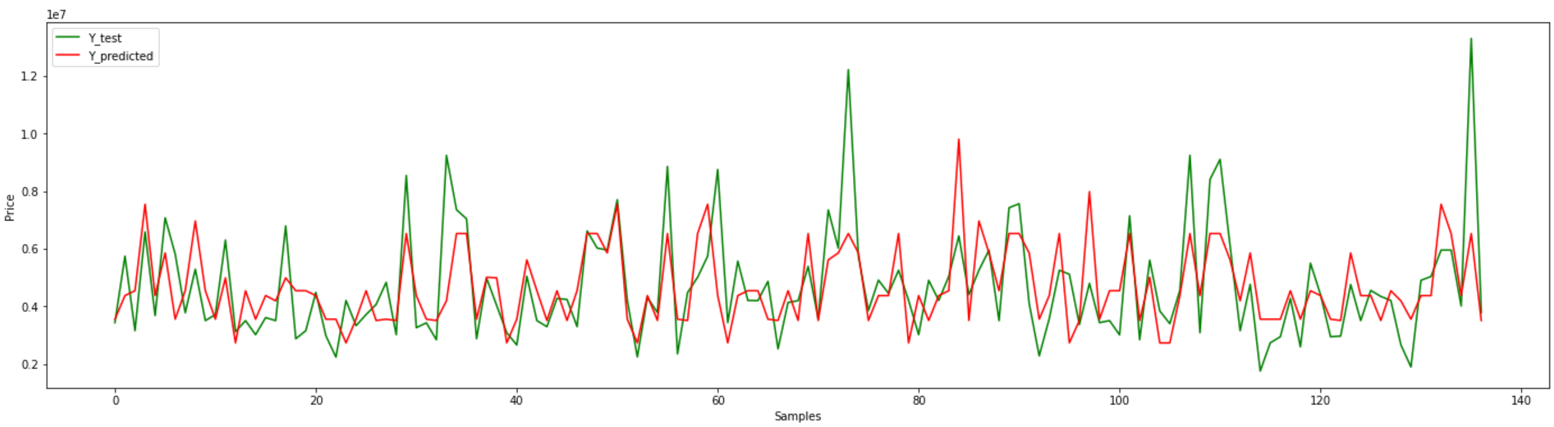
Maniya Yash Rajeshbhai
B20CS033

ML Lab 3 Report

Question 1 :

1. Predictions using simple Decision Tree Regressor :

- Firstly, completed preprocessing, looked upon the dataset, there were no useless features, so did not drop any features, scaled the area values.
- And then built the Decision Regressor Tree code from scratch.



- Above is the graph for Y_original and Y_predicted comparison on testing data. We see for extreme high test price values, the model is failing to predict accurate prices.

```
Metrics for the DTR model on testing data :

MSE : 2126719553927.666016
RMS Error : 1458327.66
R2 score : 0.4227
```

