Homework 3
COSC 6342: Machine Learning

Submitted by
S M Salah Uddin Kadir (1800503)
Rubayat Jinnah (1891217)

In this experiment, we are using “Ionosphere” dataset. The dataset has 351 examples, where each example has 34 attributes. All attribute values are continuous numbers.

- Number of examples: 351
- Number of attributes: 34

The target of this dataset are free electrons in the ionosphere. If the returns from the radar, the attributes, shows an evidence of some type of structure in the ionosphere, then we classify that example as a “Good” return otherwise “Bad” return. So, it is a classification problem, and the target has two categorical classes.

- Good
- Bad

We used three estimators to classify the dataset with Bagging and Boosting approach.

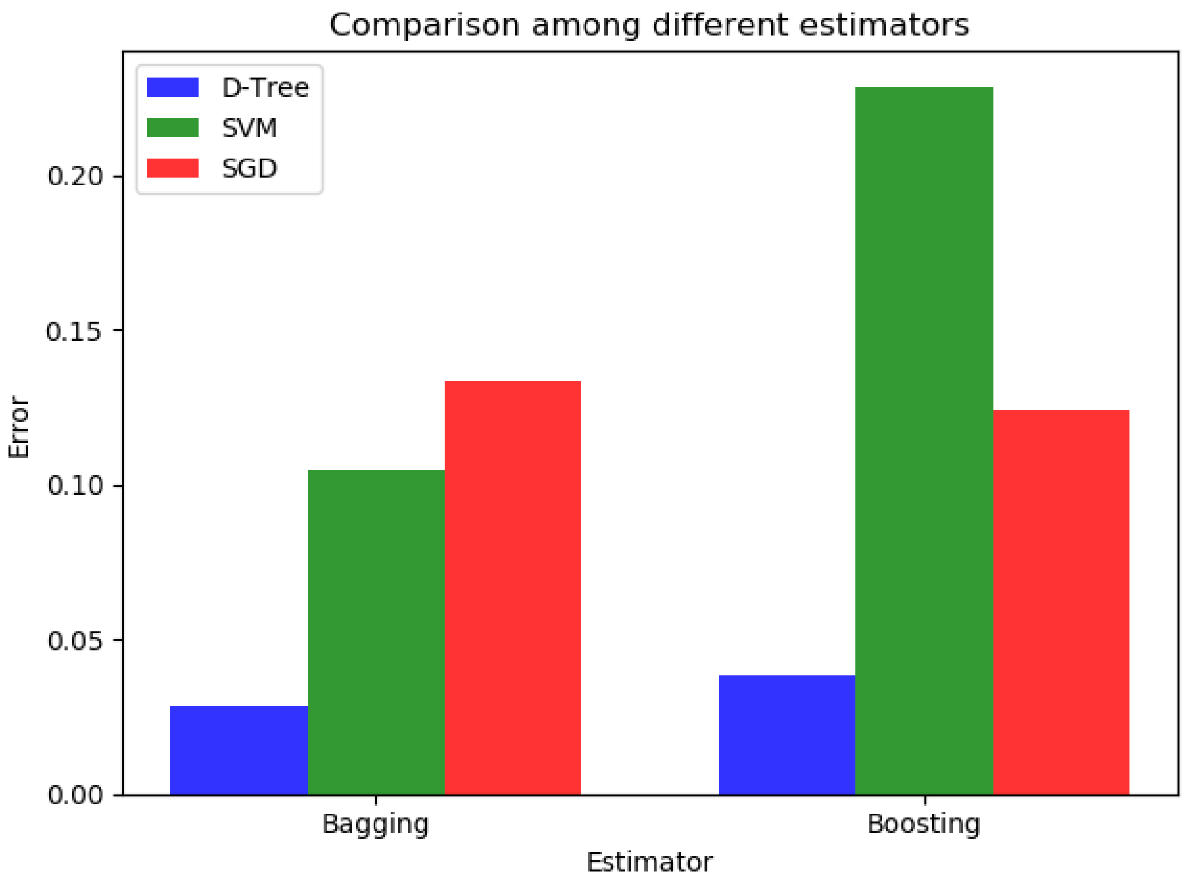
- Support Vector Machine (SVM)
- Decision Tree (d-tree)
- Stochastic gradient descent (SGD)

We used 50 trees for bagging and boosting.

(c)

<u>General approach (without k-fold):</u>

We used 70% of the dataset to train the model and remaining 30% to test the model.

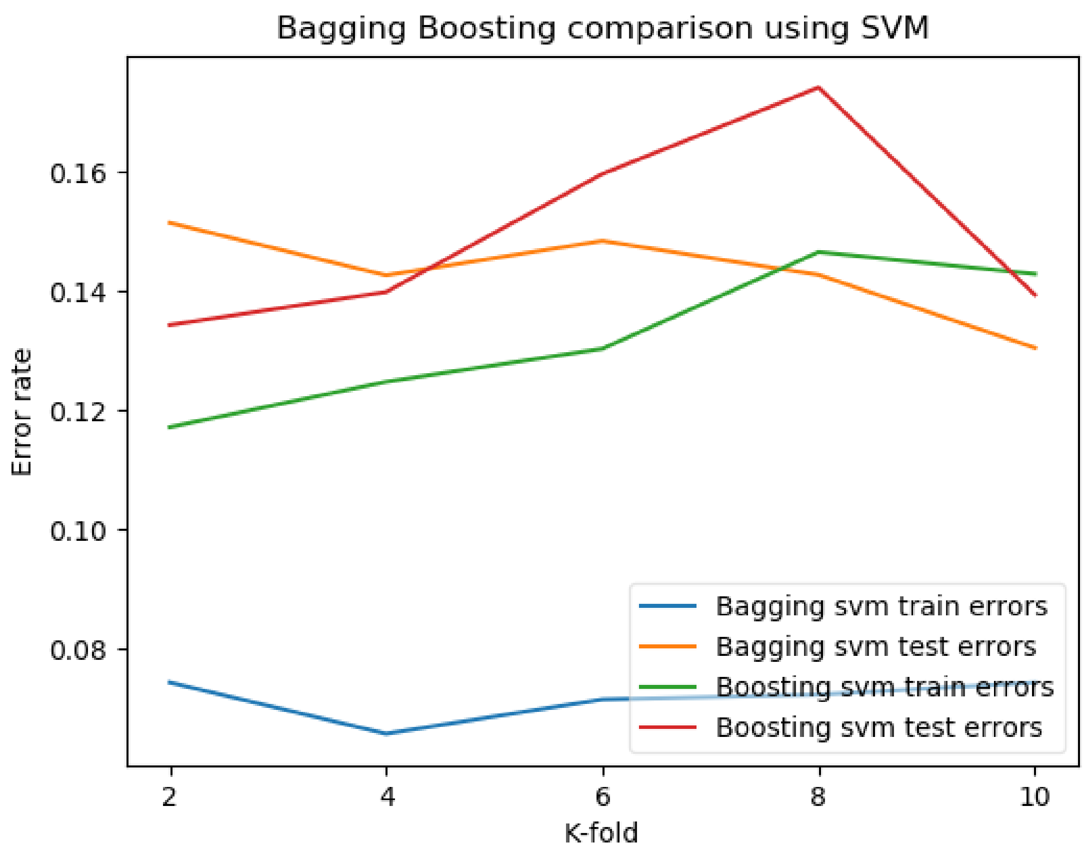


We applied Bagging and Boosting to classify the "Ionoshere" dataset. We used 3 estimators: Decision tree, Support vector machine (SVM) and the Stochastic gradient descent (SGD) algorithms. From the plot, we can see that the error rate of decision tree is lower than the other estimators for both Bagging and Boosting.

