**Pattern Recognition and Machine Learning**

**(Winter 2022)**

**Assignment 3: Bagging & Boosting**

**Report**

# Question - 1:

- **Data Pre-Processing.**

I first took the input of the given dataset. Then I performed the data pre-processing on the given dataset.

I checked if there are any null values present in the dataset. It appears that there are no null values present in the dataset.

```
price             0
area              0
bedrooms          0
bathrooms         0
stories           0
mainroad          0
guestroom         0
basement          0
hotwaterheating   0
airconditioning   0
parking           0
prefarea          0
furnishingstatus  0
dtype: int64
```

Then I checked for the categorical data in the dataset. And, I performed the categorical encoding on the categorical data present in the dataset.

Then I performed normalization/scaling on the applicable features in the dataset.

Finally, I split the data into training and testing sets.

**<u>Task - 1:</u> Use a simple Decision Tree regressor to predict the price of a house (without any validation) and report the accuracy.**

For the Decision Tree regressor, I used the inbuilt Decision Tree regressor from sklearn library and fit the training set in the Decision Tree regressor. And then stored the Predictions by passing testing data in the Decision Tree regressor.

For reporting the accuracy, I implemented the function to calculate Mean Squared Error MSE.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
$n$ = number of data points
$Y_i$ = observed values
$\hat{Y}_i$ = predicted values

For implementing the MSE Function I used the formula shown in the image.

I passed the actual test results and predicted values in the MSE function and got the MSE = 0.02013958379086949.

**<u>Task - 2:</u> Perform 5-fold cross-validation to determine what the best max_depth would be for a single regression tree using the entire 'Xtrain' feature set.**

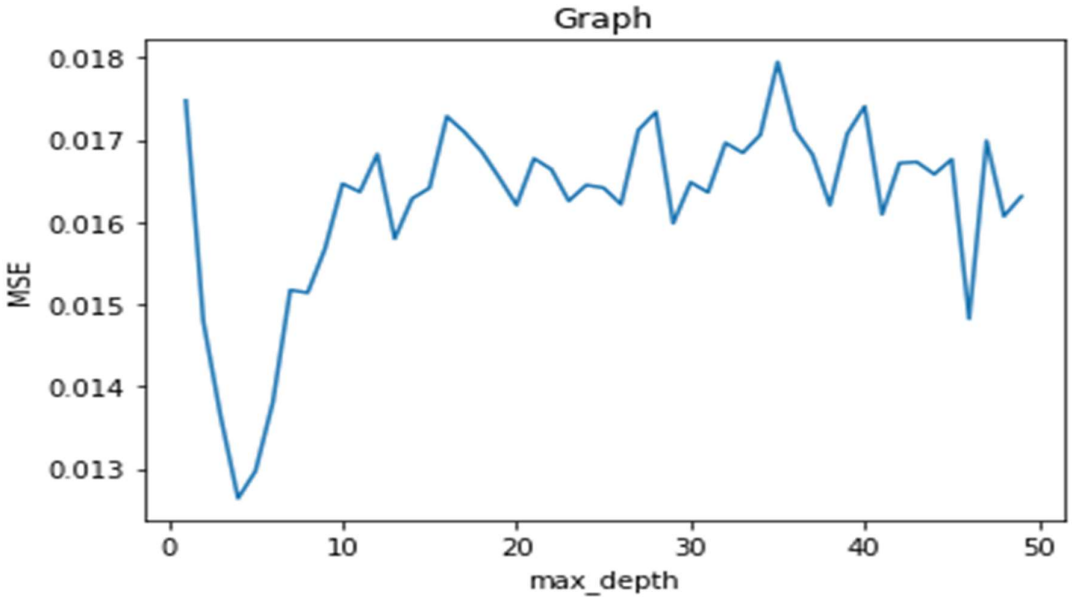
For this part I performed 5-fold cross-validation from scratch. I implemented the function that divides the entire training dataset into 5 parts. I also passed a parameter named ‘folds’ in the implemented function K-Fold so that K in the K-fold is controlled by that ‘folds’ parameter.

