Assignment - 1
By:- Kartik Garg(2019101060)

CLEANING OF DATA:

SELECTION:

We have a set of 20 features initially, out of which 4 represent location. To be precise we have 4 cells combined for the location so we do not actually need 3 cells here. So I have first removed those irrelevant features. This is evident from line 7 in the very first code snippet of the ipynb file submitted along with this report.

Subsequently removing the features like other forms of magnitude except Mw as it is mentioned by the TAs that we have to consider only Mw.

Then the serial numbers allotted to the data rows are irrelevant to my machine learning models as those are really random, there is not any useful data or trend in the serial numbers. So dropping that column too.

We then have these columns like DATE, MONTH, YEAR, UTC, IST. All these columns are trying nothing but to provide the time details of when the earthquake occurred. On thinking, I thought that dropping UTC and IST will help me as one can still get some insights from the trends on the basis of YEAR, MONTH and DATE but UTC/IST is the exact time of the day and it is a very short interval of time to be considered. So using it will mean like I am guessing on the probability of an earthquake to happen at exactly the same or nearly same time at which one of the previous ones did. Same case can also arise with using DATE as a variable as a single day is still a very short span of time but I kept it just to observe what happens.

Then I saw that we have both latitudes and longitudes along with location names, which is basically one and the same thing so I decided to work with numerical data types that are latitudes and longitudes as those describe a location more precisely rather than names.

Now that's all about selecting a set of features from all the ones that were given to us.
So i now have the following set:
YEAR, MONTH, DATE, Mw, LAT, LONG, DEPTH, REF

PREPROCESSING/CLEANING:

The very first point in cleaning was to handle the NA values and the noise, noise in the sense that in the field of YEAR where we expect numerical values there might be some random string in the dataset.

So I looked for a way online in order to check if the values given to me in the pandas series are all integers or not by replacing any commas in the input. Any other character literal like 'a' will make it a NA. After converting the data in this way i fill all the NA values with 0 as for MONTH and DATE they signify nothing as both the fields have domain starting from 1 so adding one more value zero in it to take care of the noise. For YEAR it sounds reasonable to put 0 as we would need some year value to place there, the other option was to delete all the instances where year value is NA but in performing that I thought that my data quality would reduce so i picked the first idea of placing zeroes.

This kind of preprocessing is done for 4 fields that are YEAR, MONTH, DEPTH, DATE.

For the reference field, I replaced NA values with 'NONE' as this Series was of dtype string object.

Subsequently, I did some data removal precisely noise reduction using the latitude, longitude and the Mw data available. Since only latitude data availability or only longitude data availability is practically of no use as it does not make sense. Followed by the fact that since I am predicting Mw values so the data rows which do not have the recorded data of the magnitude itself are again of no use to my model as the model won't learn anything from it.

At this point the data types are like this,

```
YEAR     int64
MONTH    int64
DATE     int64
Mw       object
LAT      object
LONG     object
DEPTH    int64
REF      object
```

So, now i tried to convert all my data to int datatype. For this i need to convert latitude, Mw and longitude from float to int and ref from string to int.

For REF data type conversion, I designed a map, to map each referrer to an integer value and then for each record I replaced the referrer name by the integer value assigned by me. Subsequently I exported the map in a json file so as to extract it directly from there for future requirements. Can be found in the location "./data/referers.json"

For Mw, on observation i saw that most of the values are restricted to one decimal point only so in a nutshell i converted each float value of type x.y into a two digit value xy.
Eg: 2.3 is converted to 23

Followed to that, the threshold that we were required to choose was lying between 4-5, which now will become 40-50 (In the new shifted system for Mw)

For latitude and longitude, a similar mechanism is followed so describing for latitude. On observing the latitude values in the dataset the regex of the values will be of the type

xy.zCC

Where x,y and z are numbers and CC are characters of the English alphabet.
So firstly I extracted xy.z from this and then converted this float value to an integer of the form xyz.
For eg: 78.9 will be converted to 789

