4 Random Forests

4.1 Own Implementation of Random Forest:

Observations from Code:

```
Accuracy score with own random forest :  0.9029688631426502
--- Time taken --- 37.53448295593262 seconds ---

Accuracy score with inbuilt random forest :  0.9312092686459088
--- Time taken --- 0.052702903747558594 seconds ---
```

- Used Test train split to split the Data into training and test sets.
- The implementation part of the Code can be found in the zip folder.
- The Accuracy obtained with Developed Random forest is close to the accuracy obtained with Inbuilt Random forest.
- Time taken for developed random forest is far greater than that of Inbuilt Random forest Classifier.
- The Inbuilt Random forest classifier is more optimised in all aspects than our code.
- These are the conclusions obtained after running the code for several times.

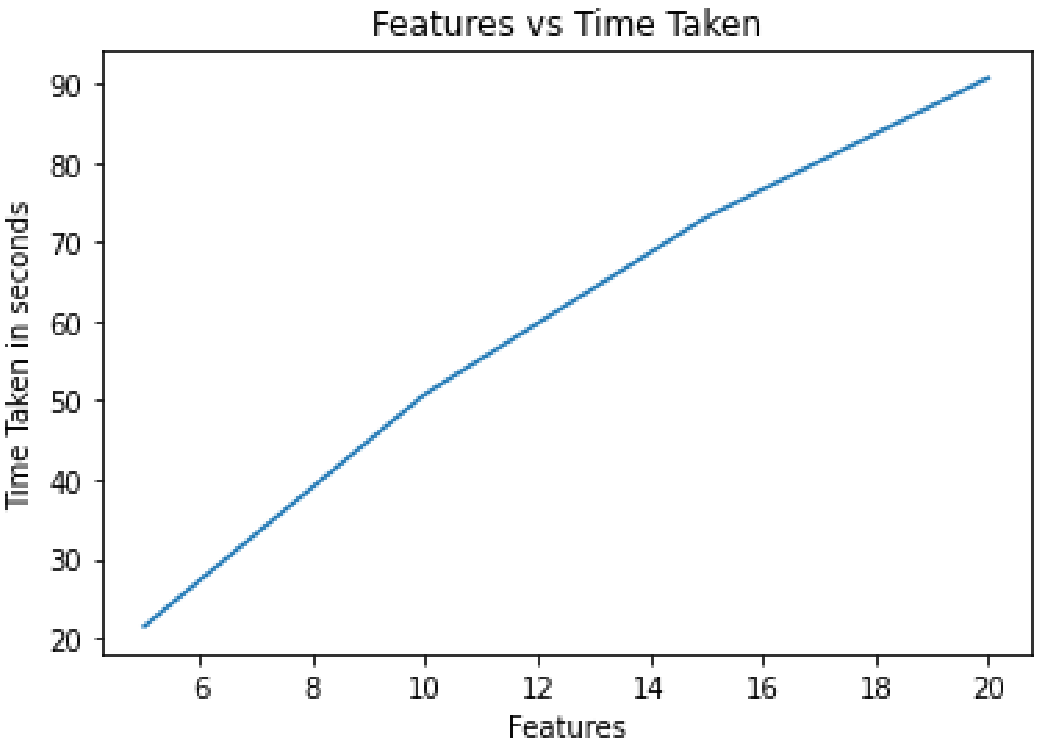
4.2 Variation of sensitivity with Number of Features