For determining the best max depth for the single regression tree. I used the logic that I iterated the depth in range(1,50) on 5 folds and stored the average of mse of 5 folds for every depth in each iteration of the loop. For all depths from 1 to 50, I also maintained a dictionary and stored the corresponding mse for depths.

Hence, the depth with minimum mse corresponds to the best max depth. For my case the best max depth came out to be 4.

**<u>Task - 3:</u> Visualize and summarize the results across the validation sets.**

For visualization across the validation sets. I used the dictionary created in the previous task to visualize how mse varies when max_depth is increased from 1 to 50.




So, we can visualize from the graph that mse is minimum when max depth is 4. And, when the max_depth is increased further it is clearly observed that mse also increases. Hence best max depth is 4.

**<u>Task - 4:</u> Apply bagging to create different training datasets (select n_estimators = 10).**

For bagging, also known as bootstrap aggregation, I created 10 random sample of data (different training datasets) from a training set and data is selected with replacement and hence stored all the different training sets in a list. Hence, applied the principles of Bagging.

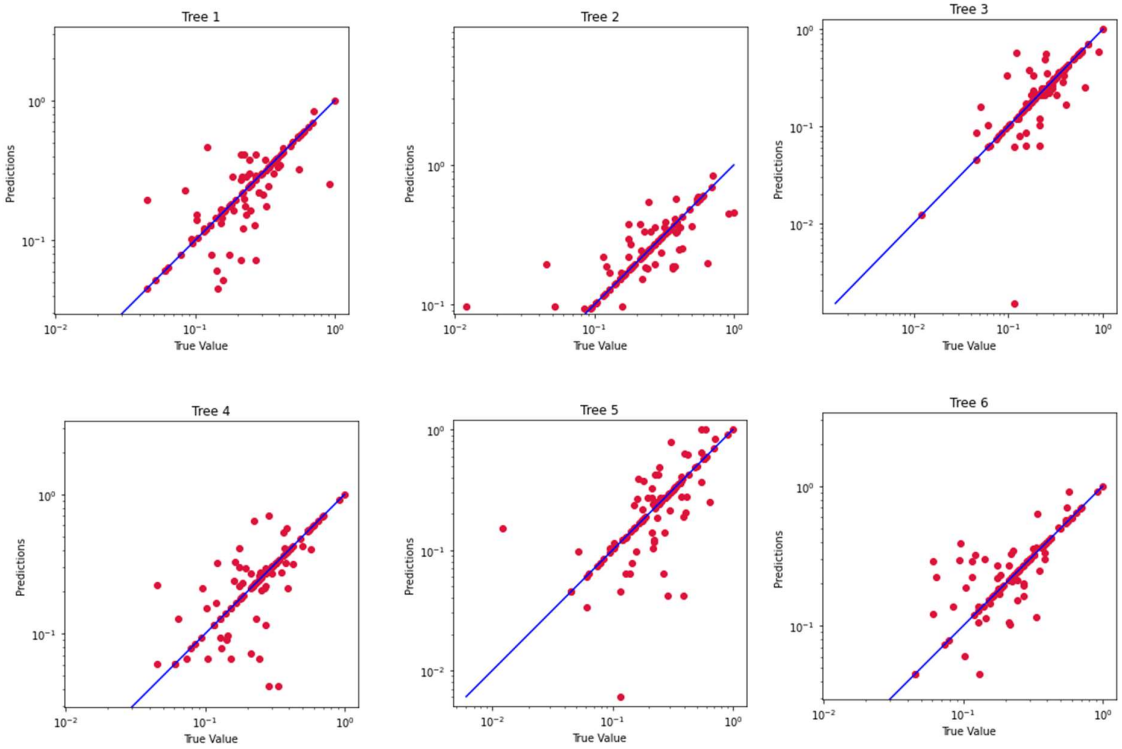
**<u>Task - 5:</u> Train on different dataset to obtain different decision trees.**

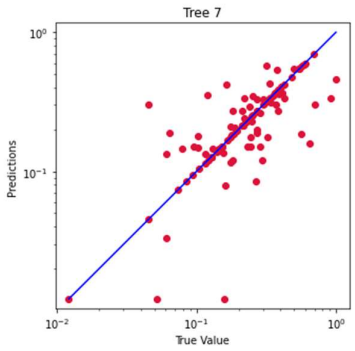
Now I trained all the different training datasets that I got after bagging. I created a list 'models' and appended all the different trained datasets (that we got by applying bagging principle in the previous task) on Decision Tree Regressor. By this I was able to obtain different decision trees.