__Comparison with k-fold cross-validation:__

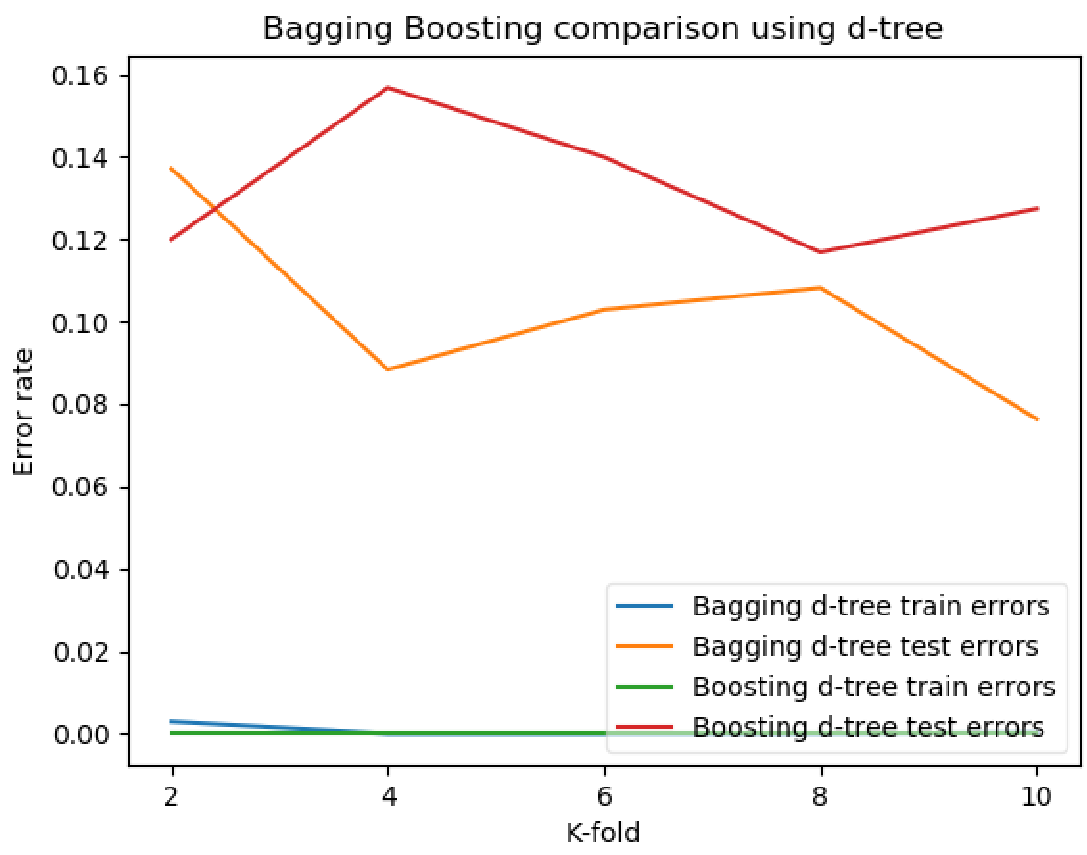
We had an experiment with k-fold cross-validation using the same estimators and dataset for the Bagging and Boosting approach. We calculated the test and train error rates with 2, 4, 6, 8, and 10 folds.

<u>Comparison using SVM classifier:</u>



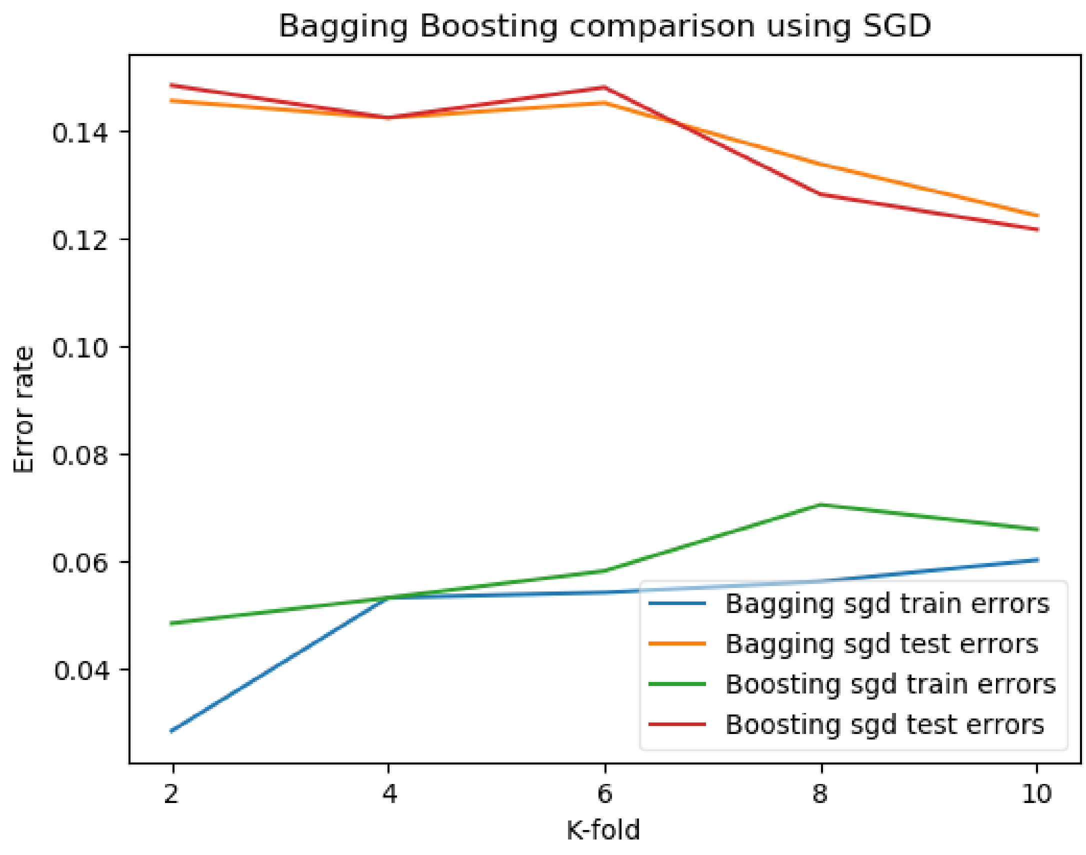
Using SVM classifier, we see that the train error is very low for bagging than the boosting. Although, the test error is similar for both.