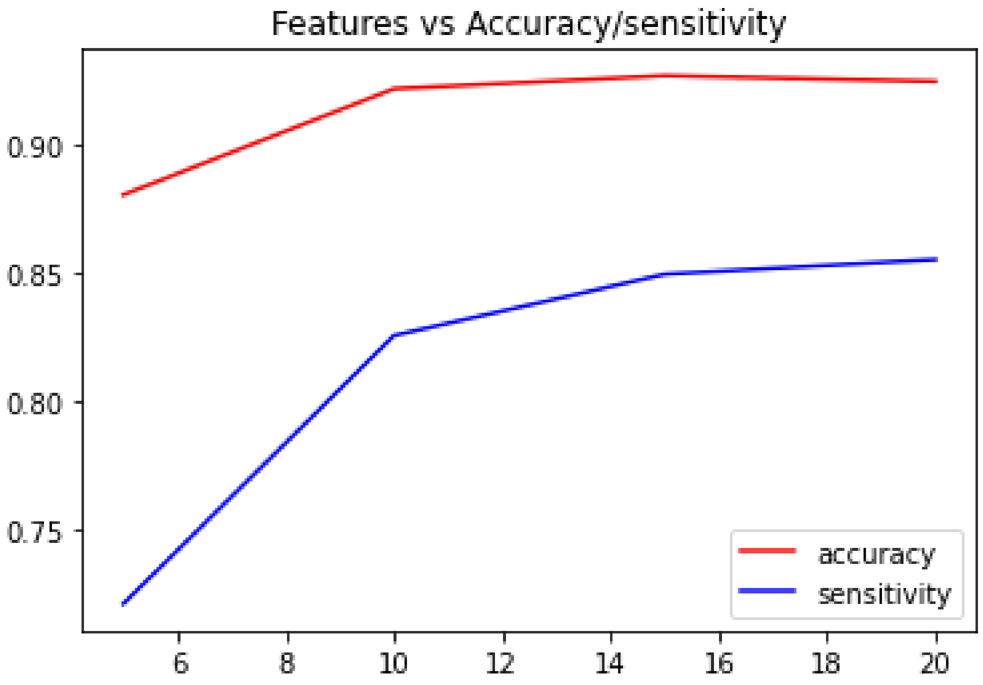
- Please see the code in the zip folder.

Observations from Code:

```
Number of features :  5 , Accuracy :  0.8805213613323678 ,
Time taken :  21.47656774520874 s, Recall Score :  0.7213114754098361

Number of features :  10 , Accuracy :  0.9217958001448225 ,
Time taken :  50.81852197647095 s, Recall Score :  0.8257575757575758

Number of features :  15 , Accuracy :  0.9268645908761767 ,
Time taken :  73.12491822242737 s, Recall Score :  0.8495726495726496

Number of features :  20 , Accuracy :  0.9246922519913107 ,
Time taken :  90.66081619262695 s, Recall Score :  0.8553113553113553
```

Visualization:


Features vs Time Taken
Time Taken in seconds
90
80
70
60
50
40
30
20
6
8
10
12
14
16
18
20
Features


Features vs Accuracy/sensitivity
0.90
0.85
0.80
0.75
accuracy
sensitivity
6
8
10
12
14
16
18
20

Observations:

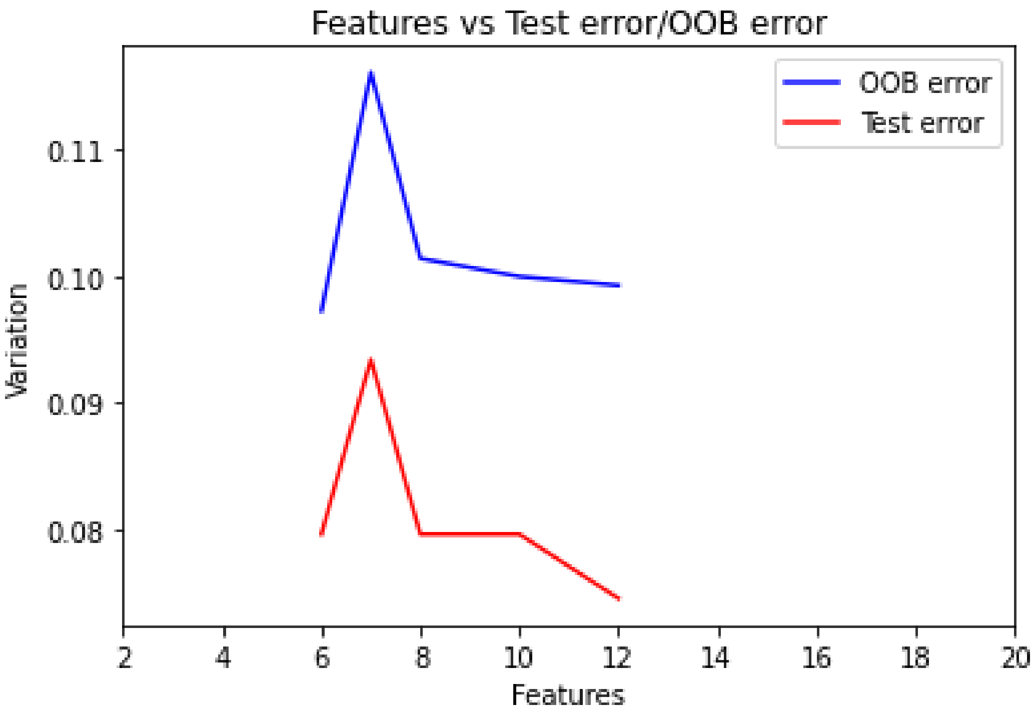
- With increase in features, Time taken is increasing linearly.
- With increase in features, Accuracy observed is increasing.
- With increase in features, Sensitivity is increasing.
- sensitivity is Calculated using `recall_score`
- Accuracy is always higher than sensitivity as observed from the graph.
- So with increase in features, Accuracy, sensitivity and time taken are increasing. To get a reasonable time, accuracy, sensitivity, the average of start and end is a viable option.

4.3 Exploring OOB error and Test error

Observations from code:

```
Number of Features :  6
OOB error :  0.09924550203134064
Test error :  0.07458363504706733

Number of Features :  7
OOB error :  0.09994222992489887
Test error :  0.07965242577842147

Number of Features :  8
OOB error :  0.10136336692353287
Test error :  0.07965242577842147

Number of Features :  10
OOB error :  0.11601884570082455
Test error :  0.09341057204923964

Number of Features :  12
OOB error :  0.09723040659988214
Test error :  0.07965242577842147
```

Visualization:



Observations:

- It is clear from the graph that test error decreases as features increasing.
- From the graph, OOB error also seems to be decreasing as features are increasing.
- At any point of time and with any number of features, OOB error is always more than the test error.
- The graph is observed for many times and every time new skewed values are obtained.

5 Pre-Processing and Gradient Boost Classifier

5.1 Pre-processing

- Removed columns having NaN values
- Removed Unique Columns in the data set
- Identify the columns which has a value which is predominant from the rest.
- Drop the identified columns

- Identify the columns which has data types other than python float
- Label encode the columns, (One hot key encoder also works but yielding almost same values)
- Split the data into test data and train data.
- Now the data is pre processed and ready for further evaluation.

5.2 Gradient Boosting Classifier

5.2.1 Best Accuracy,precision,recall :

Many models were built, best recorded values are:

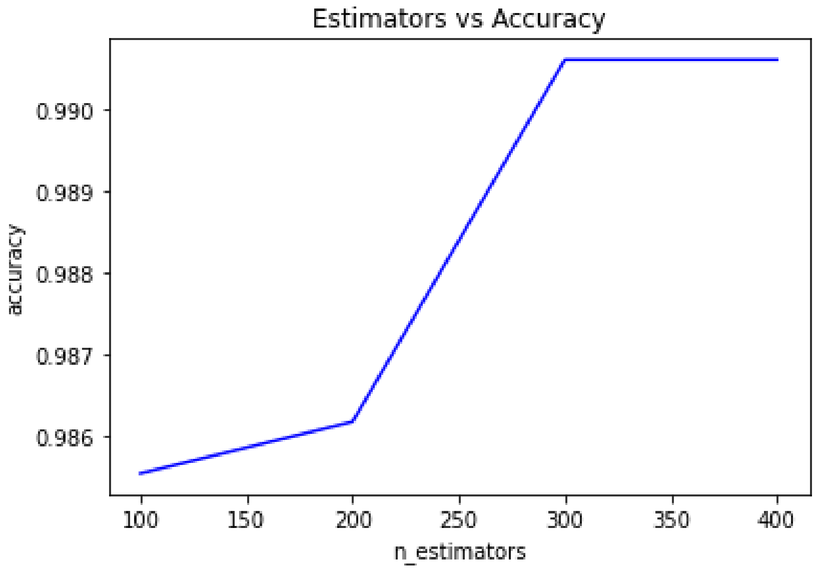
```
Best Recorded Accuracy : 0.9931256238101282
Best Precision :  0.9902622310276867
Best Recall :  0.9712843880195519
HyperParameters : N_estimators = 500, Learning_rate = 0.6, max_depth = 10
```