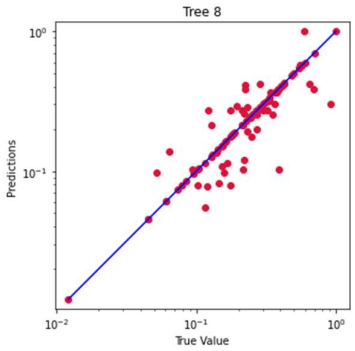
**<u>Task - 6:</u> Summarize how each of the separate trees performed (both numerically and visually) using R-squared score as the metric. How do they perform on average?**

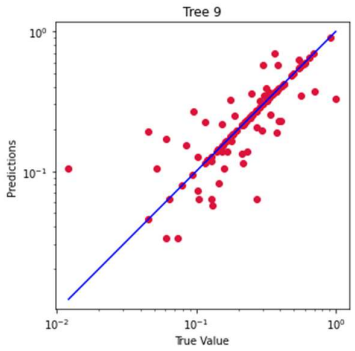
For each tree/model from the obtained different decision trees in the previous task. I calculated the R-squared score (numerical performance) for trees and also visualize how the tree predicted the values.

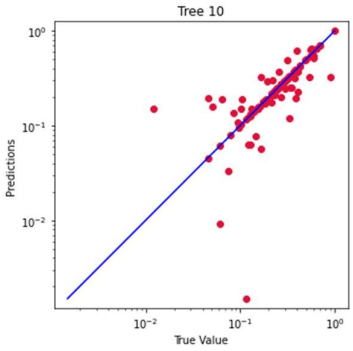
I am attaching the graphs that I got for each tree/model.



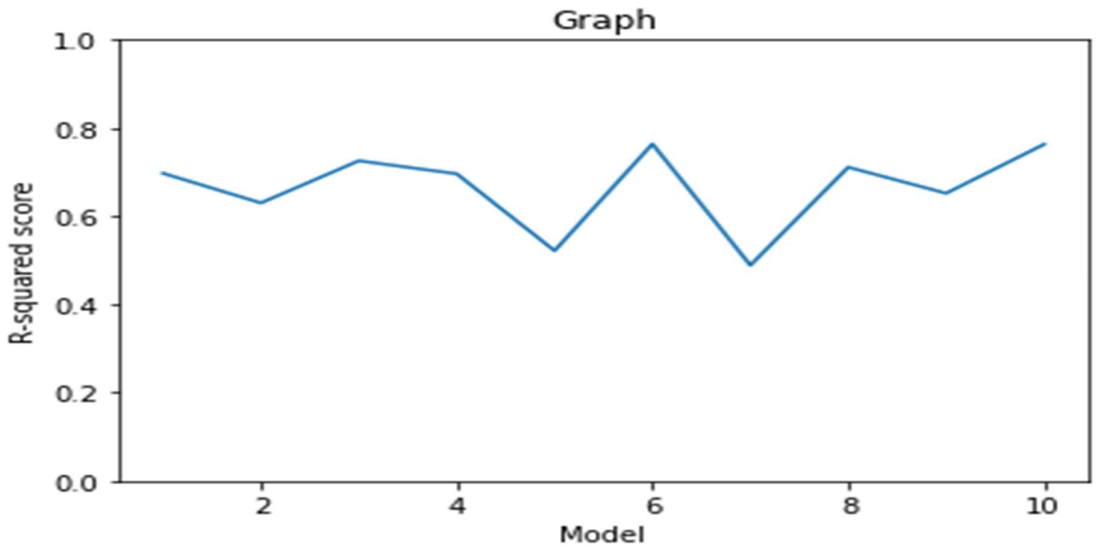



```
R-squared score for model/tree 1 is 0.6978090425940774
R-squared score for model/tree 2 is 0.6306471463956437
R-squared score for model/tree 3 is 0.7260949089958082
R-squared score for model/tree 4 is 0.6966691445332831
R-squared score for model/tree 5 is 0.5224138097103794
R-squared score for model/tree 6 is 0.7641376704911418
R-squared score for model/tree 7 is 0.4894167200855648
R-squared score for model/tree 8 is 0.7115965819598842
R-squared score for model/tree 9 is 0.6525014658343523
R-squared score for model/tree 10 is 0.7634030074876135
On average the r2Score is 0.6654689498087748
```

