Assignment 3

Ankit Gawde
011600269

1. Summary of Approximate Statistical Tests

The paper review five approximate statistical tests for determining whether one learning algorithm out-performs another on a particular learning task. These tests are compared experimentally to determine their probability of incorrectly detecting a difference when no difference exists. The paper also measures the ability to detect algorithm differences when they do exist. The purpose of this paper is to solve the statistical questions that arise in research, development and application of machine learning algorithms for classification. These problems were listed by author in a tree format for better understanding. The subject of this paper is to solve a single domain problem which analyzes algorithms and chooses the best algorithm which uses small of dataset. Or the author states the question as "Given two learning algorithms A and B and a small data set S which algorithm will produce more accurate classifiers when trained on data sets of the same size as S?".

The Five statistical tests which are the main subject of this paper are described as:
a) **McNemar's Test**- The test compares two algorithms A and B on basis of how they perform on a dataset under assumption that they have null hypothesis. Under null hypothesis, both the algorithms should have same error rate. Firstly, the dataset is divided into training and testing set. Both algorithms are applied on training data to get classifiers. The test is based on chi-square test for goodness-of-fit that compares the distribution of counts expected under the null hypothesis to the observed count. If the null hypothesis is correct, then probability that chi-square's distribution is more than 3.8414 is less than 0.05 else algorithms perform different. The shortcomings of this test are, first it does not directly measure variability due to the choice of the training set or internal randomness of the learning algorithm. Second, it does not directly compare the performance of the algorithms on training sets which are as large as the data |S|.

b) **Test for the difference of two proportions –** It is based on measuring the difference between the error rate of algorithms A and B. The probability of misclassification is calculated for both the algorithms. If we assume that both probability of misclassification are independent, then the difference between two independent normally distributed random variables is itself normally distributed. A statistic of a standard normal distribution is obtained which is given by

$$z = \frac{p_A - p_B}{\sqrt{2p(1-p)/n}}$$

The null hypothesis is rejected if $|z| > Z_{0.975} = 1.96$. The test has two shortcomings, first, the probabilities are dependent and second it shares the drawbacks of McNemar's test.

c) **Resampled paired t-test –** In this test, the author considers 30 trials, in each trail the available sample is randomly divided into training and testing set. We calculate the probability of

misclassification by each algorithm. So, we get 30 differences and we apply Student's t-test to compute t-distribution with n-1 degrees of freedom. The null hypothesis is rejected if

$$|t| > t_{29,0.975} = 2.04523.$$

There are two potential drawbacks, first the individual drawbacks won't have normal distribution as probabilities are dependent and second, the probabilities for each trail are dependent because the test sets and the training sets in the trails overlap.

d) **k-fold cross validated paired t-test –** In this test, the dataset is divided into k disjoint sets of equal size and k trials are conducted on those disjoint set. The advantage of this approach is that each test set is independent of the others. But still the test suffers as the training sets overlap. The authors illustrated this point by conducting nearest neighbor algorithm on the training set. The results which they got showed that two folds of the validation will be correlated rather than independent.

e) **5x2cv paired t-test –** The authors found that in previous cases the t statistic was large because the variance was slightly underestimated when the training sets overlapped. Hence, they ended up replacing numerator of t-statistic which led them to 5x2cv paired test. In this test, they perform 5 replications of 2-fold-cross-validation. They divide the data into two equal sized sets and perform both the algorithms. This produces four estimates. Subtracting error estimates gives them two estimated differences. This way a new t-statistic is achieved, and it is claimed under the null hypothesis. This makes the probabilities independent and training sets completely non-overlapping. First, they employ the normal approximation to the binomial distribution. Second, they assume pairwise independence between probabilities and third they assume independence between the numerator and denominator of the new t-statistic.

They conducted experiments on these five models to measure the probability of Type I error of the algorithm which occurs when the null hypothesis is true and learning algorithm rejects the null hypothesis. They found out that each algorithm has an overall error rate of E, and each algorithm has same total variance. However, for any given test example, the algorithms have very different error rates. The results show that paired-differences t-test based on 10-fold cross-validation exhibits elevated probability of Type I error whereas McNemar's test showed a low error. The 5x2cv paired test showed an acceptable Type I error. If the algorithms are executed only once, then McNemar's test is preferred else if the algorithms are executed a greater number of times then 5x2cv is recommended because it is slightly more powerful and because it directly measures variation due to the choice of training set.

2. Summary of Overfitting and Undercomputing in Machine Learning

The paper focuses mainly on avoiding overfitting which is achieved by Undercomputing. The author takes supervised learning into consideration as it has been a major concern in machine learning, as the machine has to learn from labeled training data. The author discusses various examples which needs a function to be learned and this function can be represented as a set of rules, a decision tree, a Bayes network or a neural network.

