Assignment 3 (COL 774)

By Ritvik Vij (2017CS50417)

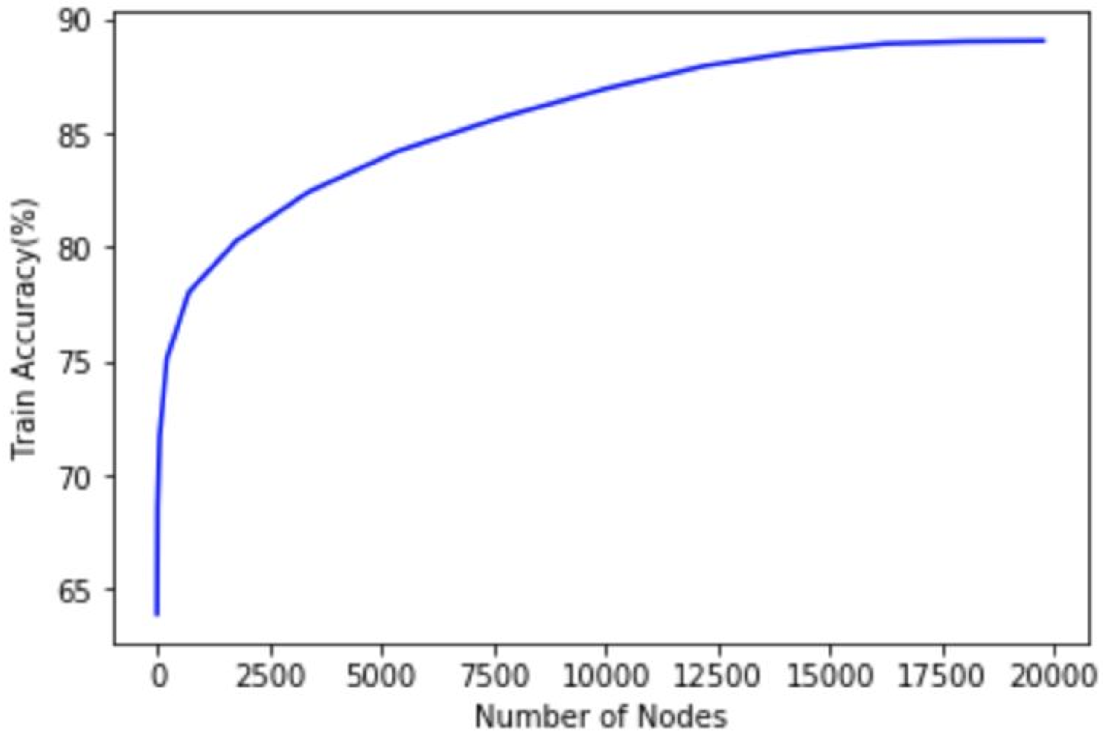
1. Construction of a decision tree

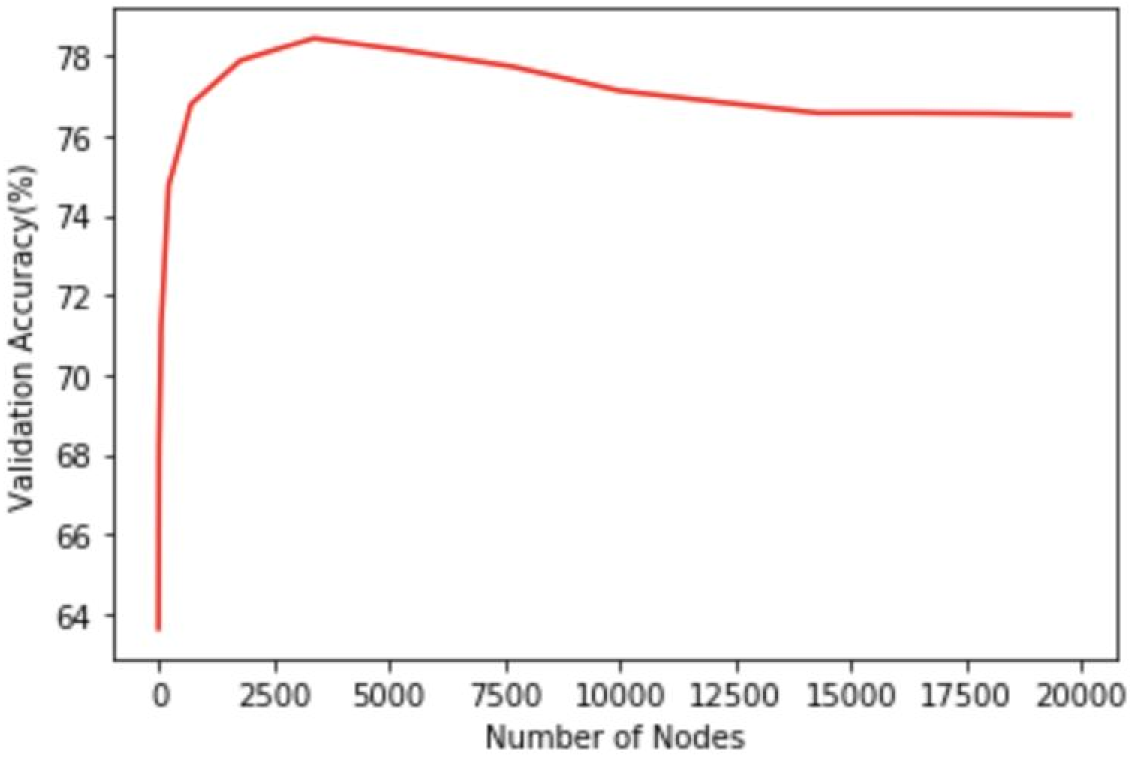
In the preprocessing step, a lib xclib was provided to us which creates a sparse matrix initially to read and fill the data filling in the values of all the attributes. I made my own functions to do I/O, by initially making a sparse matrix(np.zeros()) and then filling the values of the attributes.

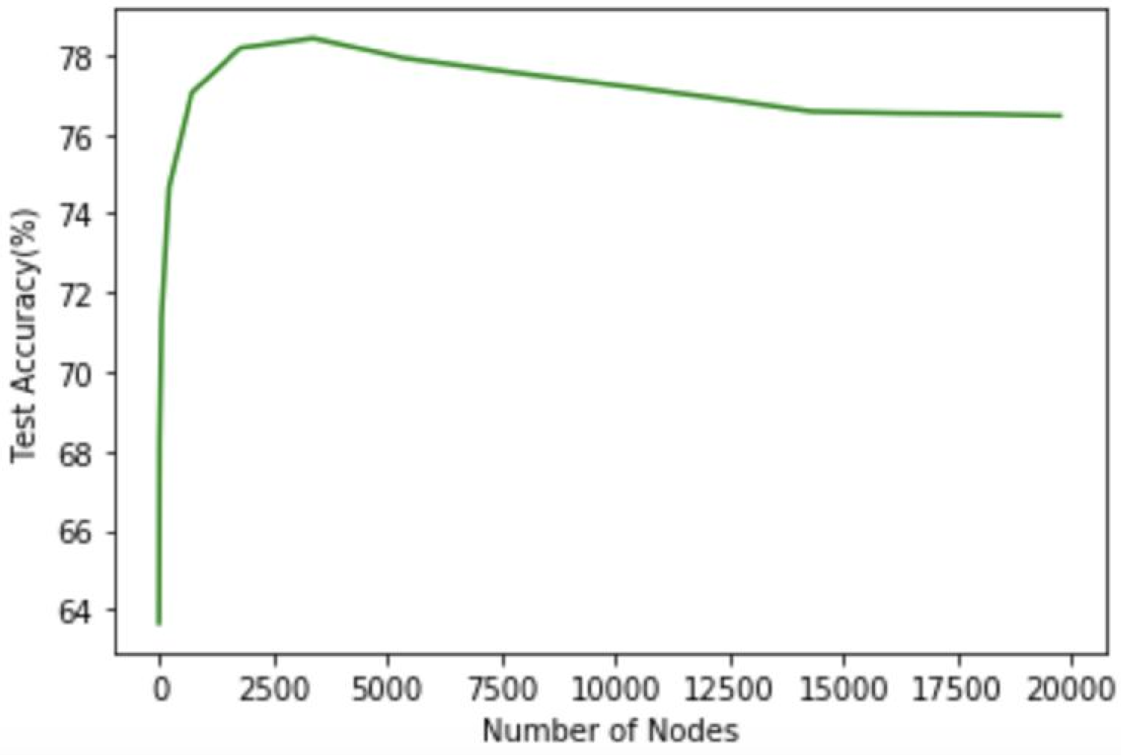
In my implementation of the tree, i have used a parameter of the tree class, max_depth which doesn't allow the tree to grow beyond a certain depth. It is noticed that the tree stops to grow around depth 30 with the number of nodes just south of 20k(19756). In my implementation, to make the growth of the tree more effective and efficient, i have ignored attributes whose atleast 3/4th values are 0(empty). Splitting on them will not be logical. This increases my run time by almost 3x and brings down train accuracy by 0.7%(Totally worth it!!!).

