Data Analytics Assignment 1 Report

Avani Gupta
2019121004

Task
* Using the following features we predict Magnitude

F: {'YEAR ', 'MONTH ', 'DATE', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9', 'Unnamed: 10', 'Unnamed: 11', 'DEPTH (km)','LONG (E)','LAT (N)','LOCATION','REFERENCE'}

Libraries used: sklearn, matplotlib, pandas, numpy

Preprocessing

* columns having too many NaN values like intensity, Unnamed: 16, Unnamed: 17 were deleted.
* LOCATION, REFERENCE were label encoded
* 'LONG (E)','LAT (N)' were pre-processed to replace E, N, and some of junk S, W values with numbers and converting some values like 5'2 as 5+ 2/60 where 5 is degrees '2 is minutes.
ref: https://www.latlong.net/degrees-minutes-seconds-to-decimal-degrees
* All other columns were converted to float
* NaN's were replaced by using fflil method of sklearn
df.fillna(method='ffill')
*** Threshold for magnitude taken as 4.085(median of the dataset)**
Given: For Mw < T, label becomes 0 (no earthquake) and for Mw ≥ T becomes 1 (earthquake)

Obtained Class distribution:

0	27059
1	25930

Train-Test split 70:30

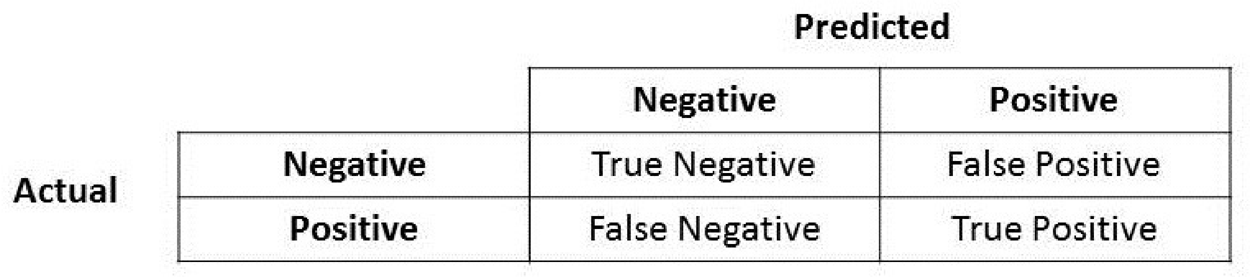
All the reported results are on the test set

We experiment with different class distributions and find the median to be most suitable since it makes the distribution for both classes as similar as possible.
Median intuitively makes sense since it divides an array into two equal parts (left and right)

Models

knn = sklearn.neighbors.KNeighbors(n_neighbors=max_k)
dc = sklearn.tree.DecisionTree(max_depth = max_depth)
en = sklearn.ensemble.BaggingClassifier(n_estimators=max_est)
Note: The notation of knn/KNN for KNeighbors, dc for Decision Tree and ensemble or en for BaggingClassifier is used throughout the report.

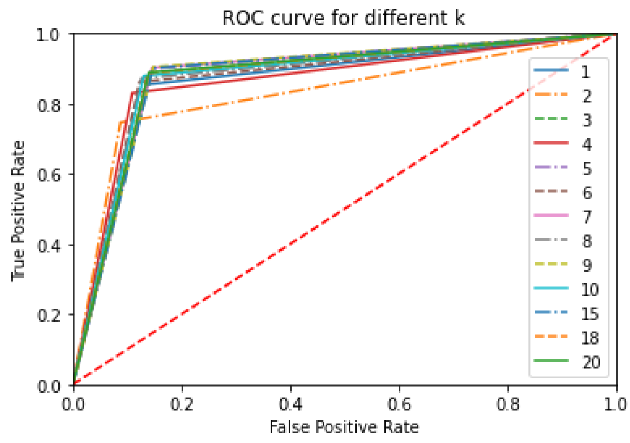
Metrics
ROC_AUC: Area Under the Receiver Operating Characteristic curve (TP vs FP curve)
where TP is True positive (samples which were positive and classified as positive,
FP is false positive: samples that were negative but were classified as positive
TN is true negative: samples that were negative and classified as negative
FN is false negative: samples that were positive but were classified as negative

