1a) Random forest algorithm is implemented in random_forest.py by using decision_tree.py algorithm implemented as part of Assignment1 with modification to take feature set based on '--max-features' parameter.

How to run: random_forest.py takes two parameters

1) -t (--number-of-trees) number of trees to construct in random forest (default : 10)
2) -f (--max-features) feature selection set to take the best split feature (default : sqrt)

    sqrt : max features will be sqrt (total_features)

    log2 : max features will be log2 (total_features)

    float : if its fraction b/w 0 and 1 then features set will be that fraction of total features

It gives out the output as accuracy and time taken for custom and library implementations.

It also plots the oob and test error rate vs feature selection set.

Sample output:

%run random_forest.py –t 10 –f sqrt

Accuracy with custom implementation 0.9414

Time taken for custom implementation 27.04 secs

Accuracy with library implementation 0.9460

Time taken for library implementation 0.07 secs

1b) After running few times by varying m value, it's been observed that accuracy increases when m is increased till some limit (sqrt(features) ~ 8) and then it drops out. This substantiates that when we increase feature selection set (m) then randomness in trees decrease and bias increases.

Samples runs:

1) *%run random_forest.py –t 10 –f 0.02*

    NUMBER_OF_TREES : 10 FEATURE_SELECTION_SET: 1

    Accuracy with custom implementation 0.9388
    Time taken for custom implementation 10.79 secs

    Accuracy with library implementation 0.9480
    Time taken for library implementation 0.04 secs

2) *%run random_forest.py –t 10 –f log2*

NUMBER_OF_TREES : 10 FEATURE_SELECTION_SET: 6

Accuracy with custom implementation 0.9454
Time taken for custom implementation 19.82 secs

Accuracy with library implementation 0.9480
Time taken for library implementation 0.05 secs

3) *%run random_forest.py –t 10 –f sqrt*

NUMBER_OF_TREES : 10 FEATURE_SELECTION_SET: 8
Accuracy with custom implementation 0.9500
Time taken for custom implementation 29.08 secs

Accuracy with library implementation 0.9500
Time taken for library implementation 0.07 secs

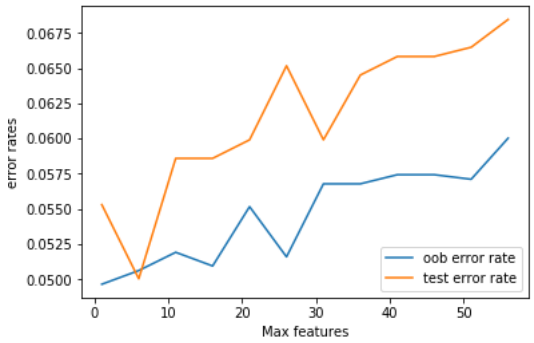
4) *%run random_forest.py -t 10 -f 0.4*

NUMBER_OF_TREES : 10 FEATURE_SELECTION_SET: 23

Accuracy with custom implementation 0.9342
Time taken for custom implementation 71.90 secs

Accuracy with library implementation 0.9381
Time taken for library implementation 0.18 secs

1c) In the below figure, oob error rate is also plotted for feature set. Out of bag error is calculated based on mean prediction error for each training sample with prediction based on trees where this sample is not present. We can observe that after doing multiple trails test error and oob error increases when m is increased.



For 2 & 3, jupyter notebooks are submitted. Please consider them as solutions