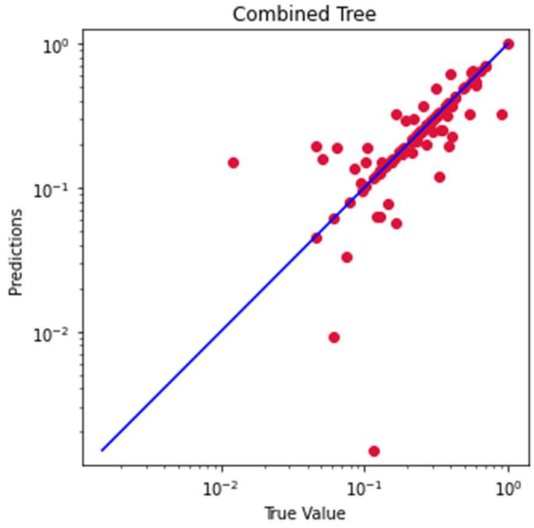



The individual graphs for each decision tree trained represents the accuracy of their predictions. The block that I have attached after the individual graphs represents R-Squared score for each decision tree also describes the how the trees performed on an average. The last graph shows R-Squared score for each model/tree that I trained.

**<u>Task - 7:</u> Combine the trees into one prediction and evaluate it using R-squared score.**

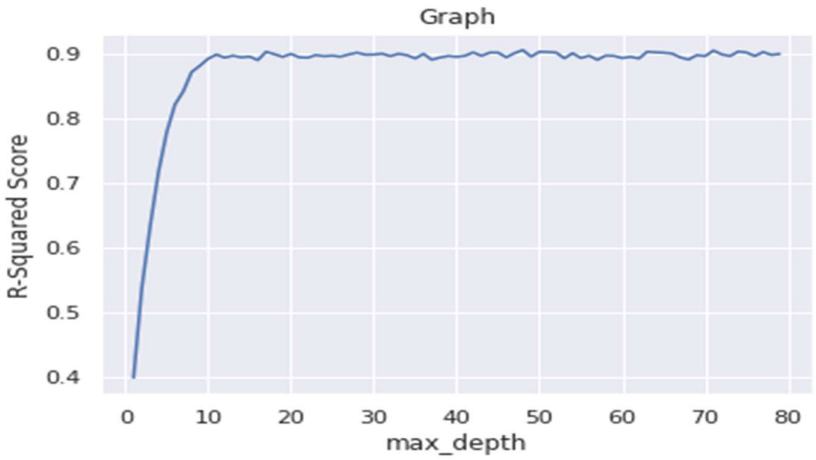
Now, in this task it was asked to combine the trees into one prediction and then evaluate it using R-squared score. So, I implemented the logic as I made a single model that returns the average of decisions/predictions of the previously trained different trees and finally evaluated the model using the R-squared score.

Hence for this model (combined the trees) R-Squared score came out to be 0.8964542718359629 and visually it can be shown as:




**<u>Task - 8:</u> How will the results above change if 'max_depth' is increased? What if it is decreased?**

I plotted the graph for R-squared error v/s max_depth to see how the results change. The graph plotted in the colab file is attached below. The graph shows if max_depth is increased, the value of R-squared score increases and after a value of max_depth, R-squared score becomes nearly stable/constant.



**<u>Task - 9:</u> Train random forest regressor, report mean squared error and mean absolute error. (from sklearn.ensemble import RandomForestRegressor).**

For this task I imported Random Forest Regressor from the sklearn library and first instantiated the Random Forest Regressor. Then, I took the training sets and trained the Random Forest Regressor using fit() method. Then I predicted the values of testing sets from the model trained using the predict() method.