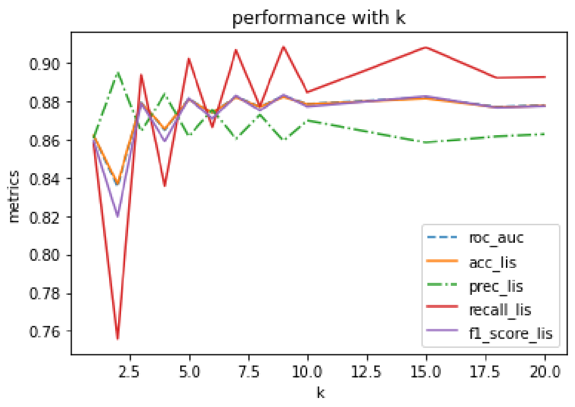


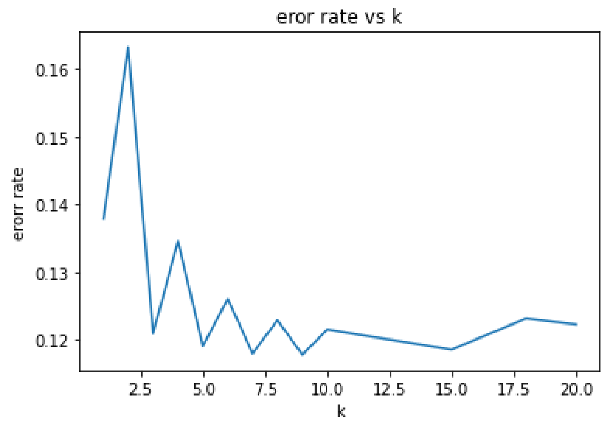
		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Accuracy = (TP + TN) / (TP+FP+TN+PN)
Precision = TP/ (TP+FP)
Recall = TP/(TP+FN)
f1-score = harmonic mean of Precition and recall = 2* precision*recall/ (precision + recall)

Results
*** different k values for knn**



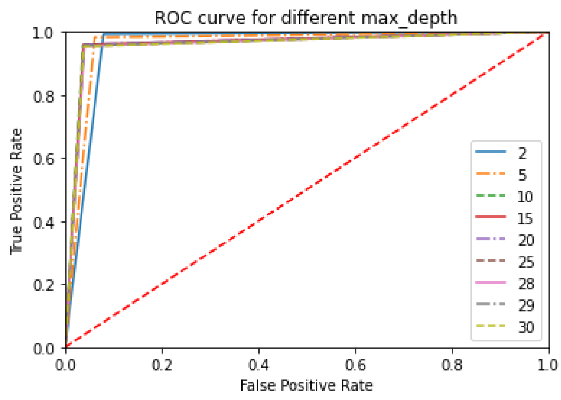


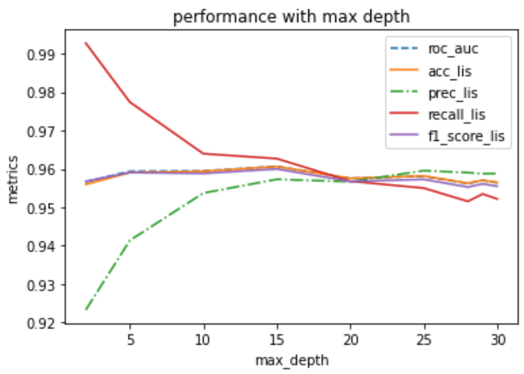


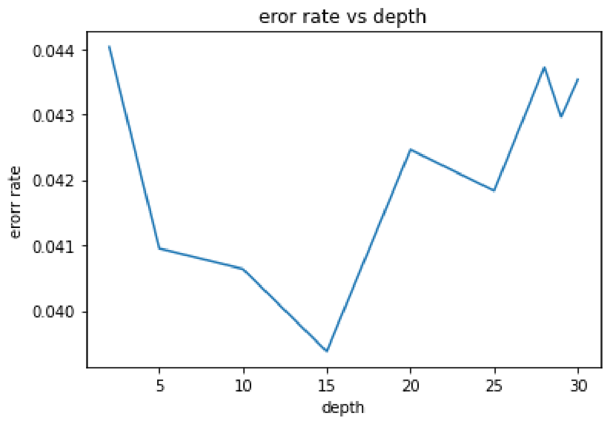
best model when selecting based on roc is the one with the highest roc_auc(area under roc) which comes at k: 9, roc_auc: 0.8826, accuracy: 88.22%