Learning algorithms try to find a hypothesis class that fits the given data i.e. a lot of classes. Computing problem of minimizing the objective function is NP-hard. Hence, author suggests of using heuristic algorithms. There are numerous heuristic algorithms and finding the right one is hard, for a given particular dataset. The experiments showed their performance was worse.

The author tries to find a connection between various previously disparate fields where the author points out the interaction between computational issues and statistical issues. The key problem arises when they try to test the objective function on the new data points. The goal is to maximize the predictive accuracy on test data and not the training data. When we try to increase the accuracy on the train data, then there are chances that we fit the noise in the data, rather we should find a general predictive rule. Hence, by augmenting the objective function with various penalty terms have been very successful and works better when applied on new data points. The experimental results confirm that this removes the paradox. It is similar to Occam's razor principle which says the simplest solution is the correct one.

3. Summary of Ensemble Methods in Machine Learning

Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a vote of their predictions. Over the years there has been man methods for constructing ensembles and few of them are discussed in this paper. This paper provides a brief survey of methods for constructing ensembles and reviews the three fundamental reasons why ensemble methods are able to out-perform any single classifier within the ensemble.

a) Bayesian Voting: In Bayesian probabilistic setting, each hypothesis defines a conditional probability. In this setting the ensemble method can be treated as an ensemble which consists of all the hypothesis in H, each weighted by its posterior probability. Applying bayes rule, the posterior probability is proportional to the likelihood of the training data times the prior probability of h. It primarily addresses the statistical component of ensembles. They found that Bayesian setting was not optimal as it does not address the computational and representational problems and it is hard to construct a space H and assign a prior that captures knowledge accurately.

b) Manipulating the Training Examples – It manipulates the training examples to generate multiple hypothesis. It works well on unstable learning algorithms. One of the techniques used is Bagging which presents the learning algorithm with a training set that consists of a sample of m training examples drawn randomly with replacement from the original training set of m items. Other technique used is to construct the training sets by leaving out disjoint subsets of the training data and the third technique is ADABOOST algorithm. The ADABOOST algorithm generates multiple hypothesis. It maintains a set of weights over the training examples. The effect of the change in weights is to place more weight on training examples that were misclassified and less weight on examples which were correctly classified. The final classifier is constructed by a weighted vote of the individual classifiers.

c) Manipulating the Input Features – It is used to generate multiple classifiers. They noted in the paper that, this technique works well, when the input features are highly redundant as deleting even a few of the input features hurt the performance of the individual classifiers. If the input features are highly redundant, then ensemble classifier can match the performance of human experts.

d) Manipulating the Output Targets – This technique is used to manipulate the y labels that are given to the learning algorithm. The technique used was error-correcting output coding. This technique basically encodes each class as an L-bit codeword where each bit is 1 if it is present in the subset. The i-th learned classifier will try to predict bit of these codewords. Results show that this technique improves the performance of decision tree and backpropagation neural network algorithm. This technique can be combined can also be combined with ADABOOST to yield an excellent ensemble classification method and main advantage of it is simplicity as it can work with any learning algorithm for solving 2-class problems.

e) Injecting Randomness- In this technique, randomness is added to the dataset to generalize it. They found that injecting randomness in training examples effects in effectiveness. The author then pointed out on some examples which were carried in other papers and noted the results of injecting randomness. In some cases, it worked well and made the learning algorithm effective, but there were some cases where it did not reach the potential of other techniques. Adding randomness into training process always helps as it generalizes the classifier. Classifier then can perform much better on test data.

Several experiments were carried out to compare ensemble methods. The results show that ADABOOST often gives the best results. Bagging and randomized tress give similar performance, although randomization is able to do better in some cases than Bagging on very large data sets.

4. Summary of Greedy Function Approximation

Function approximation is viewed from the perspective of numerical optimization in function space. Function estimation or predictive learning problem, where one has a system consisting of random output label y and a set of random input variables. Using a training sample of known y label, the goal is to obtain an approximation function that maps x to y that minimizes the expected value of some specified loss function over the joint distribution of the values. A common procedure is to restrict function to be a member of parameterized class of functions.

Numerical Optimization is used on the approximation function to make it a function optimization problem. The approximation function is usually a simple parameterized function of the input variables x, so choosing a parameterized model function changes the function optimization problem to one of parameter optimization. The solution for the parameters for function $F^*(x) = F(x; \boldsymbol{P}^*)$ can be given by $\mathbf{P}^* = \sum_{m=0}^{M} \mathbf{p}_m$.