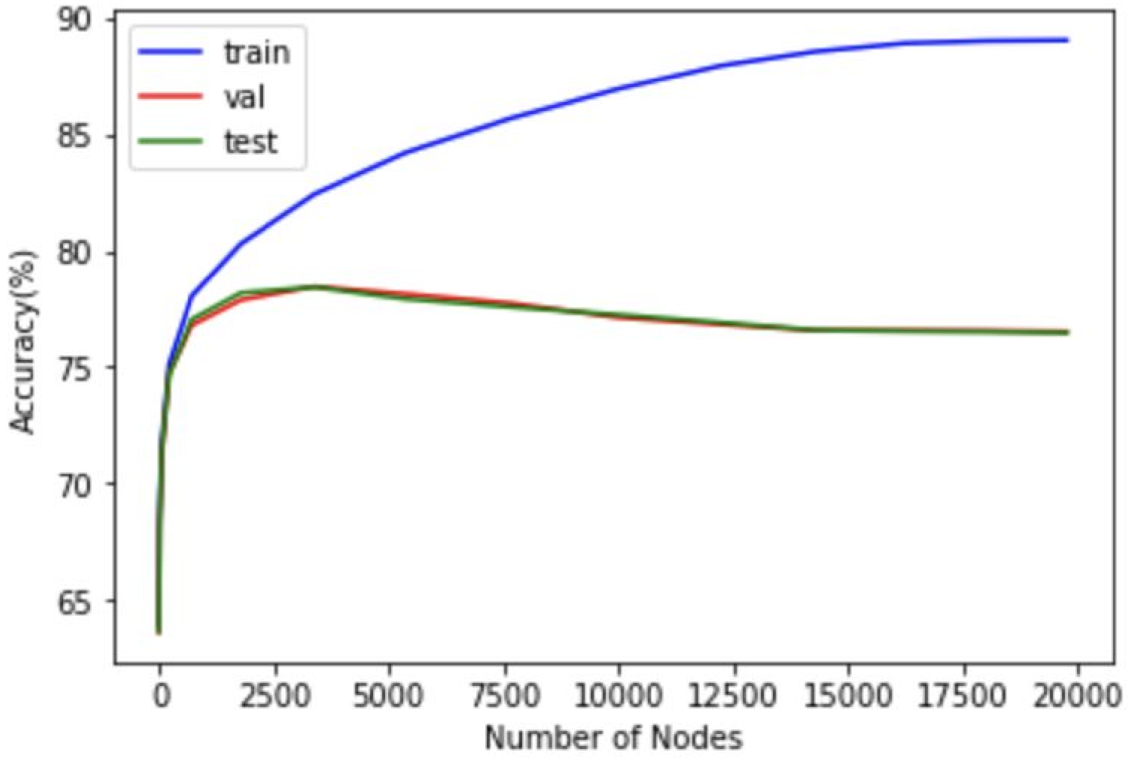
Please refer to the graph for number of nodes on the x axis, the number of nodes is after each depth is increased by a value of 2 (2, 4, 6, 8, 10…….30).