<u>Comparison Using Decision Tree classifier</u>:



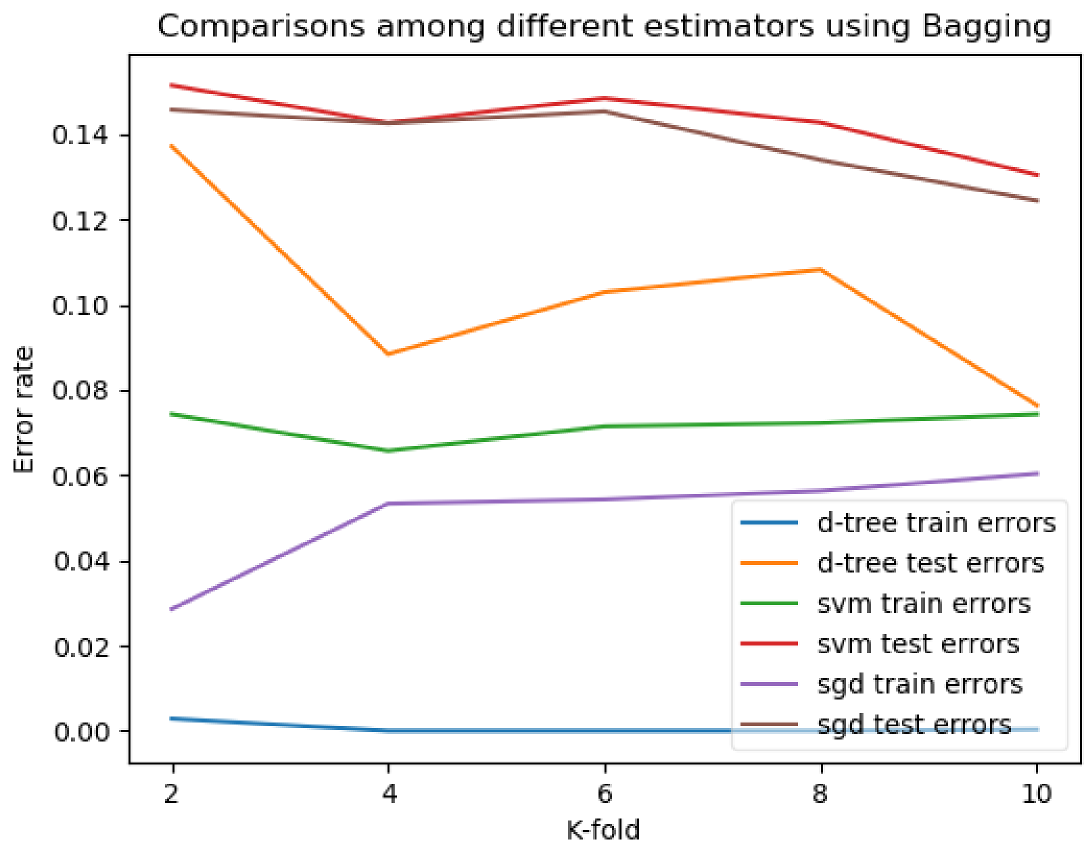
For decision tree, the train error is 0 for both Bagging and Boosting. The test error is relatively high for Boosting.

<u>Comparison Using SGD classifier:</u>



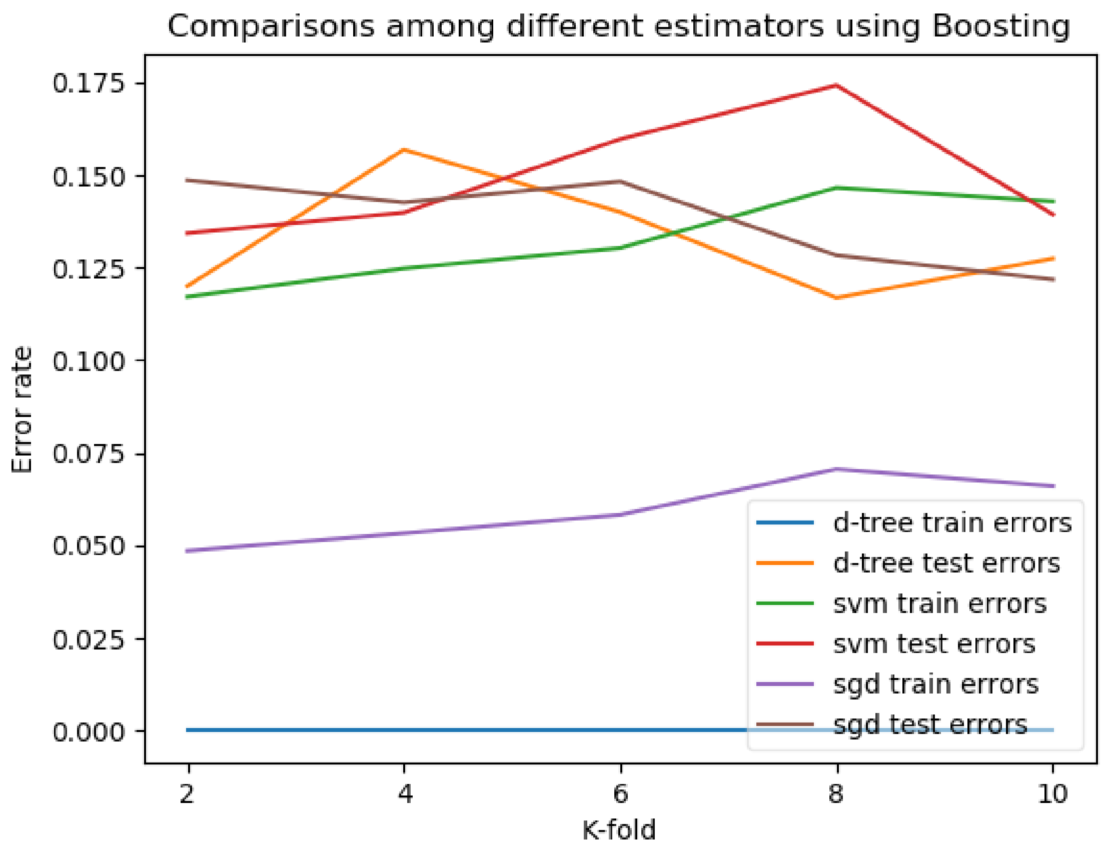
For SGD classifier, we see that the performance of Bagging and Boosting is very close on both training and testing dataset.