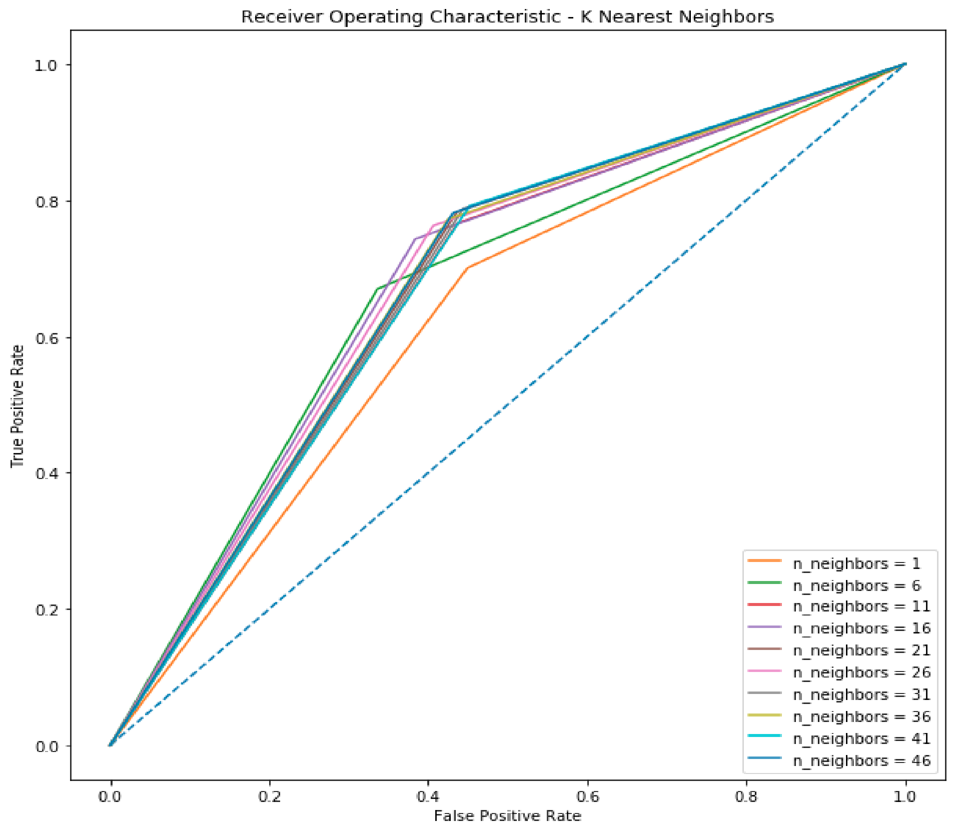
So this was all about preprocessing and data cleaning.

REQUIRED ANALYSIS:

Threshold chosen is 4.3, can be seen as variable METAVAL set at 43 (In the shifted system that i used as described above)
Train-Test split size used is 0.7 that is 70% data is used for training and remaining 30% is used for testing.

1.) ROC CURVES:

For knn as a parameter of k (number of neighbors),

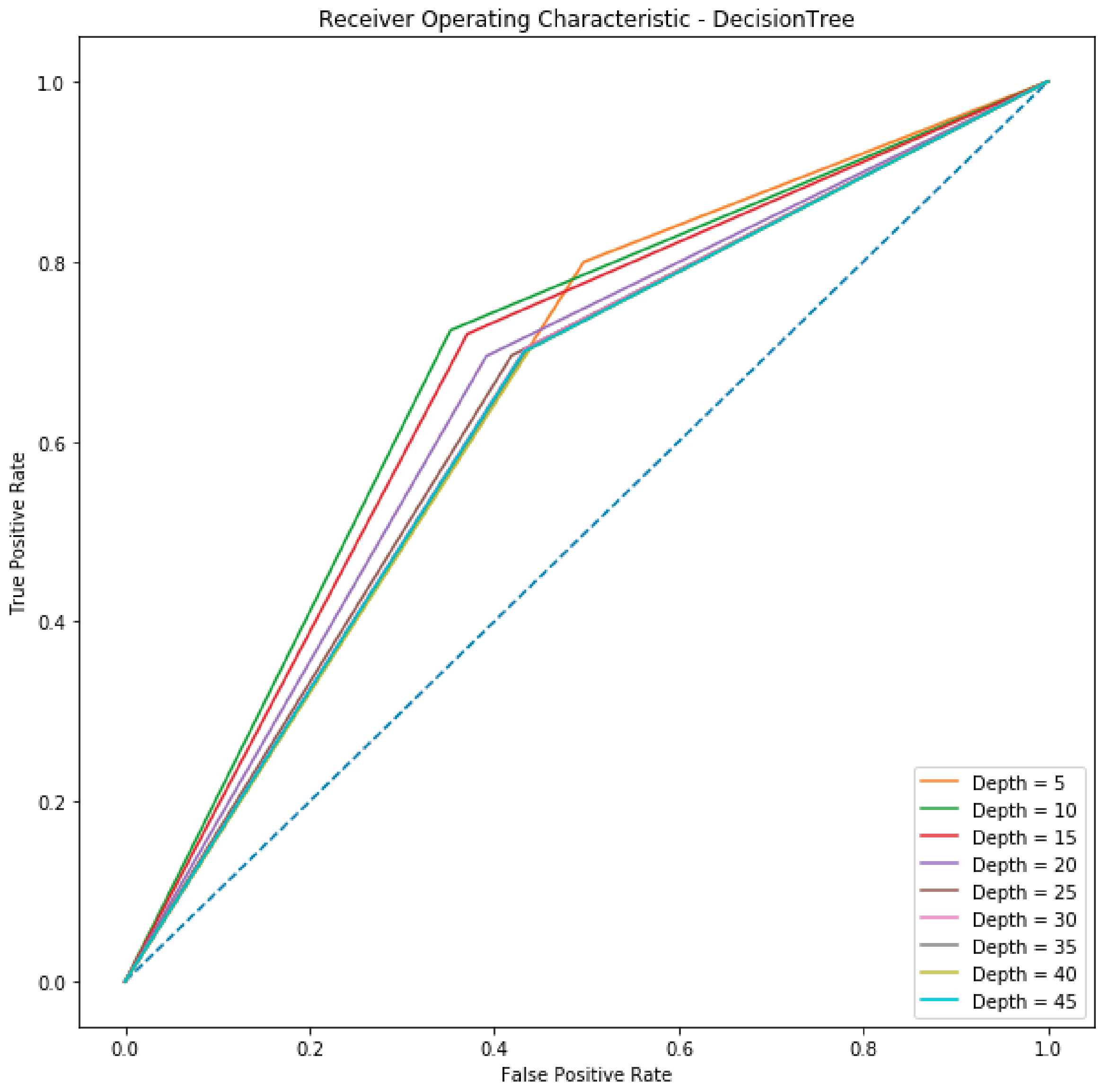


For K = 16, we achieved our maxima as shown above. More details are given in the image below

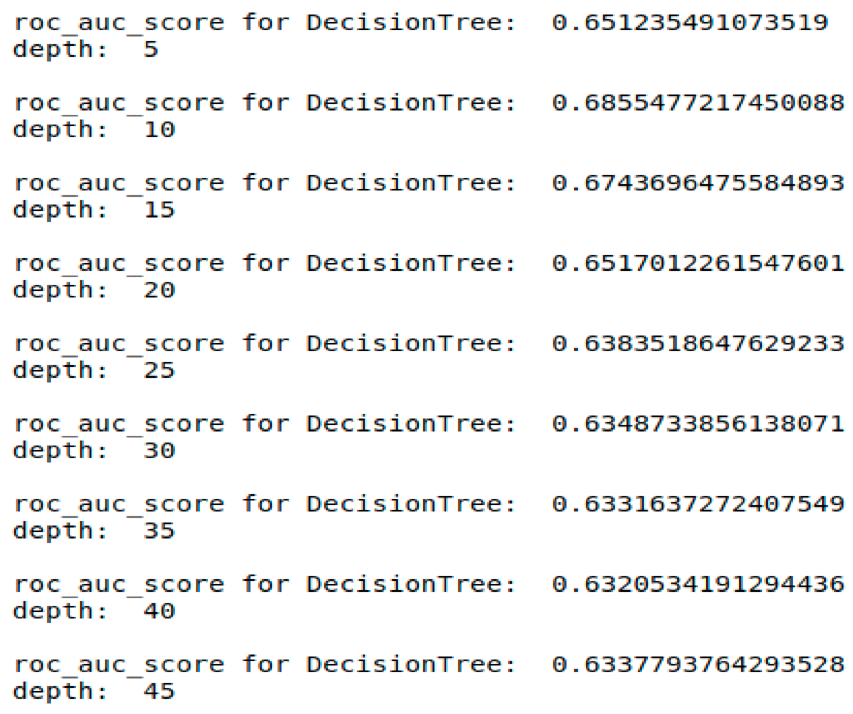
```
 roc_auc_score for nearest_neighbor:  0.62558042189475
neighbors:  1

 roc_auc_score for nearest_neighbor:  0.6667703134335863
neighbors:  6

 roc_auc_score for nearest_neighbor:  0.6646252846098797
neighbors:  11

 roc_auc_score for nearest_neighbor:  0.679602179395246
neighbors:  16

 roc_auc_score for nearest_neighbor:  0.6734070563511353
neighbors:  21

 roc_auc_score for nearest_neighbor:  0.6782614891045365
neighbors:  26

 roc_auc_score for nearest_neighbor:  0.6717909232014738
neighbors:  31

 roc_auc_score for nearest_neighbor:  0.6736535033291593
neighbors:  36

 roc_auc_score for nearest_neighbor:  0.6697412926270067
neighbors:  41

 roc_auc_score for nearest_neighbor:  0.6750894838099247
neighbors:  46
```

