Ensemble Learning

Dataset: get ionosphere: https://archive.ics.uci.edu/ml/datasets/Ionosphere

Using scikit-learn, run classification on Ionosphere first with Bagging and then with AdaBoost. Comparing two algorithms decision tree classifier and SVC classifier (so we will have two boosted classifiers and then the same two but bagged). Performed the classification using K-fold cross-validation, with K being incremented from 2 to 4 to 6 to 8 to 10. The test and train error rates as K changes are plotted.

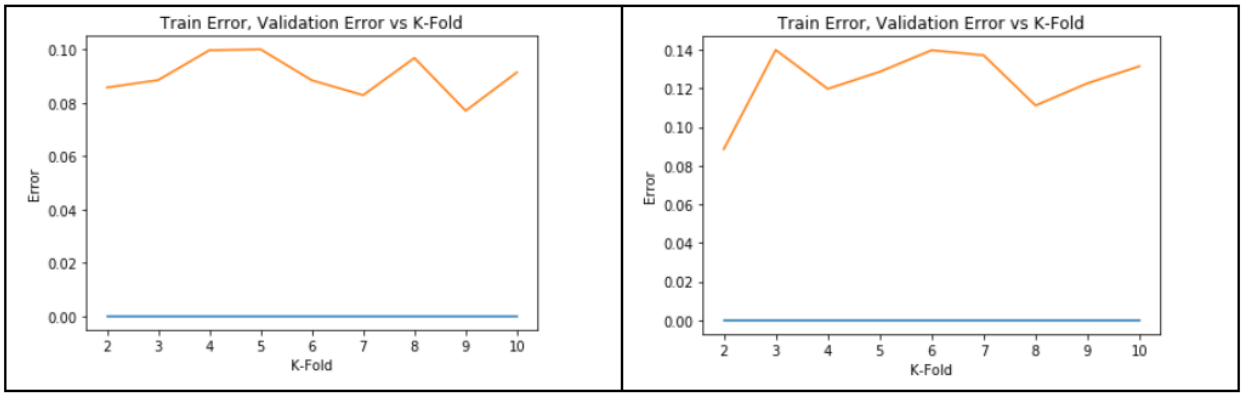

Figure1: Left: Training error(blue) and test error (red) of decision tree classifier with bagging method for different k-fold cross validation . Right: Training error(blue) and test error (red) of decision tree classifier with adaboost method for different k-fold cross validation. The number of estimator for both bagging and ada-boost is 100.

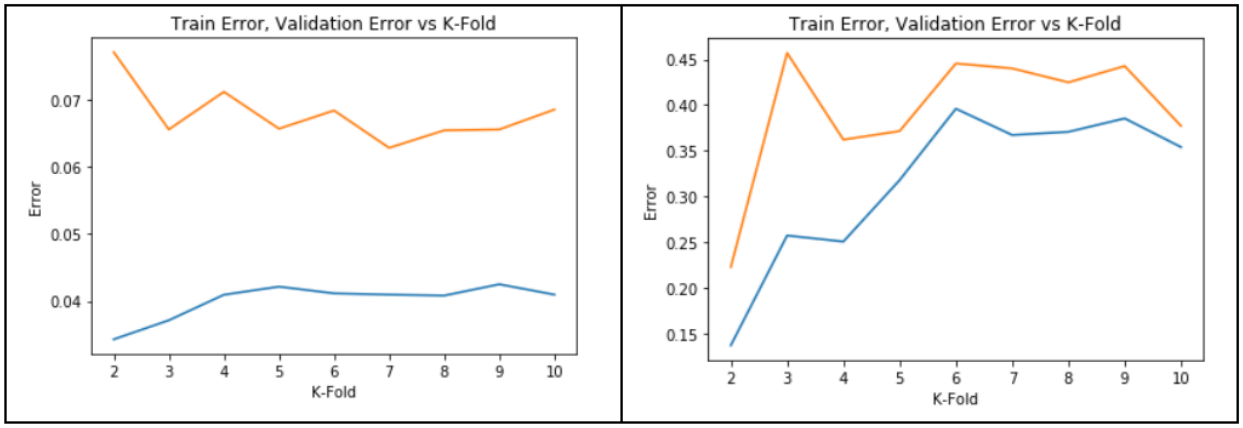

Figure2: Left: Training error(blue) and test error (red) of SVC classifier with bagging method for different k-fold cross validation . Right: Training error(blue) and test error (red) of SVC classifier with adaboost method for different k-fold cross validation. The number of estimator for both bagging and ada-boost is 100.