<u>Comparison among all classifiers using only Bagging:</u>



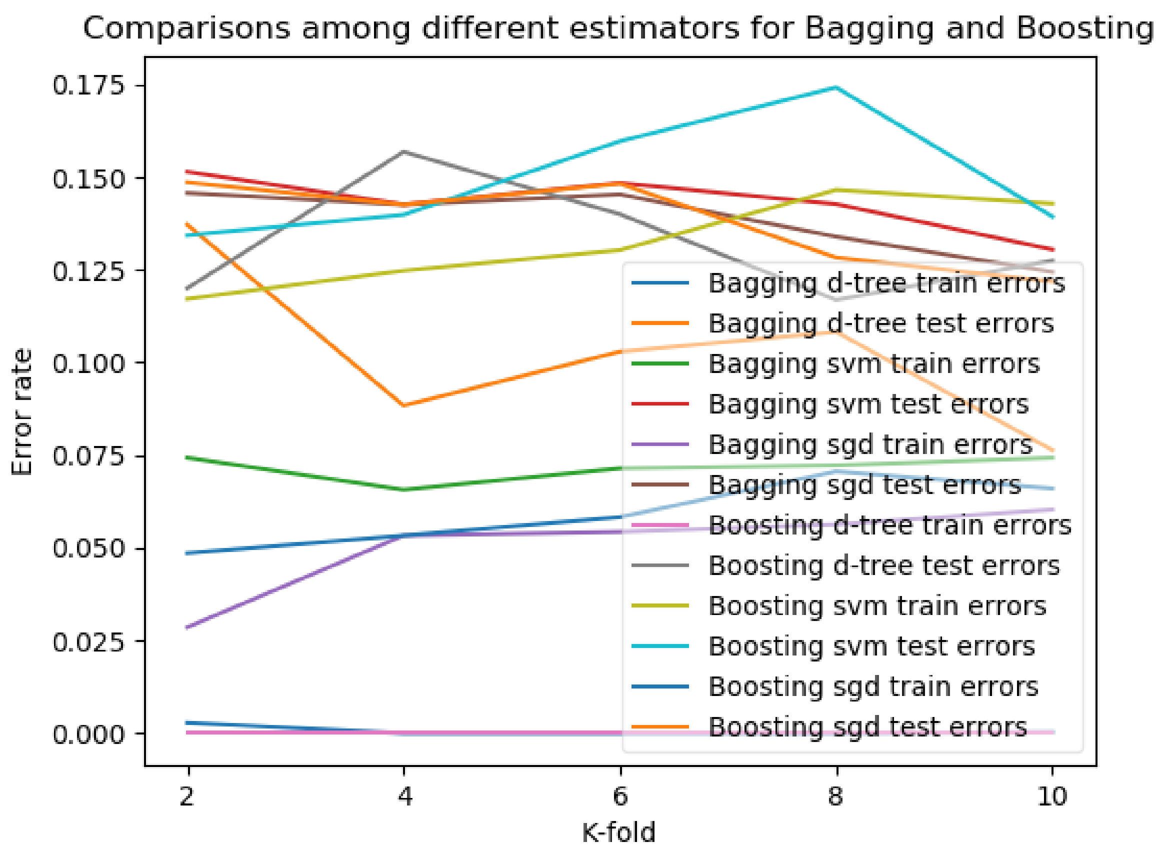
Now, we want to compare among different classifiers using only Bagging. From the plot, we see that the training error is higher for SVM among these three classifiers. We can also notice that ***the d-tree has the best performance for both training and testing among these three classifiers***.

<u>Comparison among all classifiers using only Boosting:</u>



There is ***no training error for decision tree.*** Although, the performance of different estimators on the testing dataset ***does not differ very much*** with each other.

Comparison among all classifier for both Bagging and Boosting:

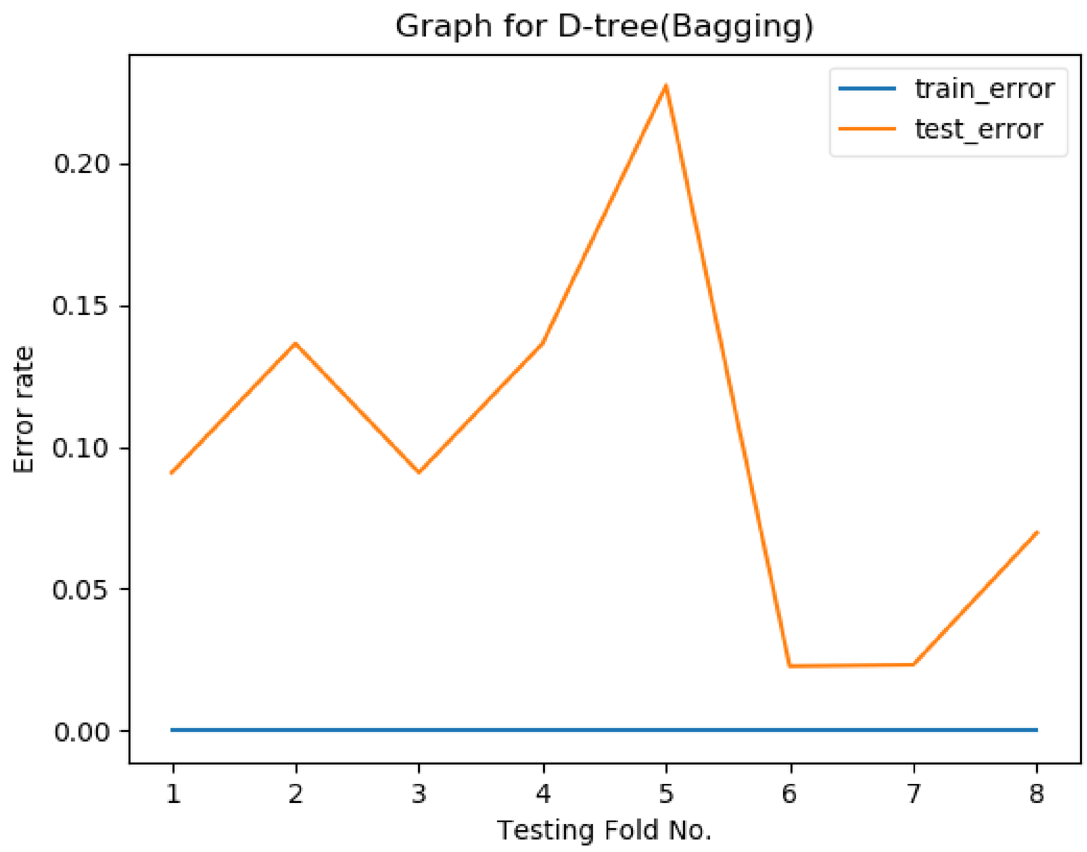


The plot is the combination of all classifiers for both Bagging and the Boosting so that we can compare with each other. We can conclude that ***the performance of Decision tree is relatively better than the other classifiers***.