Steepest-descent is one of the simplest of the frequently used numerical minimization methods. The negative gradient is called as steepest descent direction. The author takes a nonparametric approach and apply numerical optimization in function space. This nonparametric approach breaks down the join distribution of (y, x) is estimated by a finite data sample. The strategy used in this paper is a greedy-stagewise approach also known as matching pursuit in signal processing which is a squared-error loss. Gradient boosting is basically a model in the form of an ensemble of weak prediction models where it builds a model in a stage-wise manner like boosting and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

The author proposes the gradient boosting algorithm and gives an overview on some of the applications. The gradient boosting strategy is applied to several popular loss criteria.
The applications of gradient boosting are
a) Gradient boosting on squared-error loss produces the usual stagewise approach of iteratively fitting the current residuals.
b) It produces a least absolute deviation regression using any base learner.
c) Gradient boosting works well in regression trees as LAD_TreeBoost algorithm proposed in the paper uses least squares to induce the trees, also it is highly robust. The trees use only order information on the individual input variables and pseudo-responses. Squared-error loss is much more rapidly updated than mean-absolute-deviation when searching for splits during the tree building process.
d) M-Regression techniques attempt resistance to long tailed error distributions and outliers while maintaining high efficiency for normally distributed errors. M_TreeBoost algorithm performs similar to least-square boosting for normally distributed errors and similar to that of LAD regression with very long tailed distributions. For error distributions with only moderately long tails it can have performance superior to both.
e) Two class logistic regression and classification – here the loss function is negative binomial log-likelihood. The algorithm proposed in the paper, can be used for two class logistic regression and

classification which is for likelihood gradient boosting with regression trees. It uses influence trimming which basically is deleting all observations with w_i values less than w_{l_α}. w_i is the weight of I-th observation. The computation is reduced by factors of 10 to 20.
f) Multiclass logistic regression and classification – Gradient-descent boosting algorithm for the K-class problem. The K-trees are induced at iteration m to predict the corresponding current residuals for each class on the probability scale. The regions corresponding to the different class trees overlap, so that the solution does not reduce to a separate calculation within each region of each tree. The algorithm proposed in the paper has an implementation advantage in numerical stability. LogitBoost becomes numerically unstable whenever the value of $w_k(\mathbf{x}_i) = p_k(\mathbf{x}_i)\,(1 - p_k(\mathbf{x}_i))$ is close to zero for any observation x_i. The algorithm's performance is affected only when $w_k(\mathbf{x}_i) = p_k(\mathbf{x}_i)\,(1 - p_k(\mathbf{x}_i))$ is close to zero for all observations in a terminal node. Influence trimming for multi-class procedure is implemented in the same way.

The paper then gives an approach of regularization which is used to prevent overfitting which occurs when population-expected loss starts to increase. Regularizing by controlling the number of terms in the expansion places an implicit prior belief that sparse approximations involving fewer terms are likely to provide better prediction. However, it has often been found that regularization through shrinkage provides superior results to that obtained by restricting the number of components. Including shrinkage into gradient boosting provides two regularization parameters, the learning rate and the number of components. Author then simulates all the algorithms and notes the best fit and best fit value for several shrinkage parameter. The results show that decreasing the learning rate improves performance. Shrinkage model update at each iteration produces a more complex effect than direct proportional shrinkage of the entire model. Incremental shrinkage produces very different models than global shrinkage. Empirical evidence indicates that global shrinkage provides at best marginal improvement over no shrinkage.

5. Summary of XGBoost: A Scalable Tree Boosting System

The paper mainly focuses on XGBoost, a scalable end to end tree boosting system. Tree Boosting systems are highly effectively and widely used machine learning method used by data scientists to achieve state-of-the-art results on many machine learning challenges. XGBoost is scalable in all scenarios due to several important systems and algorithmic optimizations, it is a quick model exploratory as it learns faster due to parallel and distributed computing.

In the paper they have given an overview of **Tree boosting**. They regularized the learning objective by making w_i to represent score on $i-th$ leaf, unlike decision trees where each regression tree contains a continuous score on each of the leaf. They used the decision rules to classify the decision rules in the trees to classify it into the leaves and calculate the final prediction by summing up the score in the corresponding leaves. They minimized the regularized objective to learn the set of functions. The **regularized objective** is given by

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

The term Ω penalizes the complexity of the regression tree functions. The additional regularization term helps to smooth the final learnt weights to avoid over-fitting.