Both the above images can be found in the pyplots directory named as
selected_neighbors_kneighbors_best_16.png
roc_scores_kneighbors_selected_nneighbors.png respectively

For Decision Tree as a parameter of pre prune depth,



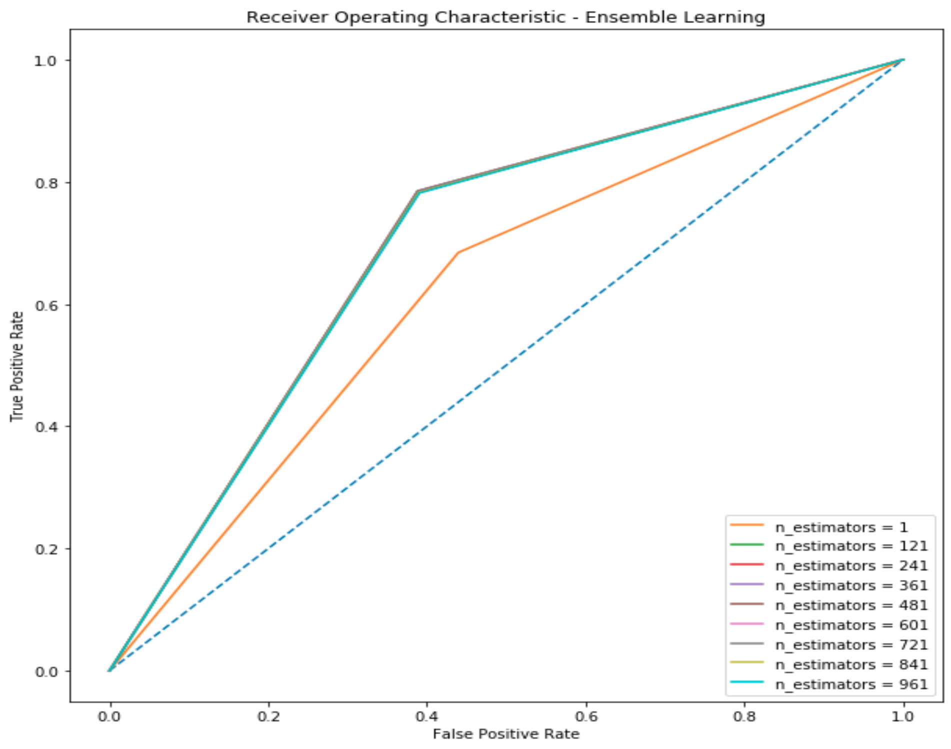
For depth = 10, we achieved our maxima as shown above. More details are given in the image below