Complexity Graph with fixed K (8 fold):

Configuration:

- Fold size = 8
- Tree depth = infinite
- Number of bags = 100



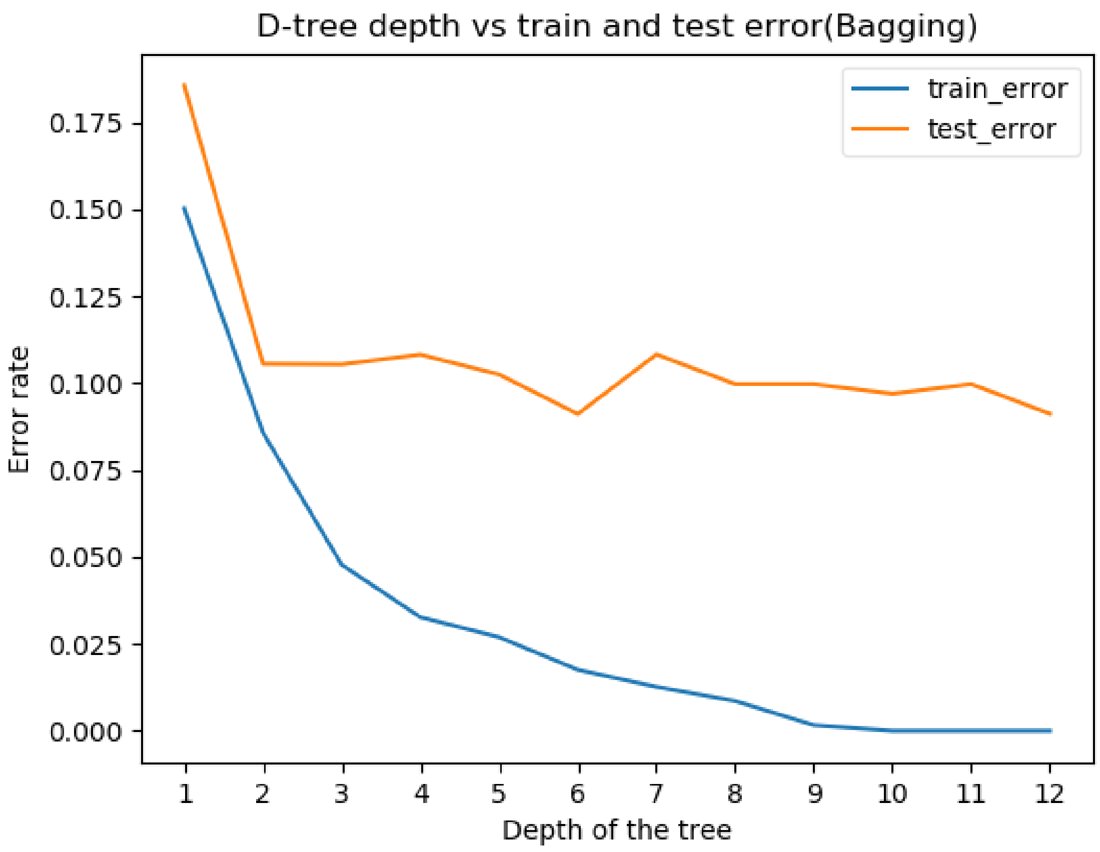
We divided the dataset into 8 groups. We used a single group to test the model when the other groups to train. From the plot, we can see that there is almost no train error, although the test error varies for different groups.

As there is no train error but test errors, we can conclude that there may exist a high variance, and the train dataset overly fit the model.

Now, we want to test the complexity performance for d-tree using the **variable depth**.

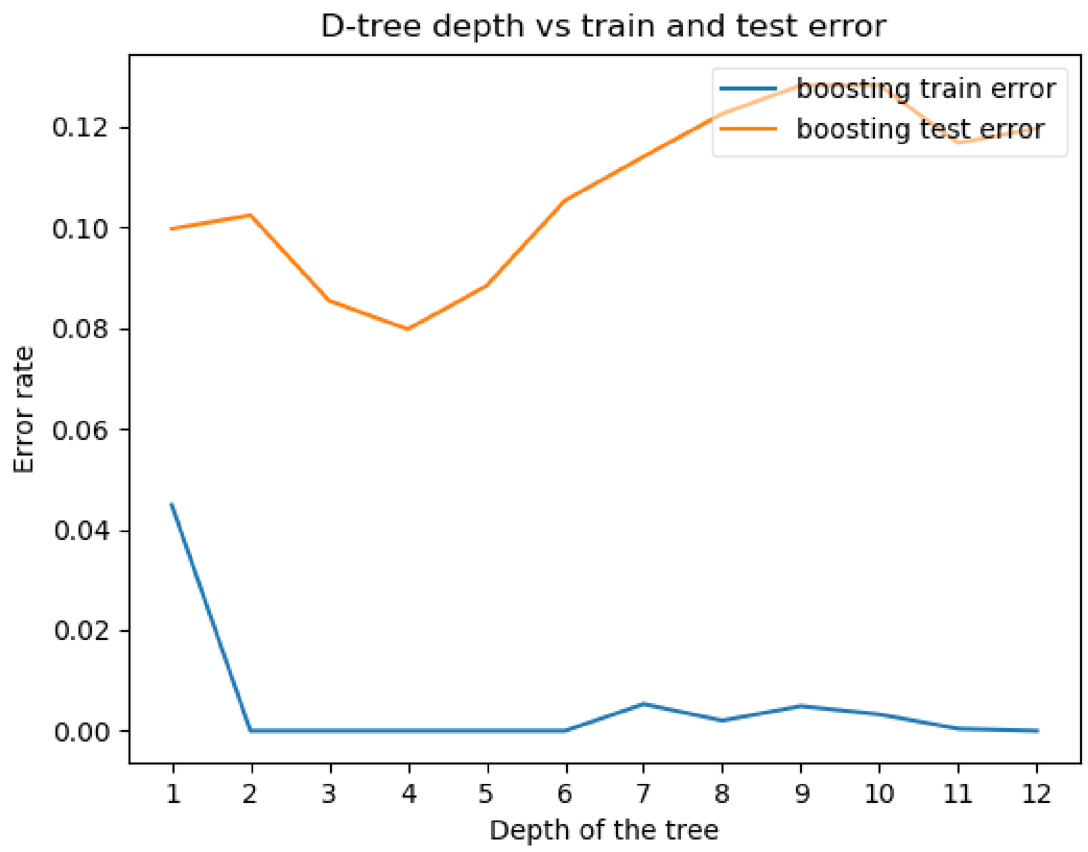
<u>Configuration:</u>

- Fold size = 8
- Tree depth = variable
- Number of bags = 100



From the graph, we can see that the train error has decreased with the increase of the level of the tree, and it has been zero at level 9, although the test error did not decrease. It seems that test error got the lowest point at level 6. By adding the depth of the tree, the complexity of the model is increasing, but it is not increasing the performance of the model on the test data.