From the graphs we can easily observe that as we increase the number of nodes, the training accurary increases steadily as the model fits, but beyond a point the model overfits because of which after that point the validation accuracy and the test accuracy decreases. To deal with overfiiting, we have done pruning in the next part.



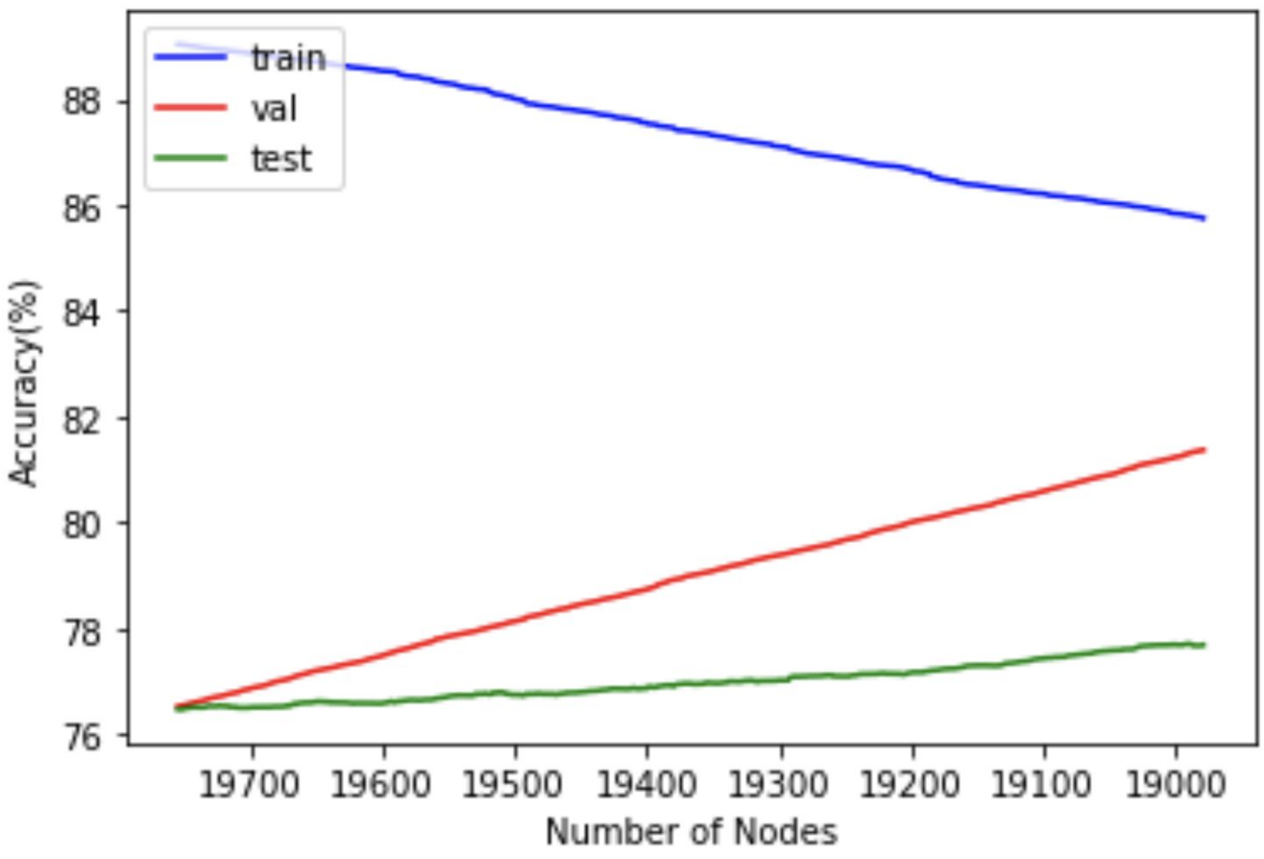

Validation Accuracy(%)
64
66
68
70
72
74
76
78
0
2500
5000
7500
10000
12500
15000
17500
20000
Number of Nodes