Observations: knn's error rate alternates as k increases but the roc_auc grows steadily for larger k's reaching a nearly constant value and decreasing slightly after it.
The error rate decreases(or accuracy increases) when roc_auc increases but the precision shows an opposite trend.
Precision ∝ 1/roc_auc

*** different pre- prune depth**



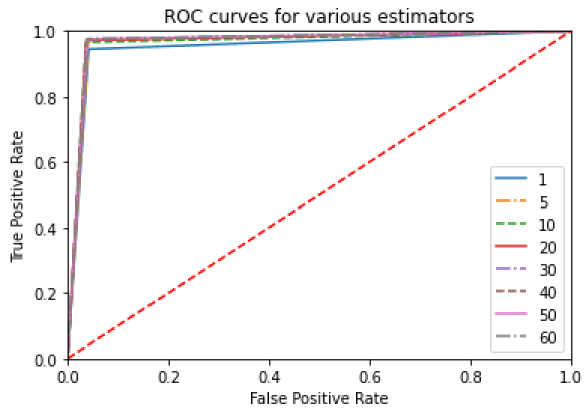


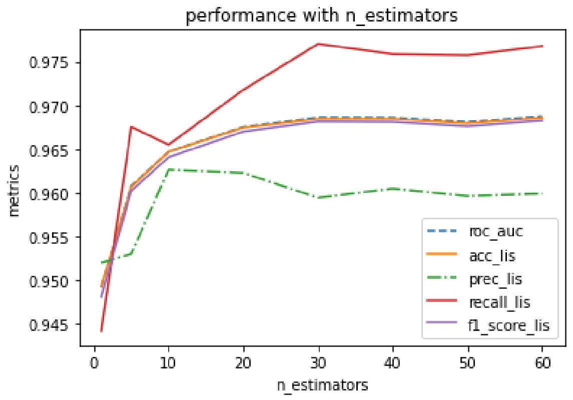


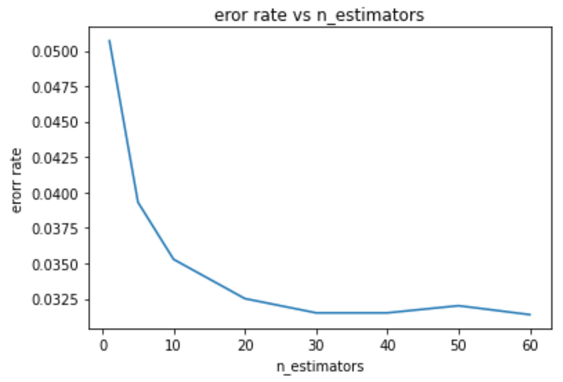
best model as best roc_auc
depth: 15, roc_auc: 0.96098, accuracy:96.10

Error rate decreases(roc_auc increases) and reaches a min value then starts increasing with increasing depth of the tree.
This is because the model starts overfitting with a larger depth.