Now, lets check for Boosting.

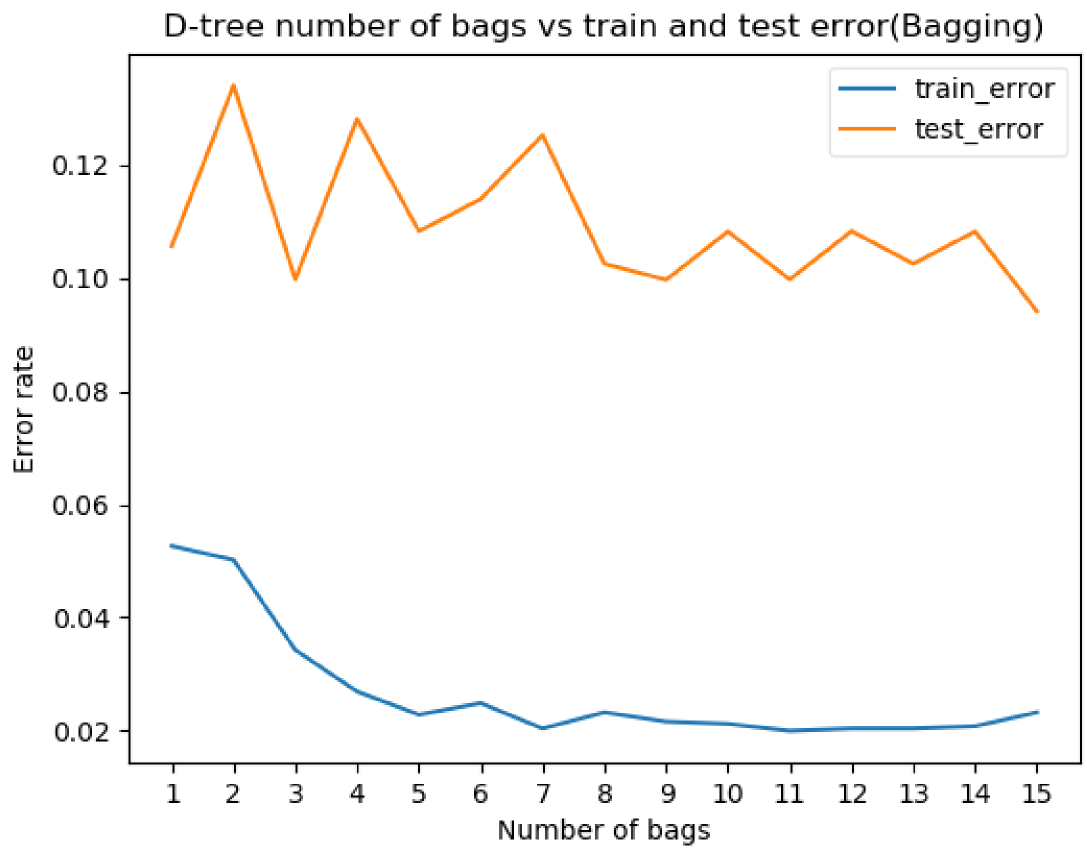


From the plot, we can see that the model is getting overfitted after level 4. So, we can say that the model is getting more complex, and over-fitting the dataset with the increase of the depth of the tree.

Now, we want to analyze the complexity of the model based on **the number of bags (sampling tree).**

Configuration:

- Fold size = 8
- Tree depth = 4 (we know, the best performance on test data from previous experiment)
- Number of bags = variable



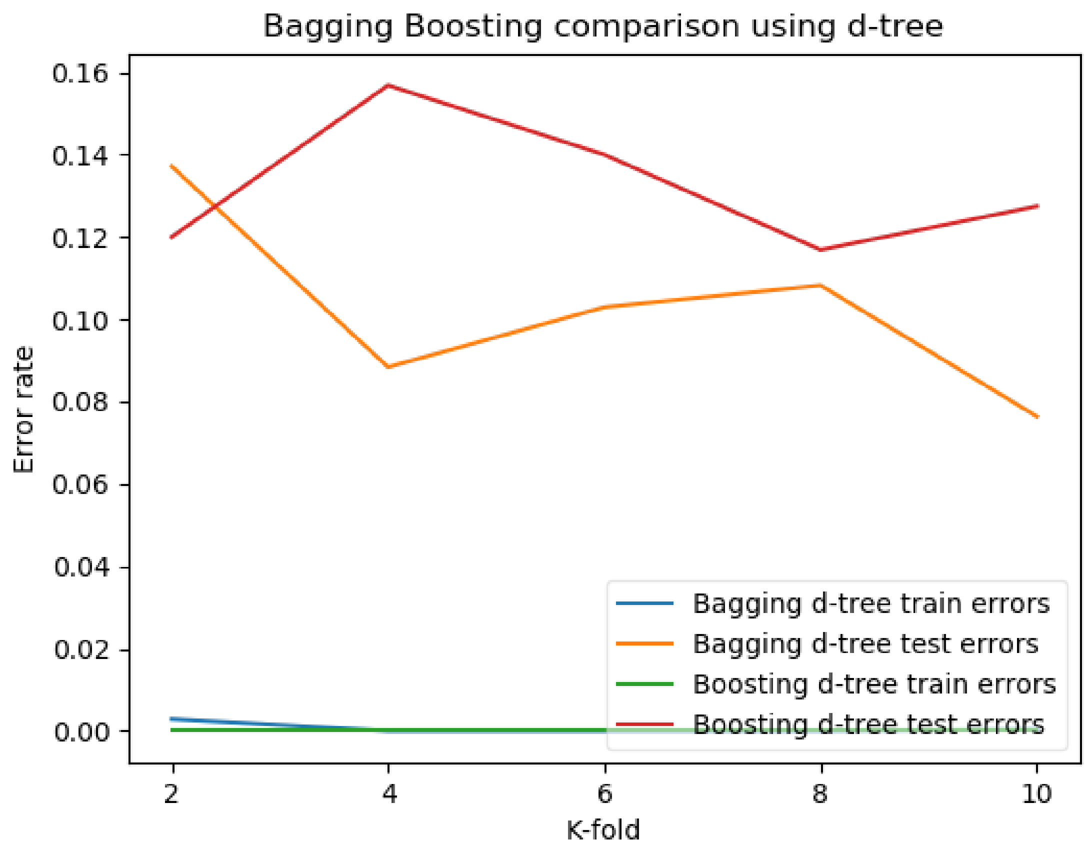
From the plot, we can see that the train error and test error is minimum at 11. So, we can conclude that if we configure the Bagging method with 11 bags (n_estimators), then we can get the best performance when the fold size is 8.