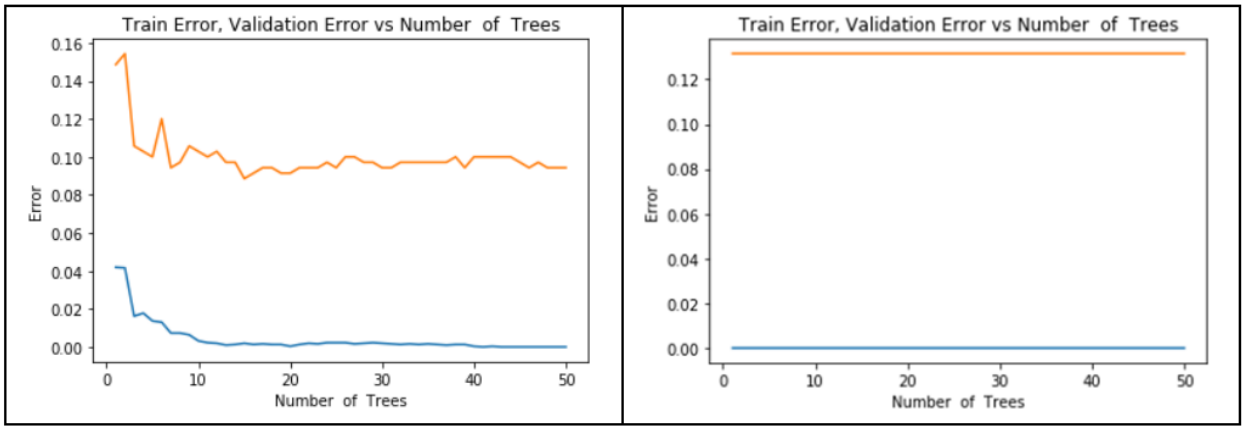

Figure3: Left: Training error(blue) and test error (red) of decision tree classifier with bagging method for different complexity of bagging. Right: Training error(blue) and test error (red) of decision tree classifier with adaboost method for different complexity of adaboost. The number of trees is the number of n-estimator which indicates the complexity of bagging and adaboost. We use the k=10 for k-fold cross-validation.

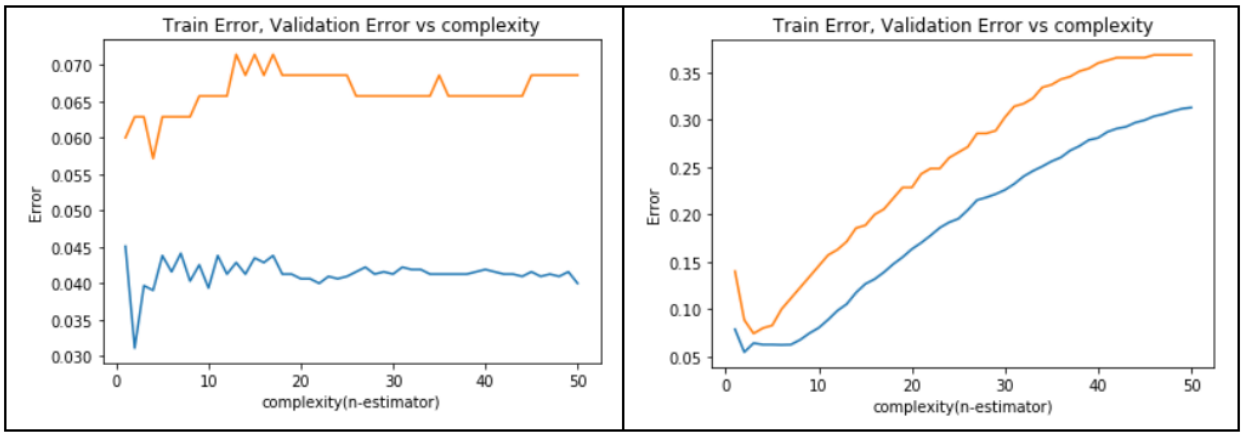

Figure4: Left: Training error(blue) and test error (red) of SVC classifier with bagging method for different complexity of bagging. Right: Training error(blue) and test error (red) of SVC with adaboost method for different complexity of adaboost. We use the k=10 for k-fold cross-validation.