To score the impurity for evaluating decision trees, we train the model in an additive manner. This is done because the tree ensemble model includes functions as parameters and this cannot be optimized in Euclidean space using traditional optimization model. From the regularized objective the optimal value for the **scoring function** they derive is

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T} \frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

There are two more techniques discussed to prevent overfitting which are **shrinkage** – adding weights by a factor of η after each step of tree boosting, which reduces the influence of each individual tree and leaves space for future trees to improve the model. The second technique is **column subsampling** i.e. feature sub-sampling which speeds up computations of the parallel algorithm.

There are numerous split finding algorithms, and some are discussed in the paper. The authors mention basic **exact greedy algorithm** which enumerates over all the possible splits which is powerful if the data is sorted according to feature values, but at the same time it is computationally demanding. So, they add an **approximation factor** ε .They use a rank function where there are $1/\epsilon$ candidate points. To solve this problem, they introduced a distributed weighted quantile sketch algorithm that can handle weighted data with a provable theoretical guarantee. They added a default direction in each node to make the algorithm aware of the **sparsity pattern** in the data. To add the direction in each branch there are two options: either

they learn it from the data or treat the non-presence as a missing value and learn the best direction to handle the missing values.

The system design was very efficient as they stored their data in **blocks.** The data was already sorted and needed to be computed only once before training. This helped in computing the exact greedy algorithm on the block to find the most optimal split. This also helped when using approximate algorithms, as it considered multiple blocks as a subset of rows in the dataset. This made the **quantile finding step** work linearly as it had to scan over the sorted columns in a linear scan. Collecting statistics for each column can be parallelized, so that a parallel algorithm can be used for split finding.

They performed experiments on four different datasets which are Allstate,HiggsBoson, Yahoo LTRC and Criteo. They randomly selected subset of the data either due to slow baselines or to demonstrate the performance of the algorithm with varying dataset size. The first dataset used was **allstate insurance**, where the task to predict the likelihood of an insurance. It was used to evaluate the impact of **sparsity-aware algorithm**. Second dataset is the Higgs boson dataset, where the task was to **classify** the events where **randomness** or approximation was tested. The third dataset is the Yahoo learning dataset where the task was to r**ank** the documents according to relevance and the fourth dataset was the criteo log dataset, which was used to evaluate the **scaling property** of the system in the **out-of-core and distributed settings**. They used first three datasets for the parallel settings on a single machine and last one for distributed setting. They evaluated the performance of XGBoost on classification and learning to rank problem.

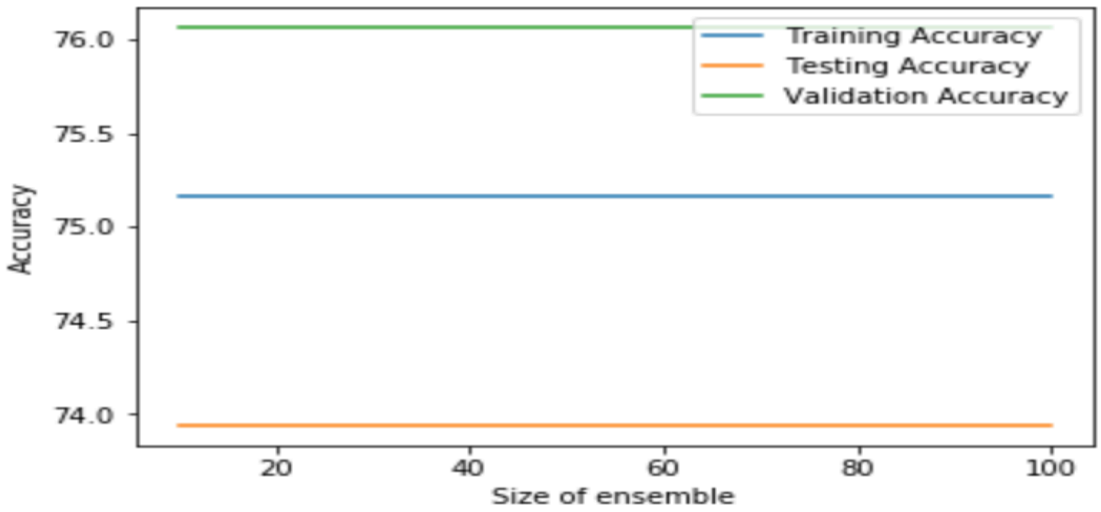
Conclusion: The major contribution of this paper is that –

1) XGBoost is an open source package, portable and reusable.
2) Designed and build a highly scalable end-to-end tree boosting system
3) Introduced a novel sparsity-aware algorithm for parallel tree learning
4) XGBoost handles all the sparsity patterns in a unified way.
5) Time complexity of the system is $O(Kd\|\mathbf{x}\|_0 + \|\mathbf{x}\|_0 \log B)$, where B is the maximum number of rows in each block, K is number of trees, d is the depth and $||x_0||$ denote the number of non-missing entries.
6) Cache access patterns, data compression and sharding are essential elements for building a scalable end-to-end system for tree boosting.