So, Our best configuration for D-tree(Bagging):

- Fold size = 8
- Tree depth = 4
- Number of bags = 11

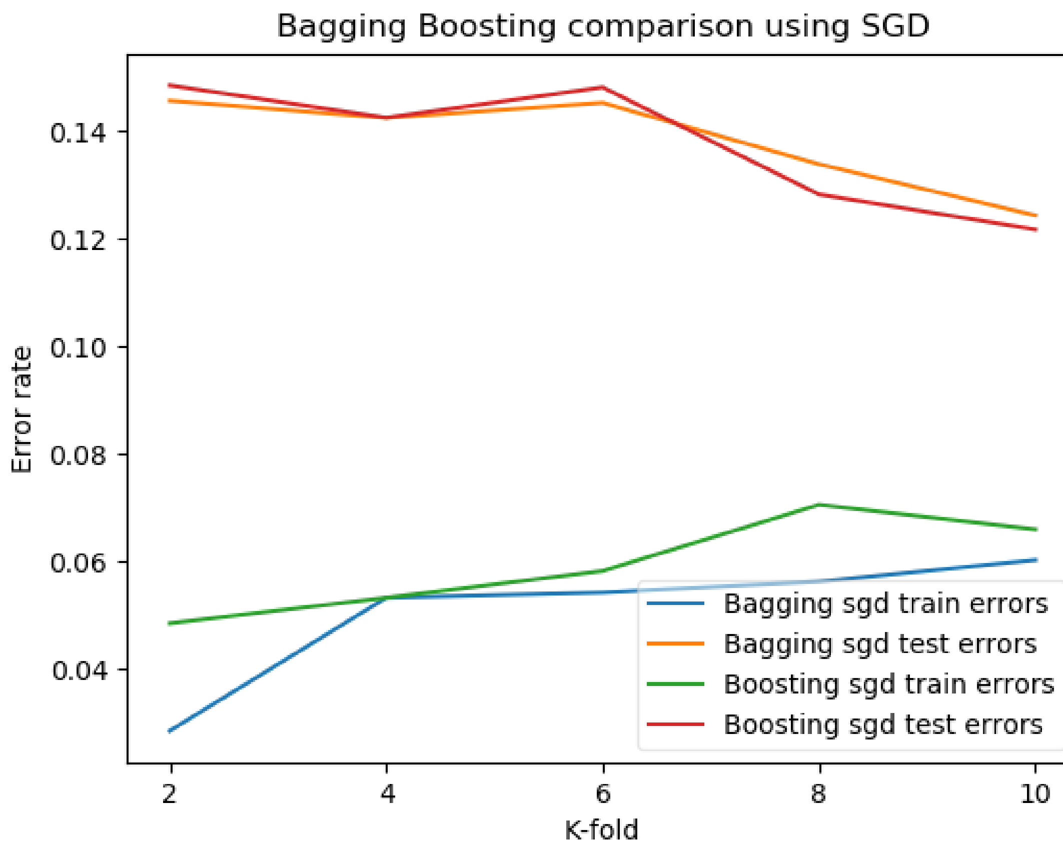
(d)

First, we want to analyze the bias/variance for **decision tree** where the bias seems almost zero, but a very high variance.



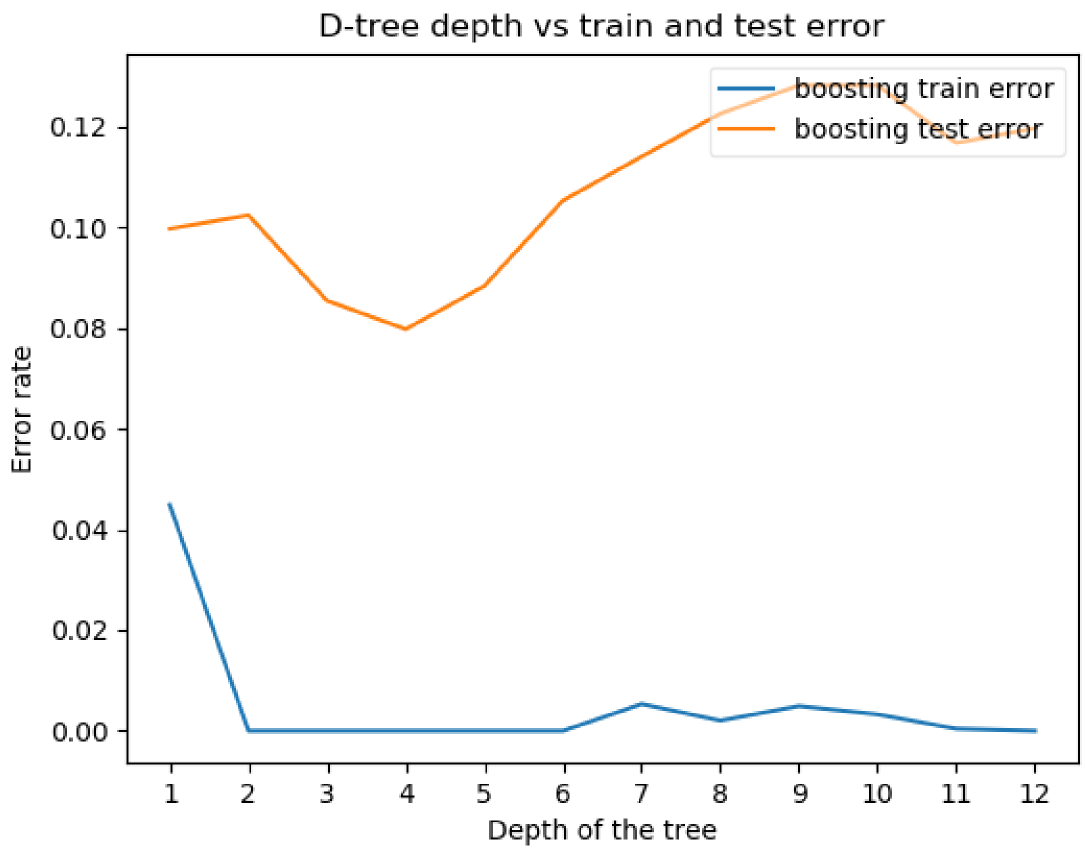
From the plot, we see that there are almost no train errors for Decision tree, but the test errors for both Bagging and Boosting. From the plot, it seems that the model has been overly complex and overfitted (highly likely, k-fold is giving the error rate after doing average) as there is no error for the train datasets. We see that the test error is almost same for 4-fold and 8-fold in case of Bagging, and 2-fold and 8-fold in case of Boosting. So, we conclude that the model is suffering by a high variance where the bias is almost zero as there is no train error. The interesting fact here is the error rate is not changing in a single direction rather it is increasing and decreasing and then again increasing, seems like maintaining a pattern.