Observations as K changes in terms of bias/variance, overfitting/underfitting:

The difference between the average training and test errors is an indicator of the overfitting situation. According to figure one, there is a very small(almost zero) change in the training error values as the value of K is changing. It indicates that the bias in this model is low. However, the difference between the average training and test errors for bagging method is smaller than the one for Adaboost. If we look at the figure two, we will observe the same trend for the SVC classifier. It implies that Adaboost has higher variance/overfitting in comparison to bagging. According to figure one (decision tree classifier), in both adaboost and bagging the training error is very low (almost zero),which indicates the low bias in the two models. While, if we look at the figure two (SVC classifier), as the value of K is increasing the Adaboost method provides more bias than bagging. If we focus on one specific K, the adaboost method has higher value of test and training errors (which are indicator of bias) in comparision to bagging method for the same value of K.

Comparing Bagging and Boosting in terms of overfitting/underfitting and affected component of the error (bias/variance):

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data. Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data. Overfitting is a modeling error which occurs when a function is too closely fit to a limited set of data points. Underfitting refers to a model that can neither model the training data nor generalize to new data. Intuitively, underfitting occurs when the model or the algorithm does not fit the data well enough. Both Bagging and Boosting decrease the variance of your single estimate as they combine several estimates from different models. So the result may be a model with higher stability. According to figure one, the difference between the average training and test errors for bagging method is smaller than the one for Adaboost. If we look at the figure two, we will observe the same trend for the SVC classifier. It implies that Adaboost has higher variance/overfitting in comparison to bagging. According to figure one (decision tree classifier), in both adaboost and bagging the training error is very low (almost zero), which indicates the low bias in the two models. While, if we look at the figure two (SVC classifier), as the value of K is increasing the Adaboost method provides more bias than bagging. If we focus on one specific K, the adaboost method has higher value of test and training errors in comparision to bagging method for the same value of K. If we look at the effect of changing the model complexity in figurs three and four which are showing the test and training errors for different complexity of the model (as the complexity of th model is increasing), for decision tree and SVC classifiers in order, the difference between the test and training error is smaller for the bagging method in comparision to Adaboost. This difference is indicating the variance of the model and in our observations is higher for Adaboost method. If we look at right side of the figure 4, which shows Training error (blue) and test error (red) of SVC with adaboost method for different complexity of adaboost, we will see an interesting optimum point for both "test/ train values" and "the difference between test/ train error" at the complexity of n-estimator equals to 4, which results in minimum bias and variance(overfitting) in the adaboost model at this point. We can see also the similar optimum point for the bagging method (the left side of fiqure 4) when the complexity (n-estimator) equals to approximately 4 where we have the minimum bias and variance(overfitting) in the model.

If the problem is that the single model gets a very low performance, Bagging will rarely get a better bias. However, Boosting could generate a combined model with lower errors as it optimises the advantages and reduces pitfalls of the single model. In this circumstance, we will choose the Adaboost. According to all our figures (all of our exeriments), since the decision tree classifier and SVC classifiers have high performance (low test and training errors), we can not make any conclusion based on the value of test and train errors. By contrast, if the difficulty of the single model is over-fitting, then Bagging is the best option. Boosting for its part doesn't help to avoid over-fitting; in fact, this technique is faced with this problem itself. For this reason, Bagging is effective more often than Boosting. According to figure 3 and 4 which are showing the test and training errors for different complexity of the model (as the complexity of th model is increasing), for decision tree and SVC classifiers in order, the difference between the test and training error is smaller for the bagging method in comparision to Adaboost. This difference is indicating the

variance of the model and in our observations is higher for Adaboost method. As a result, we will choose the bagging method.

Random Forest ensemble learning algorithm:

Random Forest is a lot like Bagging with Decision Tree. However, the significant difference has to do with what subset of the feature space is used to grow the trees in the forest. To expatiate, let N be the number of the different kinds of feature that describes an example in the data set and let n be a number much less than N. Unlike Bagging with Decision Tree, to grow a tree in a Random Forest, a randomly sampled n features are considered when deciding which feature to use as a node. Due to such random sampling, the trees in a Random Forest are less likely to be correlated when compared to the trees used for bagging. That, in turn, reduces the likeliness that the trees in the Random Forest will make the same mistake.
Random forest is a classifier that evolves from decision trees. It actually consists of many decision trees. To classify a new instance, each decision tree provides a classification for input data; random forest collects the classifications and chooses the most voted prediction as the result. The input of each tree is sampled data from the original dataset. In addition, a subset of features is randomly selected from the optional features to grow the tree at each node. Each tree is grown without pruning. Essentially, random forest enables a large number of weak or weakly correlated classifiers to form a strong classifier