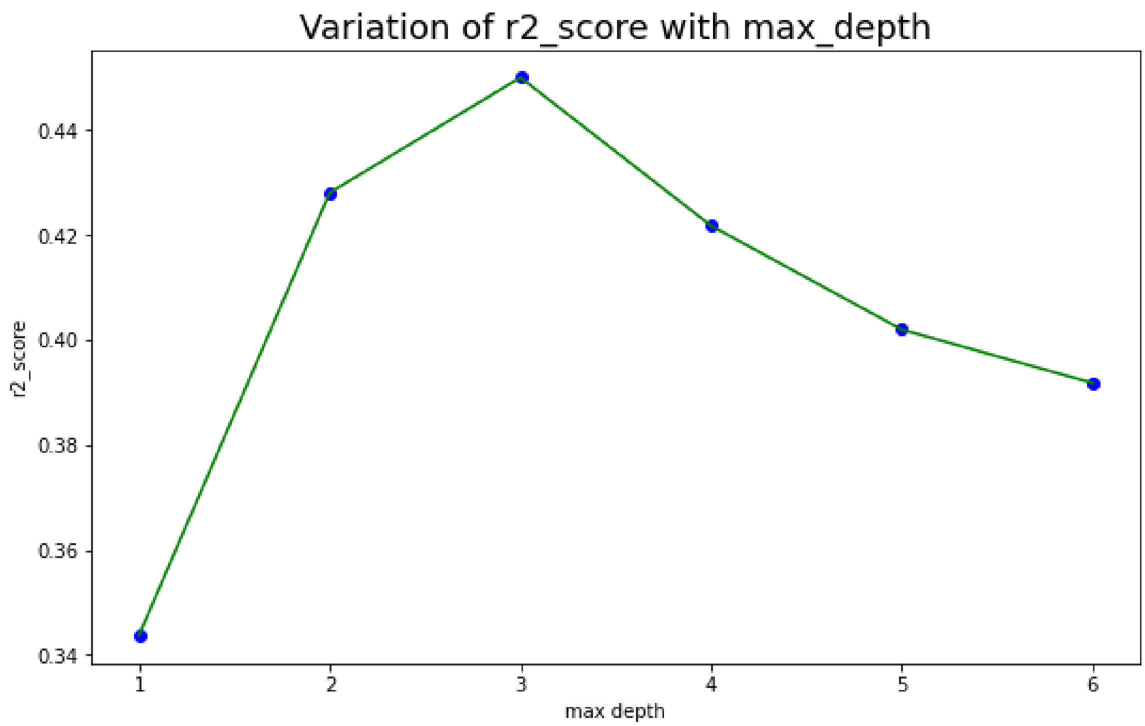
2. 5-Fold Cross Validation :

- Applied 5-fold cross validation on the whole training data and tested the variations with max_depth.
- Found optimal max_depth = 3, with maximum r2_score = 0.4499 .
- Tested the found optimal model on testing dataset and following were the results.

```
After 5-fold cross-validation and testing various max_depths,
optimal max_depth = 3
r2_score = 0.4499

Metrics of price predictions on Testing data :
r2_score = 0.4438
mean_absolute_error = 1012812.81
```

3. Visualisation of Cross Validation Results :



Hence we conclude that optimal max_depth = 3

4. Applying Bagging :

- For this task, we had to create a function which can in turn create different datasets by the BootStrap Aggregation method.

5. Training Decision Trees (Forest) :

- Now since we have created function that can create datasets, we can train Decision Trees on the created datasets.
- We have made a bagging() function to create 10 different models to train on the 10 datasets made by BootStrap Aggregation.

6. Performance of the Random Forest :

Average r2_score on BootStrapped data was 0.5542 . (varies when run again)

7. Combining the predictions of the Decision Trees :

- We have combined the predictions of the individual trees based on weights.
- The weights are scaled r2_scores of the respective trees.
- The predictions then made, and following are the results.

```
R-squared score on Testing Data, after combining the Random Forest Tree models :
0.4761
```

We see that r2_score has improved,

For single DTR,	r2_score = 0.4227	
For Random Forest,	r2_score = 0.4761	(varies when run again)

8. Varying max_depth :

Analysed the results when max_depth is varied on the forest model.
Here are the results

```
Average R-squared score for Combined Random Forest on Testing data with different max_depths :

max depth = 1 ; r2_score = 0.5833
max depth = 2 ; r2_score = 0.5351
max depth = 3 ; r2_score = 0.5602
max depth = 4 ; r2_score = 0.5192
max depth = 5 ; r2_score = 0.5629
```

I have run this many times, optimal max_depth varies on most of the times.