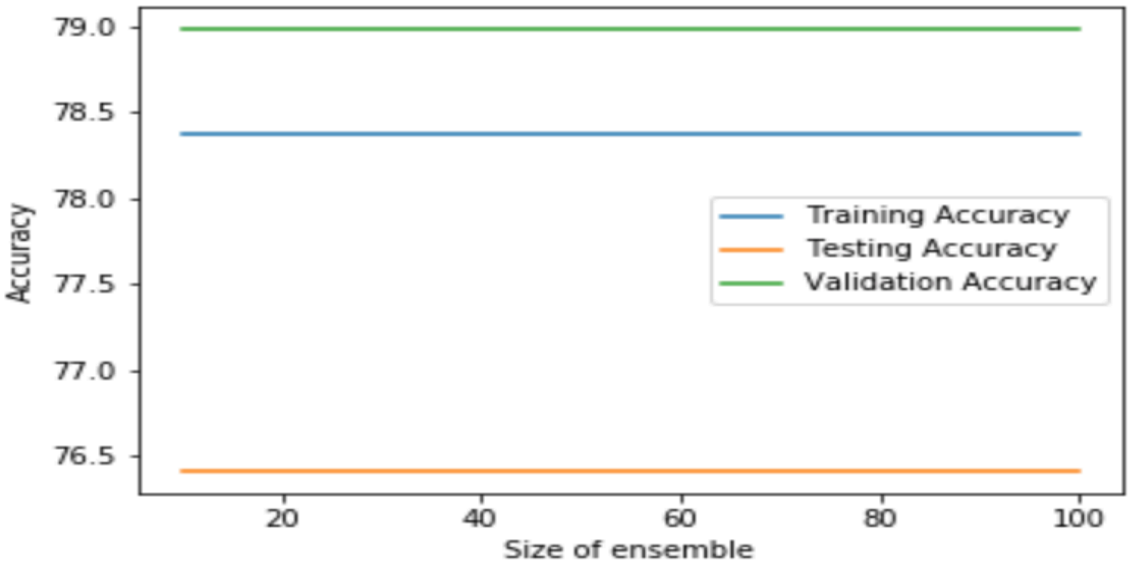
6. a)

The Bagging experiment was conducted for various depth and size of ensemble values. Following are the results obtained for Bagging: -