Test Accuracy(%)
64
66
68
70
72
74
76
78
0
2500
5000
7500
10000
12500
15000
17500
20000
Number of Nodes



2. <u>Pruning of a fully constructed tree</u>

The fully constructed tree (tree at a given depth) is dealt with pruning to deal with overfitting of the training data. Pruning was done in a bottom up manner, post order traversal of the tree was done, leaf nodes were maintained as it is and for all the internal nodes, the left and right subtrees are removed and stored. If the validation accuracy increases on this removal, the node is said to be pruned. If the validation accuracy worsens, the tree is brought back to it's original state. Pruning is also done as previously, iterating from 2 to 30 for the depth and pruning and checking for each tree constructed with the given



depth. The number of nodes pruned to be precise are .The plots of training, validation and test accuracy is plotted as above . The training accuracy is 85.749%, the validation accuracy is 81.369%(Exactly what we aimed for) and the Test accuracy is 77.683%(Around 1% improvement).

3. **Random Forests**

Random Forests are extensions are decision trees, where we grow multiple decision trees in parallel on bootstrapped samples constructed from the original training data. The range of parameters with which we have created the random forests are (a) n estimators (50 to 450 in range of 100). (b) max features (0.1 to 1.0 in range of 0.2) (c) min samples split (2 to 10 in range of 2). GridSearchCV with Cross validation argument as 2 was used to search over the space of parameters. The argument njob=-1 uses all the threads available to run this process. Using this, the implementation took 2.5 hours to run on a Tensor Processing Unit (google colab).

The results of the implementation are

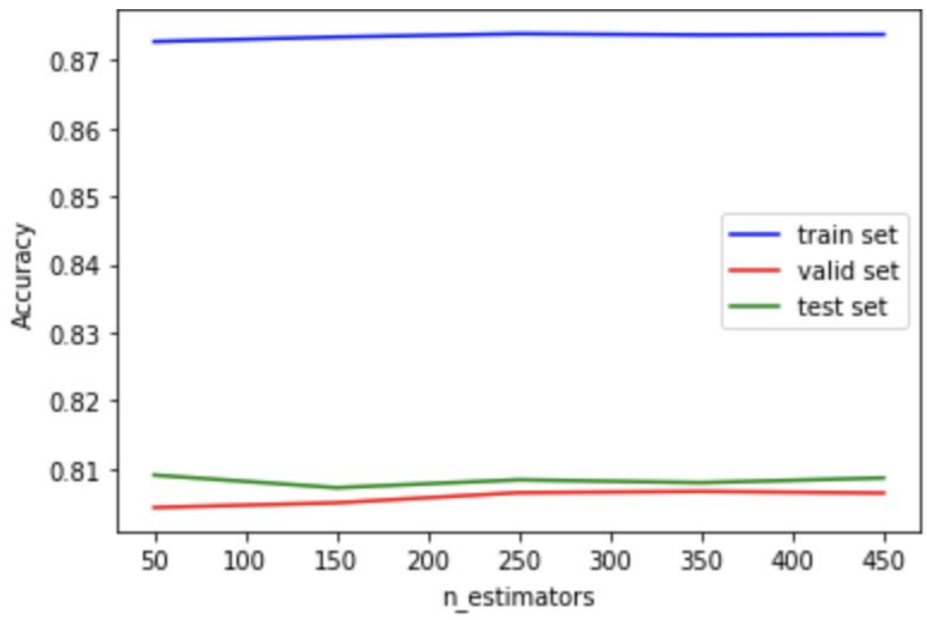
1. The training accuracy reduces down to 87.3472%
2. The validation accuracy increases mildly as compared to the pruning implementation to 80.6137%
3. The test accuracy increases to 80.73%
4. The OOB Accuracy is 81.1342%