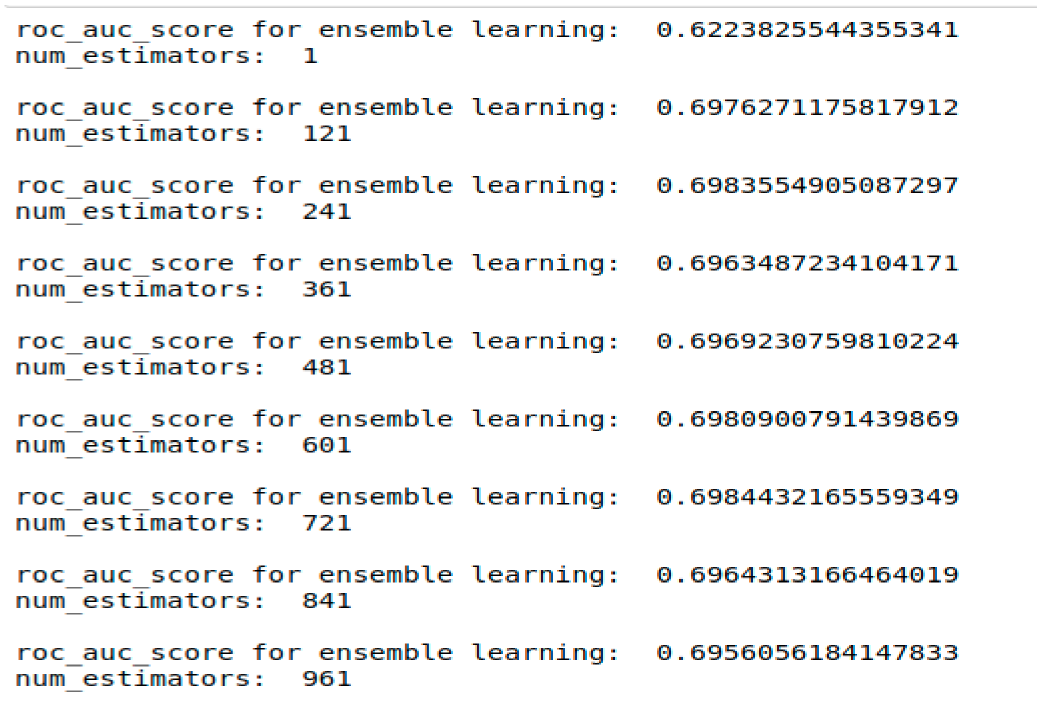
```
roc_auc_score for DecisionTree:  0.651235491073519
depth:  5

roc_auc_score for DecisionTree:  0.6855477217450088
depth:  10

roc_auc_score for DecisionTree:  0.6743696475584893
depth:  15

roc_auc_score for DecisionTree:  0.6517012261547601
depth:  20

roc_auc_score for DecisionTree:  0.6383518647629233
depth:  25

roc_auc_score for DecisionTree:  0.6348733856138071
depth:  30

roc_auc_score for DecisionTree:  0.6331637272407549
depth:  35

roc_auc_score for DecisionTree:  0.6320534191294436
depth:  40

roc_auc_score for DecisionTree:  0.6337793764293528
depth:  45
```

Both the above images can be found in the pyplots directory named as
selected_depth_decision_tree_best_10.png
roc_scores_decision_tree_selected_depths.png respectively

For ensemble Learning as a parameter of the pre prune depth. I used a bagging classifier for ensemble learning which in itself uses a decision tree to further process.



For estimators = 721, we achieved our maxima as shown above. More details are given in the image below



```
roc_auc_score for ensemble learning:  0.6223825544355341
num_estimators:  1

roc_auc_score for ensemble learning:  0.6976271175817912
num_estimators:  121

roc_auc_score for ensemble learning:  0.6983554905087297
num_estimators:  241

roc_auc_score for ensemble learning:  0.6963487234104171
num_estimators:  361

roc_auc_score for ensemble learning:  0.6969230759810224
num_estimators:  481

roc_auc_score for ensemble learning:  0.6980900791439869
num_estimators:  601

roc_auc_score for ensemble learning:  0.6984432165559349
num_estimators:  721

roc_auc_score for ensemble learning:  0.6964313166464019
num_estimators:  841

roc_auc_score for ensemble learning:  0.6956056184147833
num_estimators:  961
```

Both the above images can be found in the pyplots directory named as
limited_extimators_ensemble_learning_best_721.png
roc_scores_ensemble_learning_selected_estimators.png respectively

For ensemble learning I initially went on with trying each value from 1 - 1000, but that was kind of time consuming. Then I reduced this size of 1000 to 40, running for the 40 values I got the maxima and then for visualisation purposes I further reduced that 40 to approx 10.
We see that ensemble learning performs best among all the three cases.

2.) BEST CLASSIFIER

We observe in the above attached graphs that the best classifier according to the ROC values is ensemble learning. The reason being obvious in the sense that ensemble learning provides more randomness and becomes more fair to use. As it does not limit its output to just a single classifier and uses a fairly large number of classifiers and takes the majority vote and gives that as output. Now, this approach is given an upper hand because a single model might be trained wrong or might get overtrained or remain under trained but when a large number of classifiers are considered then this type of unseen difficulties on an average will get nullified.