9. Training RandomForestRegressor from sklearn :

Following are the results for sklearn Random Forest.

```
For sklearn RandomForestRegressor, metrics on Testing data are

Mean Squared Error = 1964367302421.53
Mean Absolute Error = 988983.94
R-squared score = 0.4668
```

10. AdaBoost Regressor from sklearn :

Following are the results for sklearn AdaBoost.

```
For sklearn AdaBoostRegressor, metrics on Testing data are

Mean Squared Error = 1709461030984.37
Mean Absolute Error = 970131.22
R-squared score = 0.5360
```

Question 2 :

1. Simple Decision Tree from Scratch :

Implemented Decision Tree Classifier from scratch, and evaluated it on Testing data. Following are the results.

```
Overall Accuracy for Trained Model : 92.40 %
```

2. 5-Fold Cross Validation on Simple DTC :

- Applied 5-fold cross validation on the simple DT classifier.
- Selected the best classifier model and evaluated it on the Testing dataset.
- Following were the results :

```
Progress Bar  : 100%|██████████| 5/5 [00:41<00:00,  8.31s/it]
After 5-fold cross-validation and testing various max_depths,
optimal max_depth = 3
Accuracy = 96.25 %

Metrics of Classification on Testing data :
Accuracy = 92.98
```

We see that Accuracy after cross validation and selecting the best model is 92.98 %.
This is greater than previous model accuracy i.e. 92.40 %.

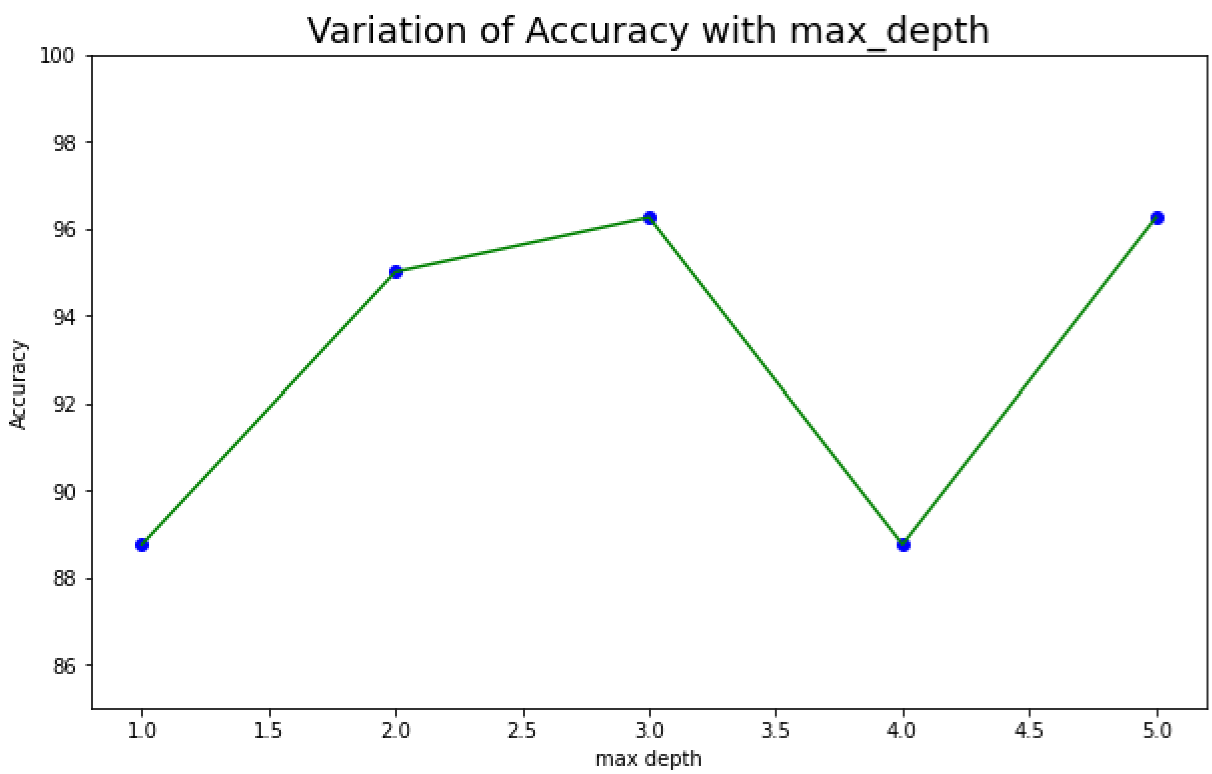
However after cross validation, we did not reach desired results, hence we go for boosting the Decision Trees with XGBoost and LightGBM.