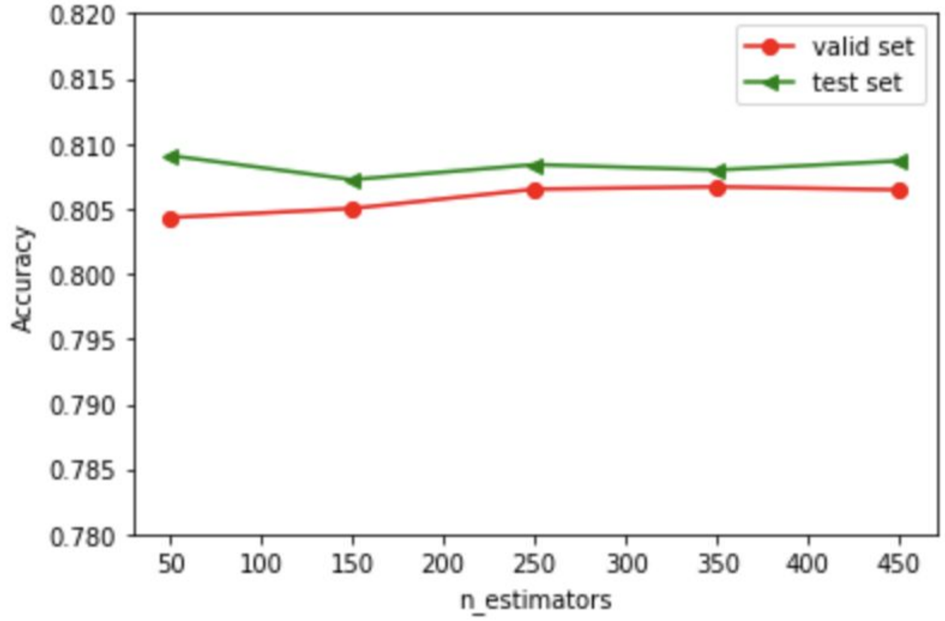
The optimal parameters that GridSearchCV has found by Searching the parameters based on the best oob_score:

1. N_estimators : 350
2. Max_features : 0.1
3. Min_samples_split : 10

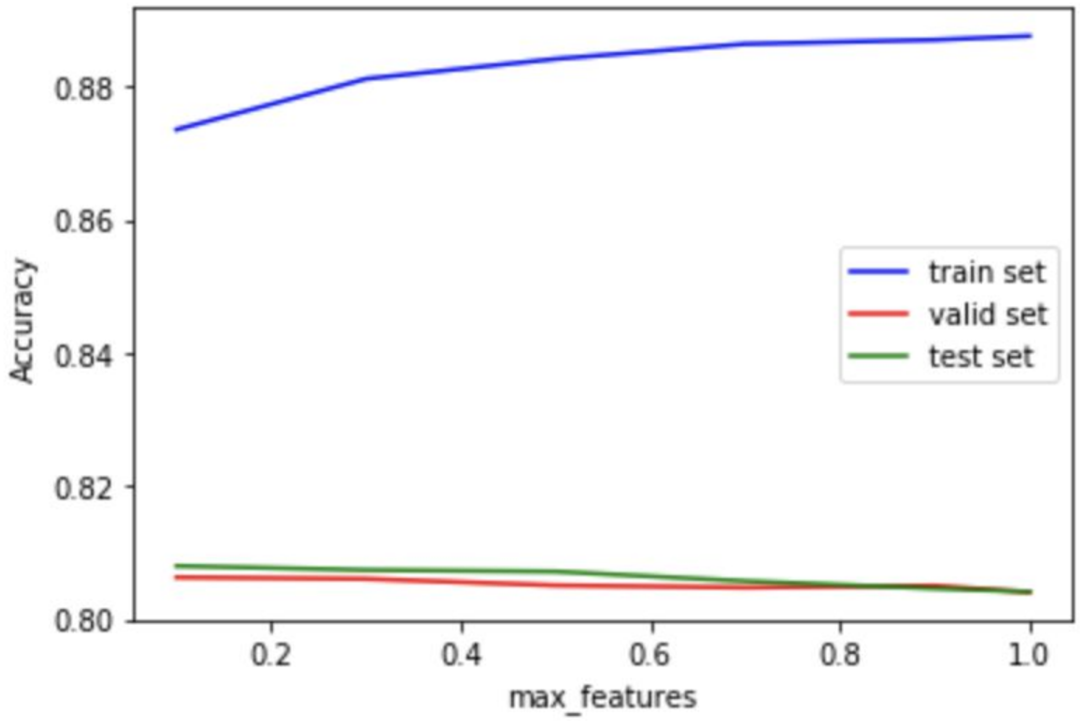
4. Parameter Sensitivity

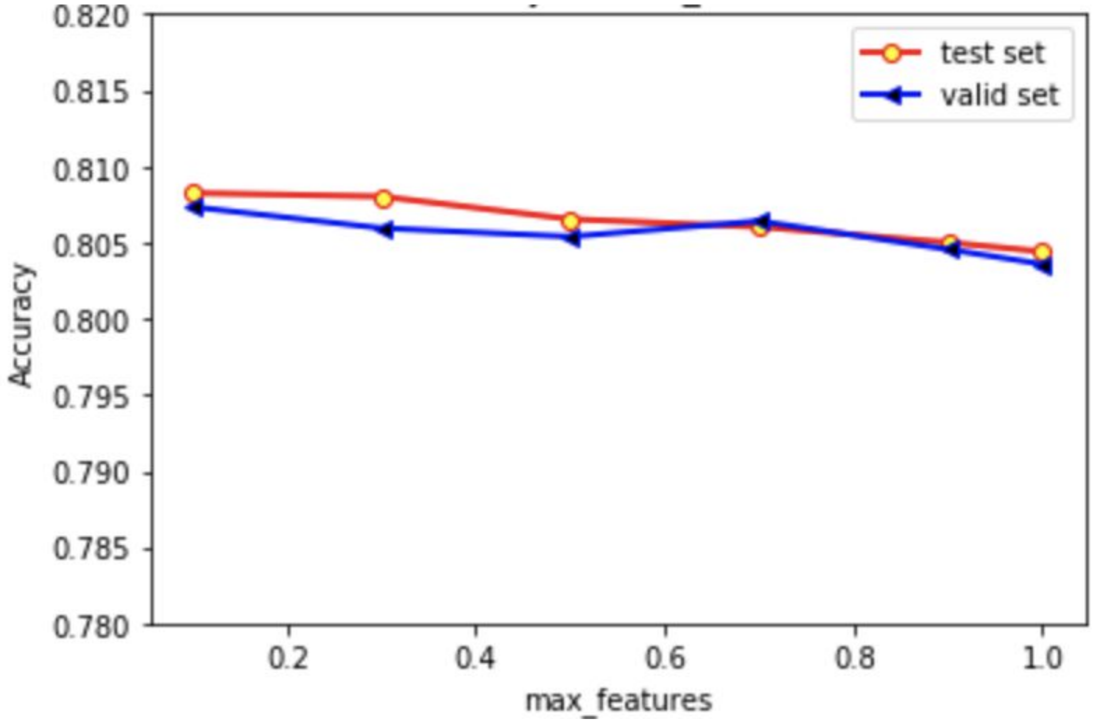
Initially n_estimators were varied keeping the list of other parameters constant. The plot below shows that the variation isn't much by varying this value and hence n_estimators isn't much of a sensitive parameter.