*** different number of estimators in the ensemble**





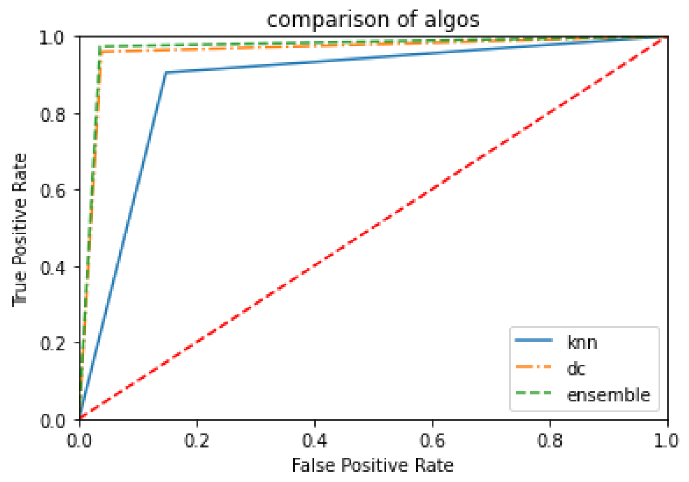


best model as best roc_auc
estimator: 30, roc_auc, 0.9686, accuracy: 96.85

Selecting the best models from above and comparing them

The error rate keeps on decreasing with increasing n_estimators, reaches a minimum,and then starts increasing. This is because the more the number of weak learners, the better their predictions (they supplement each other and reduce error) but when there are too many learners, overfitting starts happening and hence performance decreases on test set.

Comparison of algorithms



knn roc_auc 0.8787 Accuracy 0.8782 Precision 0.855 Recall 0.9054 f-score 0.8795
dc roc_auc 0.9606 Accuracy 0.9602 Precision 0.9393 **Recall 0.9823** f-score 0.9603
ensemble roc_auc 0.9698 Accuracy 0.9697 Precision 0.964 Recall 0.9746 **f-score 0.9693**

Model	**ROC_AUC**	**Accuracy (%)**
KNN	0.8787	87.82
DC	0.9605	96.05
Ensemble	**0.9688**	**96.87**

If we consider roc_auc, accuracy, precision, recall, f-score values, the ensemble method performs best for feature set F.

Selecting only 2 features

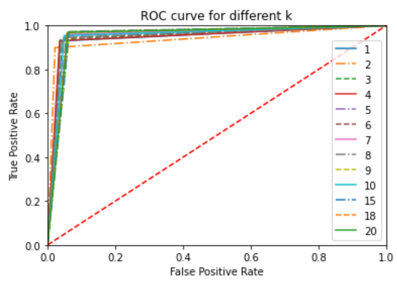
We use the decision tree's feature_importance value for the selection of features
The feature importance value is Normalized total reduction of criteria by feature (Gini importance).
feature importance =
*N_t / N * [impurity - (N_t_R / N_t * right_impurity) - (N_t_L / N_t * left_impurity]*

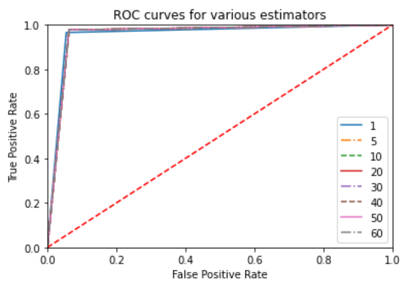
where,
impurity is the gini/entropy value,
N is the total number of samples,
N_t is the number of samples at the current node,
N_t_L is the number of samples in the left child, and
N_t_R is the number of samples in the right child.
reference:
*https://stackoverflow.com/questions/49170296/scikit-learn-feature-importance-calculation-in-decision-trees
*https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

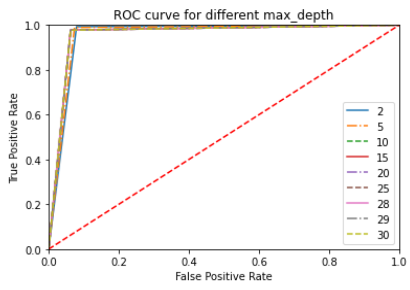
Using sklearns feature_inportance we get

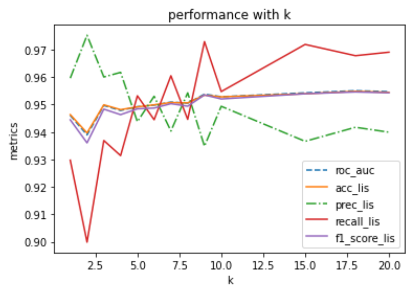
'LOCATION': 0.00039902652487388104,
'Unnamed: 11': 0.002485919723669457,
'Unnamed: 10': 0.0032411937218002843,
'Unnamed: 9': 0.004795823679251393,
'DEPTH (km)': 0.00510355542874139,
'Unnamed: 8': 0.008296241445613629,
'LAT (N)': 0.011442293561359628,
'LONG (E)': 0.011658266183065308,
'MONTH ': 0.012312865163954427,
'DATE': 0.019047279227264968,
'YEAR ': 0.027295579402277538,
'REFERENCE': 0.040415717100273606,
'Unnamed: 7': 0.8535062388378546