Now, we want to discuss for **SGD classifier** where the bias and variance is merging with the increase of number of k-fold.



From the plot, we can see that the train error is very low initially, but a very high test errors. The test errors are decreasing with the increase of train errors by increasing the value of K. So, we can conclude that there is a high variance and low bias for smaller K values. The variance error is getting low with the increase of K value, but it increases a little bias error also.

Now, we will have a plot with the depth of the tree to describe better the overfitting and underfitting scenario.



The graph shows the change of train errors and test errors with the increase of the depth of the tree. We see the error rate on train data has been zero at level 2, the test error got the lowest point at position 4, and then the error is increasing with the increase of the depth of the tree. So, the model is getting very complex with the increase of the depth which also increases the variance and overfits the dataset.

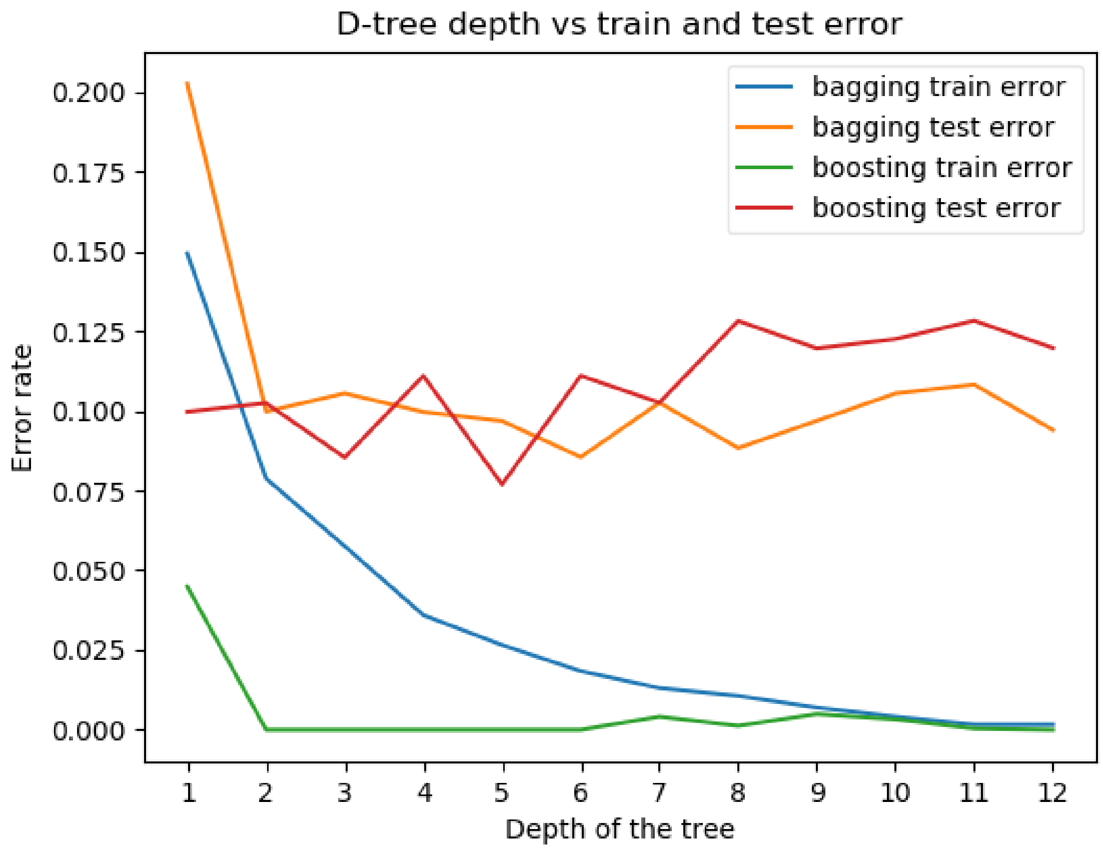
From the plot, we can say that when the depth is 1, it underfits the dataset, and there is some bias. The model overfits the dataset with the increase of the depth of the tree, and it also increases the variance.

(e)

An overfitted model is the outcome of high variance and low bias. To reduce the error rate of an overfitted model, bagging can be implemented. In Bagging approach, the variance is reduced to an average value so that the model can fit.

On the other hand, an underfitted model is the outcome of high bias and low variance. To reduce the error rate of underfitted model, boosting can be implemented. In Boosting approach, the bias is reduced significantly so that the model can be fit.

Now, we will have a comparison plot between Bagging and Boosting using our Ionosphere dataset.



We see that the dataset is **getting overfit for Boosting** with the increase of the depth of the tree, where the test error didn't change for Bagging. The train error has been zero with the increase of the depth, the model best follows the training data, it is too dependent on that data and it is likely to have a higher error rate on new unseen data.

(f)

We need to choose Bagging or Boosting based on the dataset. We can decide based on the result by running a single model.

We know, a underfitted model is a model with high bias and low variance. So, if the problem is that the performance of a single model is very low, then we can **choose Boosting**. Boosting could generate a combined model with lower errors as it optimizes the advantages and reduces pitfalls of the single model. To increase the accuracy and reduce the error rate of both training and testing data, **boosting should be chosen**.

By contrast, an overfitted model is a model with high variance and low bias. So, if the problem is over-fitting, then **Bagging can be the best option.** Boosting for its part doesn't help to avoid the over-fitting rather it may sometime increase the problem that we have seen in our previous plots. In an overfitted model the testing error is higher than the training error, as training model fits all the noise and outliers. Bagging helps to reduce the noise and outliers. For this reason, Bagging is effective more than Boosting for over-fitting problem.

(g)

Each individual tree in the random forest splits out the class prediction. The class with the most votes becomes our model's prediction.

- The random forests algorithm applies the general technique of bootstrap aggregating like bagging to tree learners.
- Additionally use "feature bagging"
    - Random subset of features. Eliminate strong features bias. Ensures the correlation of the trees in an ordinary bootstrap sample.

So, random forest ends up with trees that are not only trained on different sets of data like bagging, but also use different features to make decisions. This forces even more variation amongst the trees in the model and ultimately results in lower correlation across trees and more diversification.