Hence, for the predictions, I computed the mean squared error and mean absolute error and it came out to be as mse = 0.012686472420285427, mae = 0.07401366363503978.

**<u>Task - 10:</u> Train Adaboost regressor, report mean squared error and mean absolute error. (from sklearn.ensemble.AdaBoostRegressor).**

For this task I imported the AdaBoost Regressor from the sklearn library and first instantiated the AdaBoost Regressor. Then, I took the training sets and trained the AdaBoost Regressor using fit() method. Then I predicted the values of testing sets from the model trained using the predict() method.

Hence, for the predictions that we got, I computed the mean squared error and mean absolute error and it came out to be as mse = 0.01308860884364702, mae = 0.08276128208166425.

# Question - 2:

- **Data Pre-Processing.**

I first took the input of the given dataset. Then I performed the data pre-processing on the given dataset.

```
mean_radius          0
mean_texture         0
mean_perimeter       0
mean_area            0
mean_smoothness      0
diagnosis            0
dtype: int64
```

I checked if there are any null values present in the dataset. It appears that there are no null values present in the dataset.

Since no categorical data was present in the dataset, hence, no categorical encoding was required. Finally, I split the data into training and testing sets.

**Task - 1: Use a simple Decision Tree classifier to predict the outcome (without any validation) and report the accuracy.**

In this task I implemented the decision tree classifier from scratch to predict the outcome and report the accuracy. I implemented all the functions from scratch that are required for a Decision Tree Classifier like entropy, information gain, best attribute, make split, build tree etc. Hence, finally I trained my training set on Decision Tree Classifier and got the accuracy as 87.71929824561403 %.

**Task - 2: Perform 5-fold cross-validation to determine what the best max_depth would be for a single regression tree using the entire 'Xtrain' feature set.**

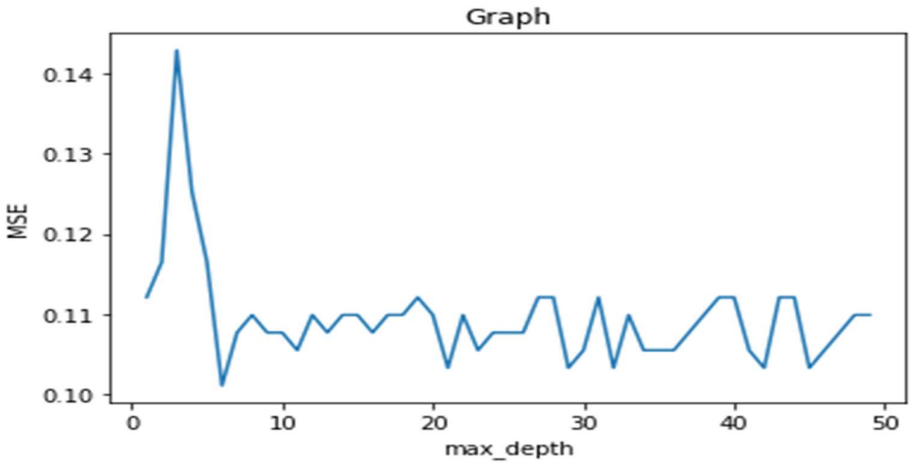
For this part I performed 5-fold cross-validation from scratch. I implemented the function that divides the entire dataset into 5 parts. I also passed a parameter named ‘folds’ in the function K-Fold so that K in the K-fold is controlled by that ‘folds’ parameter.

For determining the best max depth for the single classifier tree. I used the logic that I iterated the depth in range(1,50) on 5 folds and stored the average of mse of 5 folds for every depth in each iteration of loop. For all depths from 1 to 50, I also maintained a dictionary and stored the corresponding mse for depths.

Hence, the depth with minimum mse corresponds to the best max depth. For my case the best max depth came out to be 6.

**<u>Task - 3:</u> Visualize and summarize the results across the validation sets.**

For visualization across the validation sets. I used the dictionary created in previous task to visualize how mse varies when max_depth is increased from 1 to 50.




So, we can visualize from the graph that mse is minimum when max depth is 6. Hence best max depth is 6.