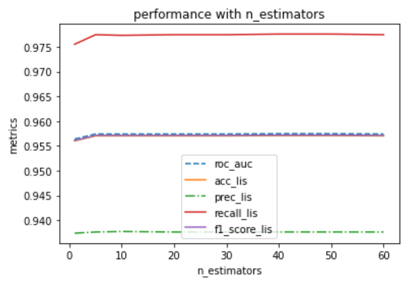
The features with the highest importance value are selected
which are **'REFERENCE' and 'Unnamed: 7'**

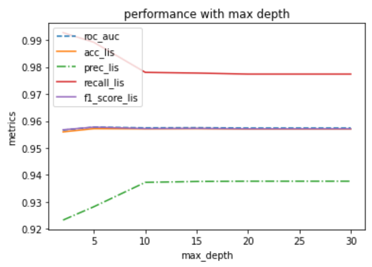


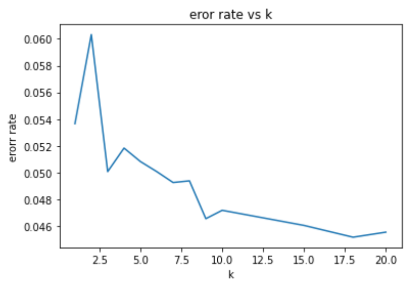


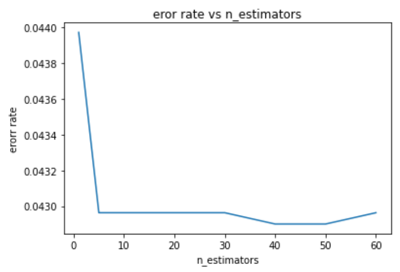


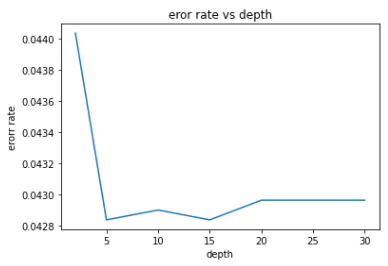






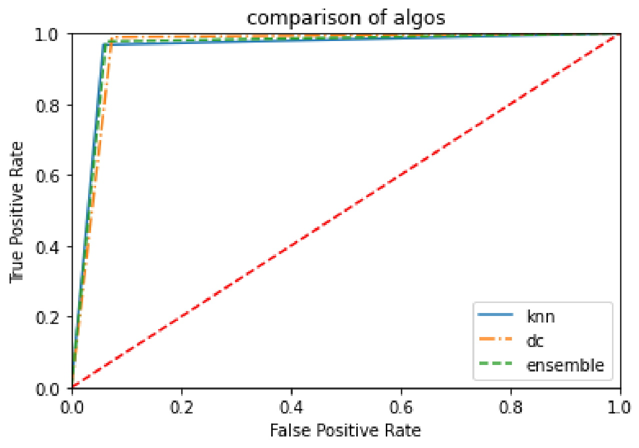






The best values for above are obtained at
KNN: k=18, roc_auc = 0.95507 (error rate at its min and roc_auc at its max)
Ensemble:estimators = 40 roc_auc 0.9575
DC: depth = 5, roc_auc = 0.9577

Comparing algorithms for best values



knn roc_auc 0.9551 Accuracy 0.9548 Precision 0.9417 Recall 0.9678 f-score 0.9546

dc roc_auc 0.9577 Accuracy 0.9572 Precision 0.9282 **Recall 0.9892 f-score 0.9577**
ensemble roc_auc 0.957 Accuracy 0.9566 **Precision 0.936** Recall 0.9785 f-score 0.9568

Decision tree performs best when only 2 features are selected

Feature processing

- *Making additional features*
  Origin time can be divided in the time of day and used