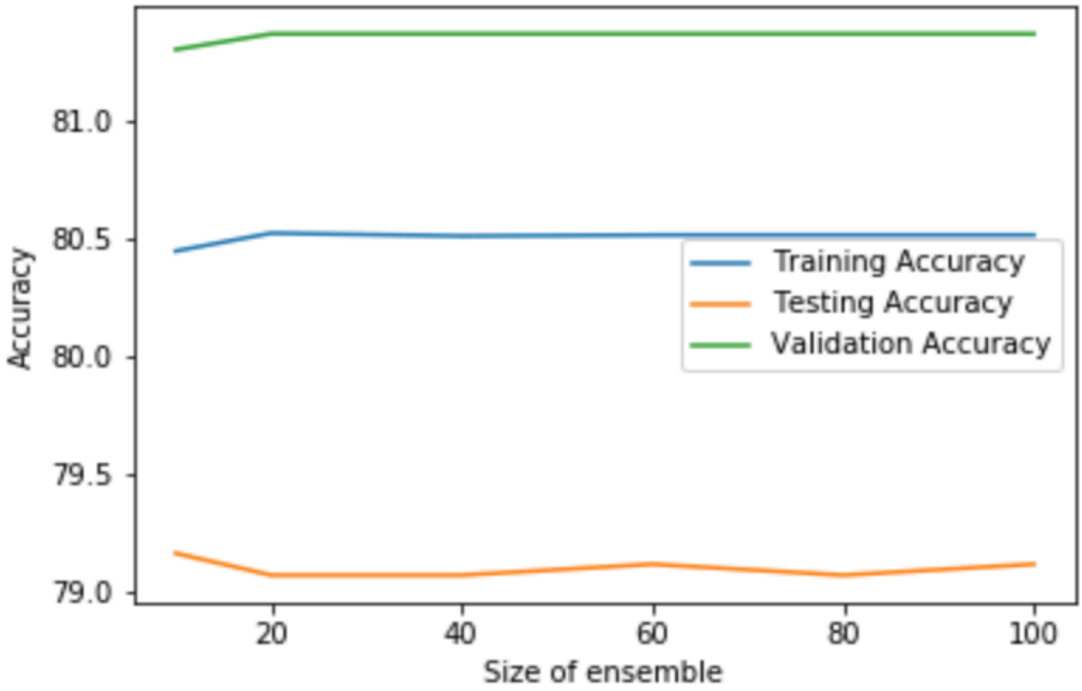
For Depth =1



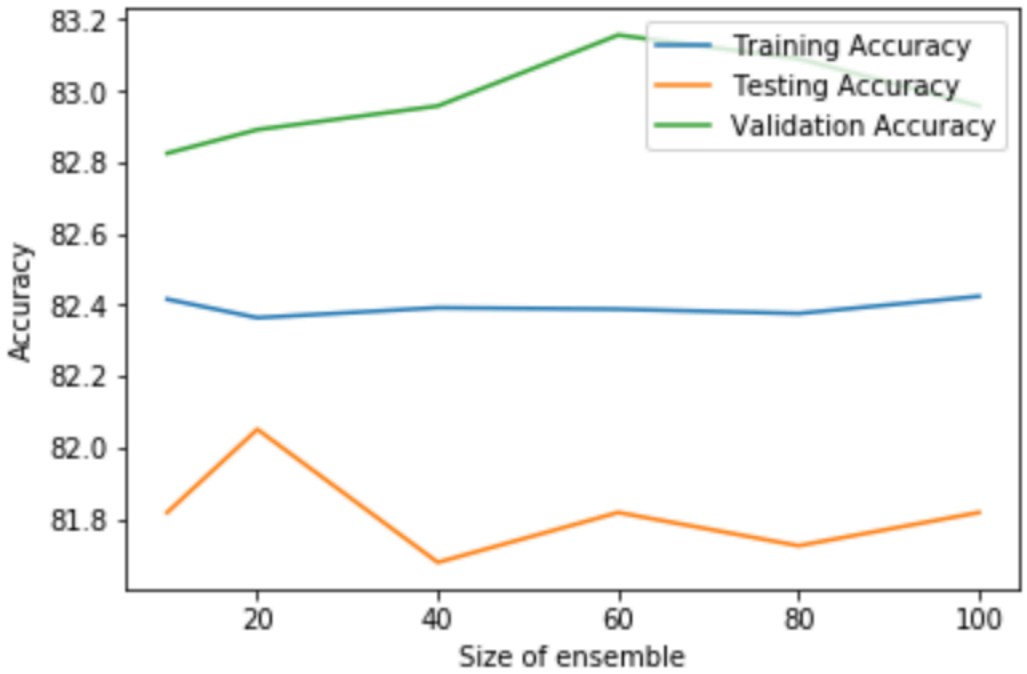
For Depth = 2



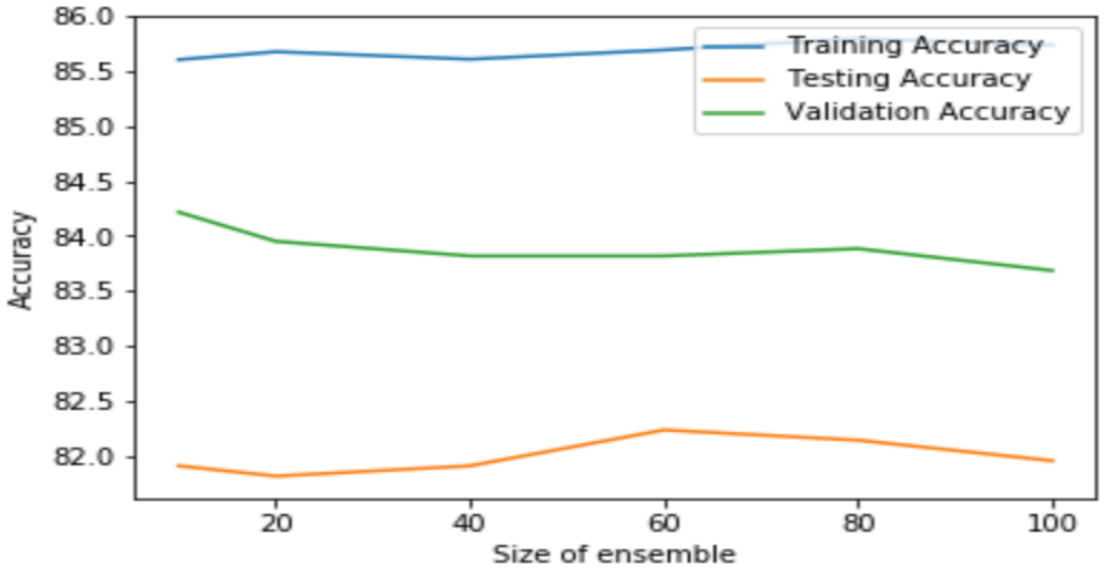
For Depth = 3



For Depth = 5



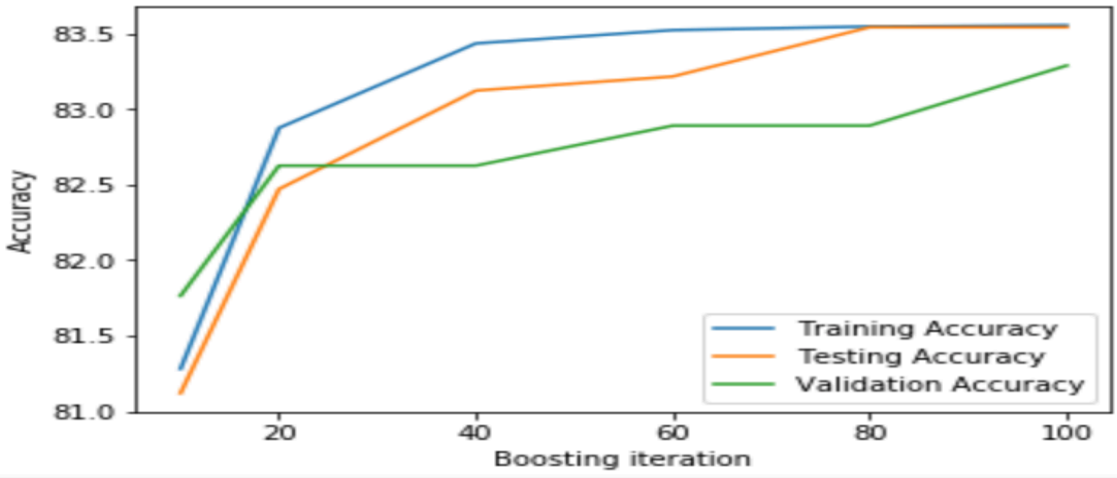
For Depth = 10



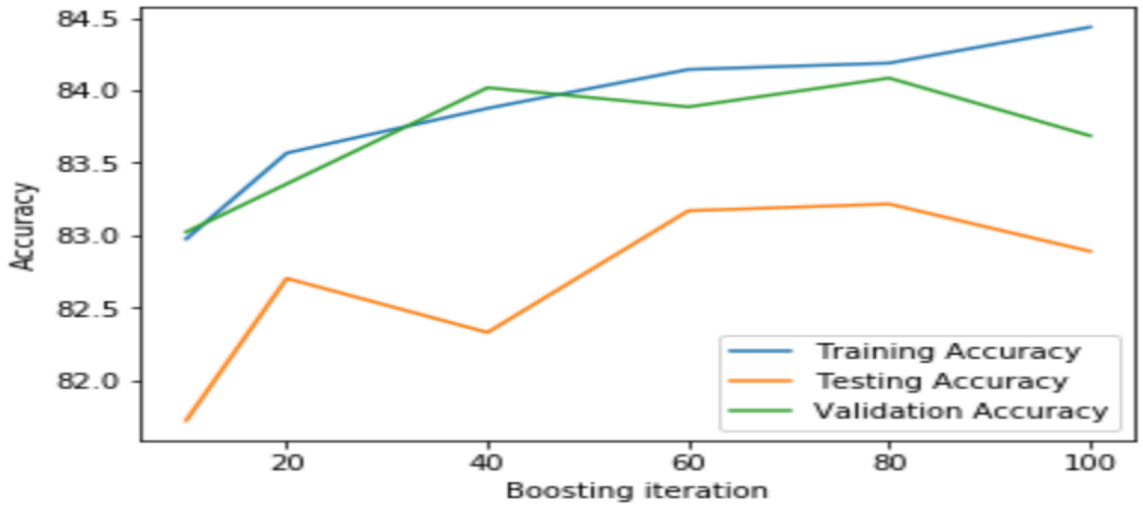
Bagging reduces variance in decision trees. As the depth is increased it can learn more and the results show that in further depths, heuristics were better to find the optimum nodes, hence the accuracy was increased. The size of ensemble gives us a greater number of different trees for computation. Hence accuracy increases as we increase ensemble size. But after a point it starts to overfit or trains redundant data.

