	
	In Sample	Out Sample	In Sample	Out Sample
Decision Tree	0.901325026	0.820247722	0.842664093	0.83437404
AdaBoost	0.83950509	0.838878084	0.847139347	0.844917596
KNN	0.856396981	0.824342307	0.851526852	0.841027741
Neural Network - MLP	0.795015795	0.753813082	0.781370656	0.779916061
SVM - Linear	0.231002106	0.234210257	0.35499298	0.349472822
SVM - rbf	0.2002457	0.199815744	0.342005967	0.340771829

Dataset 1

Model	Times Without Hypertuning		Times With Hypertuning	
	Learning Time	Query Time	Learning Time	Query Time
Decision Tree	0	0.015632153	0	0
AdaBoost	0.078124046	0	0.015636683	0.015613556
KNN				
Neural Network - MLP				
SVM - Linear	0.015661716	0.015626907	0.031266689	0
SVM - rbf	0.031280518	0.015629292	0.030236721	0.010159254

Dataset 2

Model	Times Without Hypertuning	Times With Hypertuning

	Learning Time	Query Time	Learning Time	Query Time
Decision Tree	0.216300488	0	0.103063345	0.010386705
AdaBoost	1.543000698	0.171881914	3.687225103	0.420659304
KNN				
Neural Network - MLP				
SVM - Linear	2.047363043	0.52892375	0.112511635	0.046870232
SVM - rbf	0.199815744	0.2002457	0.152851343	0.040553808

From the results, for the features and hyperparameter chosen, decision tree does well on dataset 1 followed by SVM and Neural network models. However, the data size is small as compared with dataset 2 and this could also impact the overall model choice based on the performance metrics of the small dataset. The features chosen were also limited to facilitate the experiments which has probably added more noise and outliers in the data than expected.

From the results, for the features and hyperparameter chosen, AdaBoost tree does well on dataset 2, followed by KNN and decision tree. Neural network does not do bad either. SVM does not do well at all and this could be because of the kernel and hyperparameters chosen along with the features. The features are limited for this dataset as well to facilitate the experiments. The chosen features help with clustering the data and thus KNN does well here.

The SVM and neural network models ran for a long time. The iterations and epochs chosen need to be considered carefully for large datasets in particular. One of the performance issues with MLP could be due to not normalizing/standardizing the data.

The training time for AdaBoost is the longest. This is due to the number of weak learners provided in the hyper parameter. The query times for the decision trees are relatively small as the trees are pruned and are not deep.

Unfortunately, I missed to capture the times for KNN and MLP. Theoretically, since KNN is a lazy learner, it is fast to train but longer to query in case of many observations.

References

1) Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.