**<u>Task - 4:</u> Implement XGBoost in which subsample=0.7 and max_depth=4.**

For implementing the XGBoost Classifier, I imported the XGBClassifier from the xgboost library and first instantiated the XGBClassifier. I set the hyperparameters subsample as 0.7 and max_depth as 0.4. Then, I trained the classifier with the training set.

**<u>Task - 5:</u> Print the accuracy on the training set and test set.**

For getting the accuracy on the training set and test set. I calculated the score for training set and testing for the model.

The Training score and Testing score came out to be:

```
Training set accuracy: 0.9950
Testing set accuracy: 0.9240
```

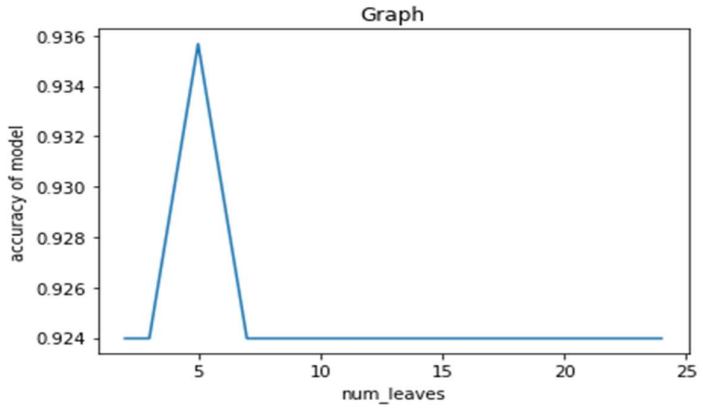
On observation, the training and testing set accuracy are quite comparable. So, we cannot say that there is overfitting.

**<u>Task - 6:</u> Implement LightGBM with max_depth value as 3 and choose different value for num_leaves.**

For implementing the LightGBM Classifier, I imported the LGBMClassifier from the lightgbm library and first instantiated the LightGBM Classifier. I set the hyperparameters max_depth as 3 and iterated over different values of num_leaves simultaneously training the classifier with the training set.

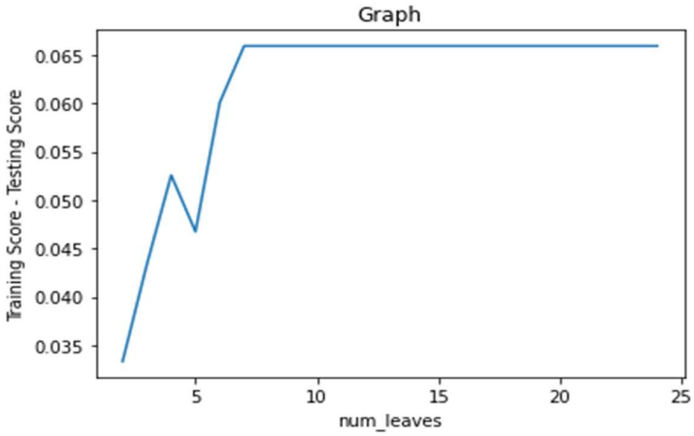
**<u>Task - 7:</u> Analyse the relation between max_depth and num_leaves, and check for which value the model starts overfitting?**

For analysis I plot two graphs. One graph is accuracy of model v/s num_leaves i.e.,




The graph shows that the accuracy of model is best when num_leaves = 5.

Also, I plotted a graph (Training Score - Testing Score) v/s num_leaves i.e.,



This graph also clearly shows that num_leaves greater than 5 (close to 8) leads to overfitting.

Hence, on analysing the graphs and accuracies/mse for different values of num_leaves we can say that the value of num_leaves should be less than or equal to 2^(max_depth). Value of num_leaves more than this will result in overfitting.

**<u>Task - 8:</u> Report which parameters can be used for better accuracy and also which parameter can be used for avoiding overfitting.**

Parameters that can be used for better accuracy are:

- Use large num_leaves.
- Using bigger training data.
- Use large max_bin
- Use small learning_rate with large num_iterations.

Parameters that can be used for avoiding overfitting are:

- Use small num_leaves, max_bin.
- Use ensemble approach i.e., use bagging principle by set bagging_fraction and bagging_freq.
- Use feature sub-sampling by set feature_fraction.
- Set max_depth to avoid growing deep decision tree.