b)
The Boosting experiment was conducted for various depth and size of ensemble values. Following are the results obtained for Boosting: -

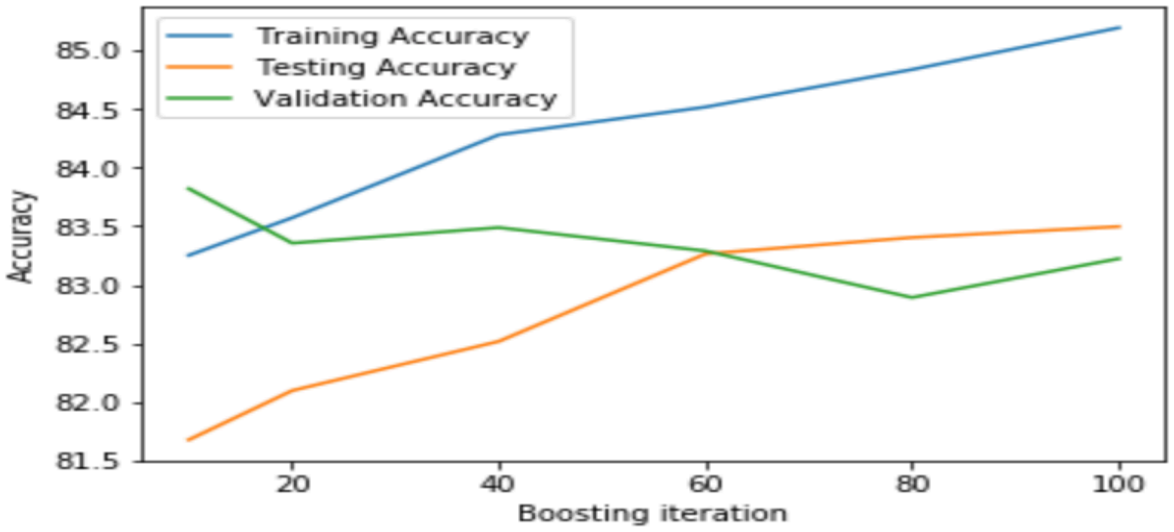
Depth = 1



Depth =2



Depth = 3

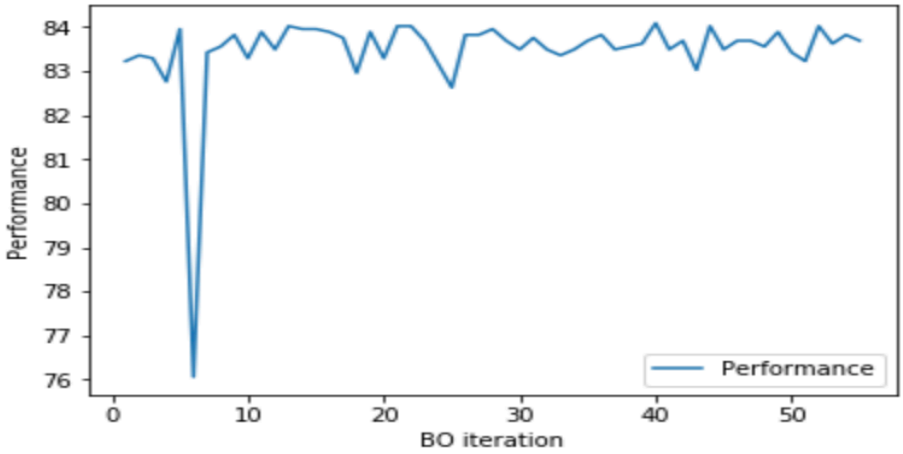


Boosting is used to convert weak learners to strong learners which was achieved in most of the cases as the testing data had better accuracy than validation. Test accuracy improves with the size of the ensembles. The principle of increasing weights for misclassified data and decreasing for data points that are correctly classified is maintained as the training accuracy never decreases.

7.)

Using Bayesian Optimization, hyperparameter tuning was performed and best hyperparameter were extracted. The values for best hyperparameters selected for Bagging by the software were [83.22281167108754, 83.35543766578249, 83.95225464190982, 84.01856763925728, 84.0848 8063660478] and the graph obtained was

It was used to get global optima value for each iteration. It gives us the best value which can be used for bagging to get best accuracy while using decision tree. It also provides with max-depth and number of estimators which should be used as a parameter.



And for Boosting was: -

[82.82493368700266, 83.48806366047745, 83.55437665782493, 83.6870026525199, 84.08488 063660478]

It was used to get global optima value for each iteration. It gives us the best value which can be used for boosting to get best accuracy while using decision tree.