  Dividing origin time in four parts: morning, afternoon, evening, night

  24 hr format

  6-12: morning

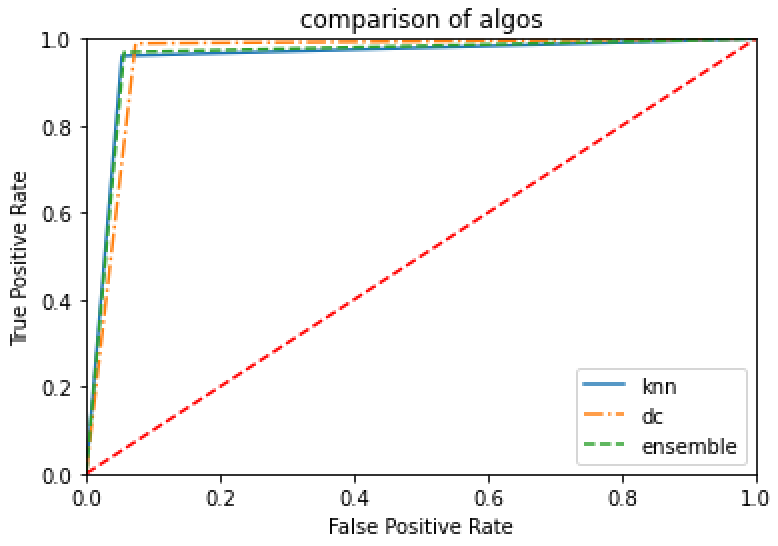
  12-17: afternoon

  17-20: evening

  20-6: night

Taking best performing algorithm for different parameters based on roc_auc and comparing the models.

k = 18, depth = 5, n_estimators = 5

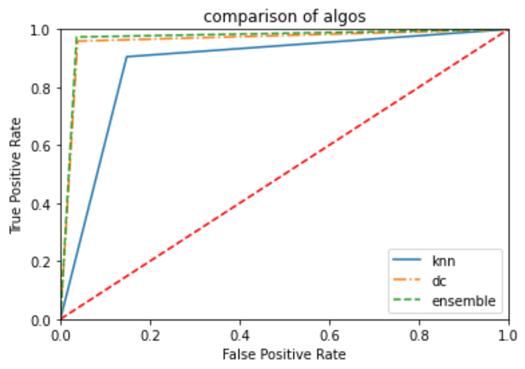


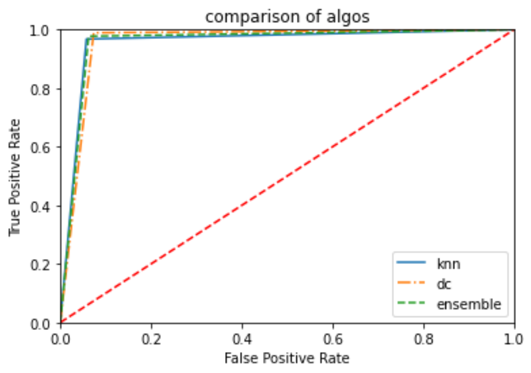
knn roc_auc 0.9536 Accuracy 0.9535 Precision 0.9456 Recall 0.9605 f-score 0.953
dc roc_auc 0.9577 Accuracy 0.9572 Precision 0.9282 Recall 0.9892 f-score 0.9577
ensemble roc_auc 0.9562 Accuracy 0.956 Precision 0.9428 Recall 0.9691 f-score 0.9557

Again decision tree performs best.

Comparing results obtained in many features vs best 2 features

many features **best 2 features**





We clearly see that performance of KNN improves a lot when features are reduced to the best 2 features whereas the performance of decision tree and ensemble decreases slightly as features are decreased. This is because the decision tree learns a set of rules to follow and when features are decreased the performance decreases as well.

The same goes with the ensemble, ensemble is a combination of weak learners which when combined make an overall stronger learner. We use the bagging method for our ensemble. In bagging weak learners are learned independently and combined in a deterministic manner. Hence many decision trees are learned randomly and are combined together to give an ensemble model leading to increased performance with a number of features.

Please refer to notebook A1.ipynb for additional details on preprocessing, setting of threshold, and experimentation along with feature selection.