3.) BEST PARAMETER VALUES

For knn we found the best k to be 16, for decision tree the best depth value came out to be 10 and for ensemble learning the optimal number of estimators for best ROC value was around 700.
For knn, we see that values less than 16 give more of a random result and they are lucky by chance if classified correctly. Because in order to classify properly we need to check for sufficient number of neighbors so that i don't get a premature result and assuming the

KNeighborsClassifier will be using euclidean distance then nearest neighbors mean close depth values, close location coordinates, time of occurrence needs to be close enough.
One idea here is that since we can reason on the factor of time considered here, time can be considered a factor or not, its diplomatic in some sense as on one hand time makes the two earthquakes of similar parameters, one with year as -300 and the other with year as 500, more distant apart but they must be close because they occurred on the same place and has same intensity as before so it is likely to experience a similar kind of earthquake in future too. This approach tells us to drop the YEAR column but i tried that, it does not help much.
Moving on to the decision tree classifier, the optimal depth came out to be 10. I tried building the tree to its maximum possible depth which in this case was 46. On observing the results given by depth = 10 and depth = 46 was that in the latter case it was considering way too many factors, very narrow windows of the parameters for classifying which did make it perform a little bit poorly. Basically it has been an instance of overfitting in cases where depth was 46. On reducing the depth to 10 we found that the ROC values reached a maxima because this depth seemed to divide the intervals for different values better than the other tried values and gave better results. On further reduction we see that the interval size grows much bigger and we start getting bad results. Basically, underfitting is observed.
For ensemble learning, the number of estimators varied from 1 to 1000 and on doing so we found a maxima at around 700. I am not stating the exact value 721 because in this case I didn't actually try all the values but values at an interval of 40. So the maxima came at 721 so the exact optimal value should be around it so stating 700 seems reasonable. The reason for this beng optimal is, choosing a value smaller than 700 leads to a state where the we fall short of diversity basically we can't cover all the necessary details from the training set using that number of estimators so we need to increase the number of estimators whereas after 700 we see that the number of estimators us now this big that

either the estimators start repeating themselves or they learn something that was not intended that is overfitting.

4.) CHOOSE TWO FEATURES

For this part I tried different pairs of features selected from the set of features used initially and I tried to come up with a reason why that is the best pair of features in accordance with the decision tree graph given by the DecisionTreeClassifier.
So on doing so I found that the ROC values reached a maximum value of 0.72 for features YEAR and LONG, along with the fact that the hyperparameters like the depth of the decision tree was set at 10 even for ensemble learning.
On seeing in the decision tree output by the classifier, i observed that the first two levels were mostly based on these two parameters so choosing these two makes sense from there plus we realise that with changing years there has been a trend in the earthquakes reported so that trend is evident from choosing YEAR as one of the parameters. Choosing only longitude but not latitude is something that sounded disturbing to me but one can easily to design a new feature which is a mixture of both of these like

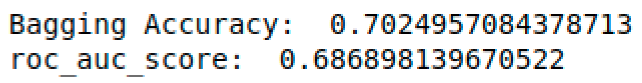
(MAX_LAT_VAL)*LONG_VAL + LAT_VAL

So that disparity can be removed easily.

5.) BEST MODEL

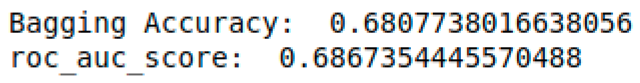
Best model in my case came out to be ensemble learning and the original set of 7 features was used to achieve this.
a.) Feature Processing

For original set of feature we have



```
Bagging Accuracy:  0.7024957084378713
roc_auc_score:  0.686898139670522
```

After dropping REF and DATE, we have



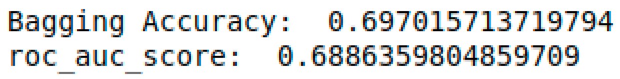
```
Bagging Accuracy:  0.6807738016638056
roc_auc_score:  0.6867354445570488
```

Accuracy has been reduced after the above change which signifies one of the two removed features or both of them has some influence on the decision making. So I added DATE again to the list and I observed a little increment in the accuracy.

Accuracy seemed an increment when REF was dropped and latitude and longitude was combined to get a new feature which is kind of a weighted mean of the two

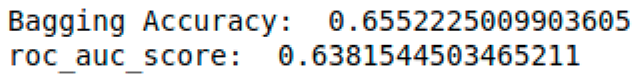
    0.7*longitude + 0.3*latitude

Longitude is given more weight as when I had to select just two features, longitude came out to be a significant one.



```
Bagging Accuracy:  0.697015713719794
roc_auc_score:  0.6886359804859709
```

On combining both latitude and longitude into one, and combining year,month and date into one we see a fall in the accuracy like this



```
Bagging Accuracy:  0.6552225009903605
roc_auc_score:  0.6381544503465211
```

Using just combination of the latitude and longitude as described above was really a waste as my accuracy dropped to 0.61

So I tried this kind of feature processing and in some cases my accuracy did improve but the improvement wasn't significant like the maximum that I went was 0.72.