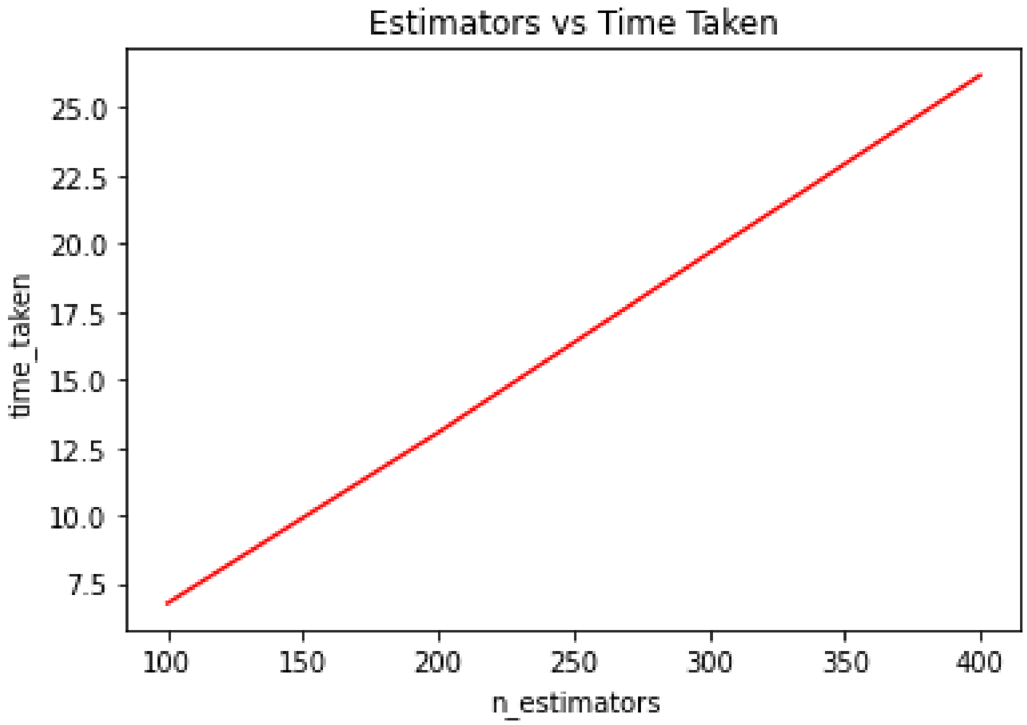
5.2.2 Change in Accuracy/Time vs Estimators

Observed values:

```
Estimators = [100,200,300,400]
Accuracy :  [0.9855311587764944, 0.986165757075771,
    0.9906079451707069, 0.9906079451707069]
Time Taken :  [6.787179231643677, 13.06709623336792,
    19.666242599487305, 26.19341540336609]
```

Visualisation:





5.2.3 Decision Tree Comparision

Accuracy, precision, recall using Decision Tree:

```
Accuracy :  0.9854042391166392
Precision :  0.9714219913751554
Recall :  0.9689849424269265
```

Comparision:
Clearly, gradient boost classifier is out performing Accuracy, precision and recall when compared to Single decision tree.

$\LaTeX$ generated document

THE END