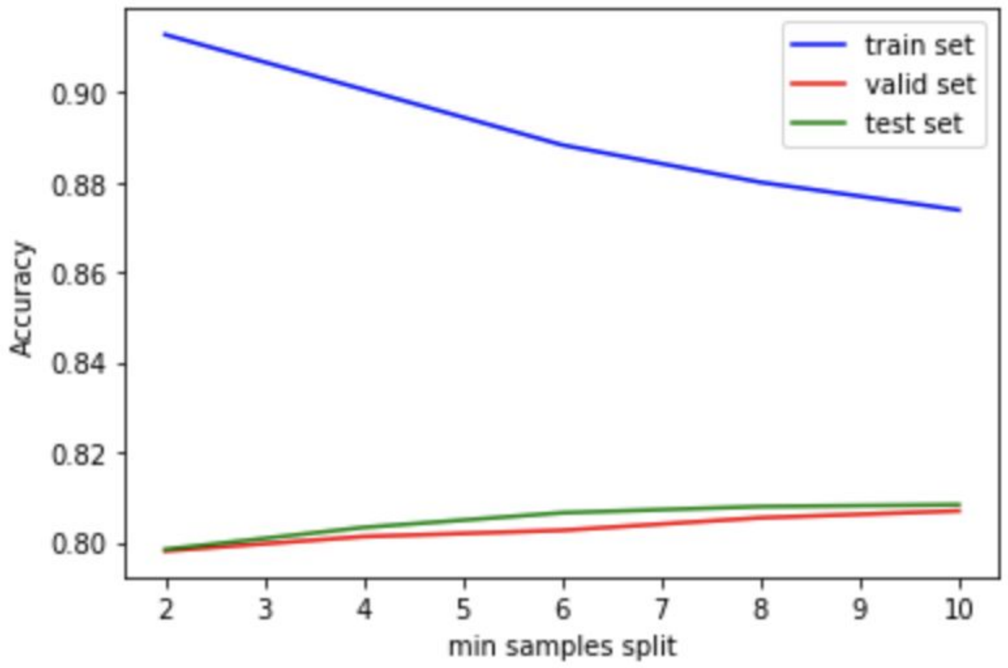


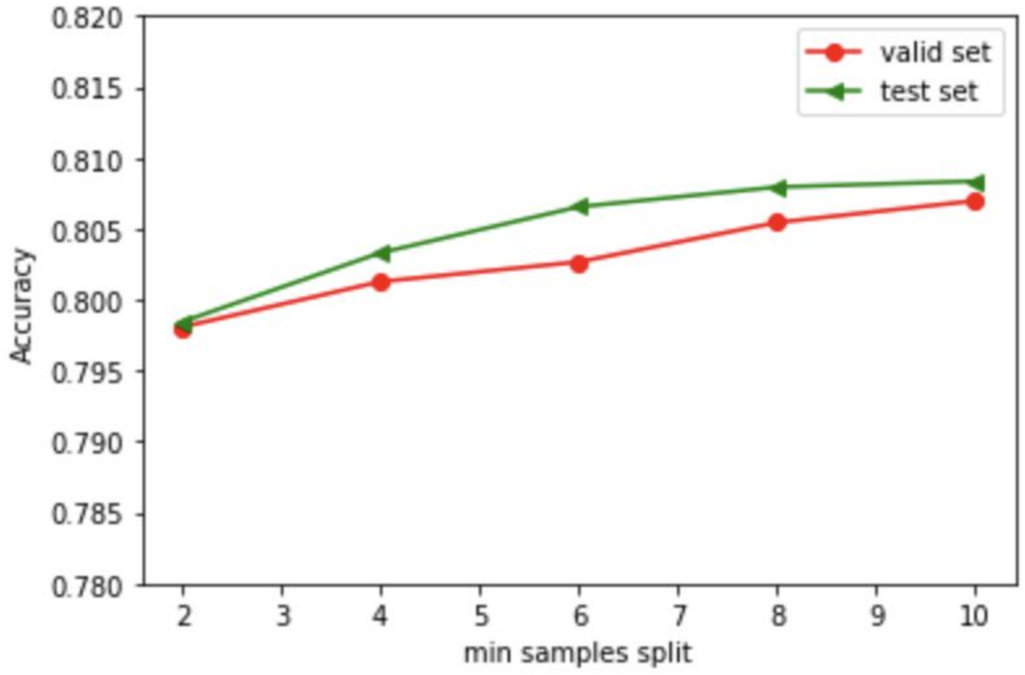
Now Max_Features are varied and the plot are a shown below.





We see that max_features, as we increase the value the training and test accuracy both decrease whereas the training accuracy increases steadily.





The figures above are of min_samples_split. This feature according to the accuracy is the most sensitive and increases test and val accuracy with increasing value. The contrary can be said for the training accuracy.

We can conclude by saying that the accuracy is most sensitive for min_samples_split and the least for n_estimators. N_estimators increases the number of trees whereas splitting feature increases the split hereby decreasing the overfitting on the training data which explains the behaviour of the sensitivity.