Question 4:

Brief Description of Implementation:

As mentioned in Random Forest is build on top of previous assignment 1 model. Random Forest contains n decision trees with m features used for best split. Each tree learns based on a different subset of training data obtains by choosing samples randomly with replacement with a subset of size m features chosen randomly. Each tree learns on randomly subsampled data and gives it prediction for a particular sample. Vote of each tree is considered and output of the forest is taken as majority vote for a particular class. This vote is then compared to the original label and then accuracy of the forest is calculated.

For calculation of OOB score https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710 this link is used as a reference for calculating the OOB error. Implementation keeps a record of the samples chosen for each tree within a forest. This set is used to calculate OOB samples for each tree at run for each OOB sample tree that contain same sample as OOB are considered and the sample is given to each such tree for prediction, result is decided by majority vote. As soon as an OOB sample is exposed to the tree it is added to selected sample list of these trees as it is no longer an OOB sample thus reducing the number of times same sample is passed. Each result from the trees are then compared with actual labels and score is calculated and labelled as OOB score.

Each forest can return three values corresponding to input number of trees and features those are accuracy sensitivity and oob_score. As shown

```
BuildRandomForest(num_trees=10,num_features=26)
```

```
Building Forest...
Making Predictions...
OOB Score
0.9139987445072191
Accuracy
0.9492753623188406
(0.9492753623188406, 0.9433962264150944, 0.9139987445072191)
```

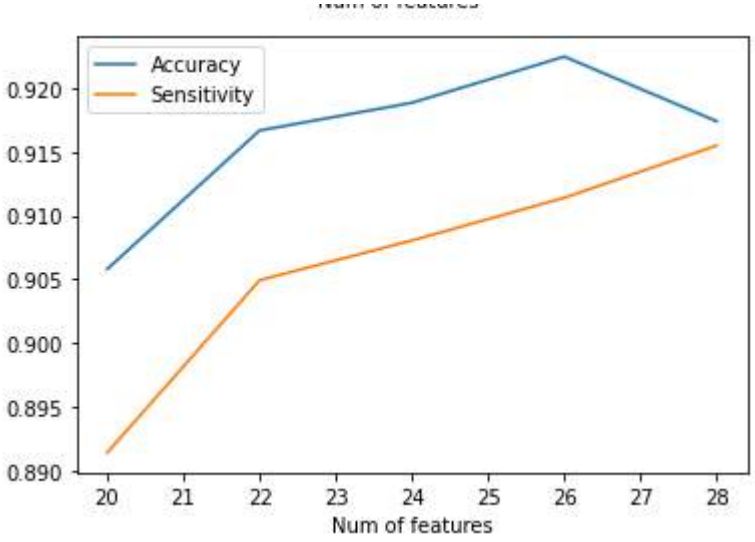
```
[38] BuildRandomForest(num_trees=5,num_features=26)
```

```
Building Forest...
Making Predictions...
OOB Score
0.890846286701209
Accuracy
0.9456521739130435
(0.9456521739130435, 0.9473053892215569, 0.890846286701209)
```

(a) Write your own random forest classifier (this should be relatively easy, given you have written your own decision tree code) to apply to the *Spam* dataset [data, information]. Use 30% of the provided data as test data and the remaining for training. Compare your results in terms of accuracy and time taken with Scikitlearn's built-in random forest classifier. (Note that you can't use in-built decision tree functions to implement your code. You can modify your decision tree code of the Assignment 1, or code a new one, to implement a random forest. You can however use the inbuilt train test split of sklearn to divide the data into train and test.)

	Sklearn(RFC)	**EnsembleRandomForest**
Accuracy	Scikit Learn RFC Accuracy 0.9492753623188406 OOB score 0.9211180124223602 num_trees=10	Building Forest... Making Predictions. OOB Score 0.9139987445072191 Accuracy 0.9492753623188406 num_trees=10 num_features=26
Time	Few seconds	8m 45s Time changes if less number of trees and features are passed it takes lesser time.

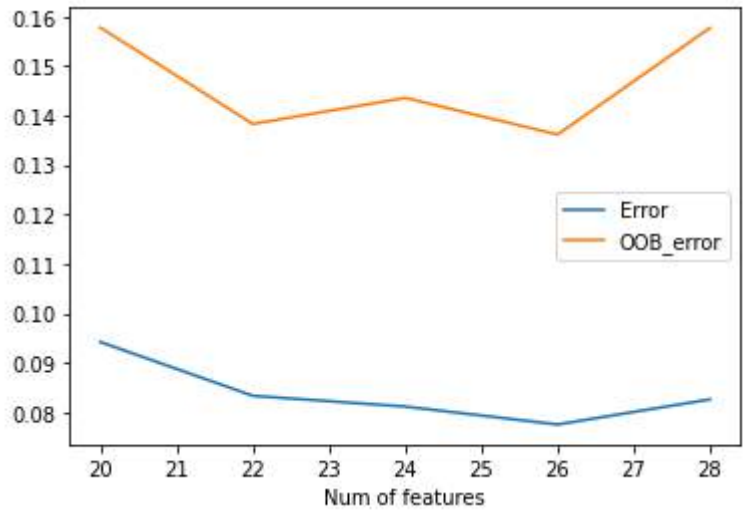
(b) Explore the sensitivity of Random Forests to the parameter *m* (the number of features used for best split).



As we can see by varying num_features with the same tree accuracy generally increases nut with increase in features the time required for building the tree also increases it also decreases the independence between the trees which is required in random forests thus there has to be a balance of num_features used for each tree as generalizing accuracy should increase but at the same time independence among the trees should also be there and speed of computing each individual should be high

The plot is a combined plot for accuracy and sensitivity to the parameter m used for best split

(c) Plot the OOB (out-of-bag) error (you have to find what this is, and read about it!) and the test error against a suitably chosen range of values for *m*. (Use your implementation of random forest to perform this analysis.)



As from the figure it is visible that OOB error is more than the test error. The observation of over estimation of OOB error for 2 class classification problem can be observed in the graph as mentioned in the article https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6078316/

The OOB error is calculated as 1-OOBscore the method of calculating score is already described the range of values chosen are 2*sqrt(n) - 4 to 2*sqrt(n) + 4 and test error is also plotted for each value of m within the above range the range is chosen by experimenting with num_features values keeping them as sqrt(n)/2, sqrt(n), 2*sqrt(n) in that order as 2*sqrt(n) gave the maximum value the range is chosen that way.

```
Building Forest...
Making Predictions...
OOB Score
0.7393364928909952
Accuracy
0.7557971014492754
Building Forest...
Making Predictions...
OOB Score
0.841624685138539
Accuracy
0.8847826086956522
Building Forest...
Making Predictions...
OOB Score
0.8521684077239633
Accuracy
0.9260869565217391
```

Accuracy and OOB score in that order m =[sqrt(n)/2, sqrt(n), 2*sqrt(n)] num_trees=10