3. Results of 5-fold Cross Validation :

Following are the results for cross validation on DTC :

```
Average Accuracies for Cross-Validation sets are :
max_depth = 1 ; accuracy = 88.75
max_depth = 2 ; accuracy = 95.00
max_depth = 3 ; accuracy = 96.25
max_depth = 4 ; accuracy = 88.75
max_depth = 5 ; accuracy = 96.25
```

We find that max_depth = 3 gives maximum accuracy.



4. XGBoost Implementation :

Implemented XGBoost with subsample = 0.7, and max_depth = 4.

5. XGBoost Results :

We get the following results on the Testing Dataset :

```
Accuracy on Training Dataset : 98.74 %
Accuracy on Testing Dataset : 95.32 %
```

6. LightGBM Implementation :

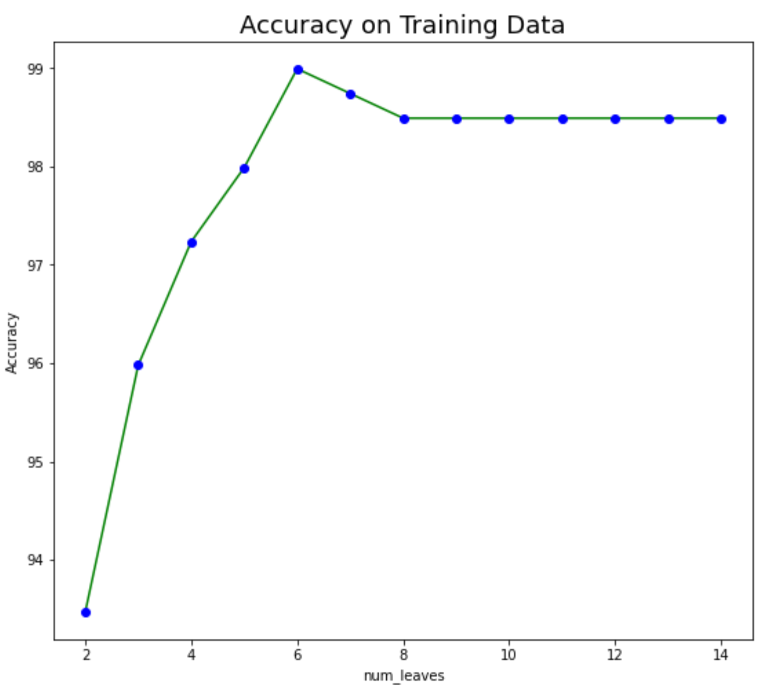
Implemented LightGBM with max_depth = 3 and different num_leaves.
Following are results we get :

```
We get Maximum Accuracy on testing dataset when num_leaves = 3

Accuracies with num_leaves = 3 :
Training Dataset Accuracy 95.98 %
Testing Dataset Accuracy 95.32 %
```

7. Analysing the relation between max_depth and num_leaves :

We make following plot for analysing the relation :



- In the above charts, we can see that for **num_leaves <= 3**, training data classification accuracy, and testing data classification is increasing.
- But for **4 <= num_leaves < 8**, testing accuracy keeps decreasing while training accuracy is still increasing.(initial overfitting)
- For **8 <= num_leaves**, testing accuracy is minimum even though training accuracy is pretty good. (complete overfitting)

Hence we conclude that for **num_leaves >= 2^(max_depth)** , the model shows complete overfitting.
So we should usually choose **num_leaves < 2^(max_depth)**.

8. Parameter Tuning :

We should control the leaf-wise tree growth for getting better accuracy on the testing dataset.

The most important parameters for tuning a LightGBM are :

1. max_depth :
   We can control the max depth of the tree explicitly.
   If we use a greater max_depth, we can overfit the tree, whereas if we keep it too low, we can underfit and have high bias. So optimum max_depth is chosen which is neither too low nor too high.

2. num_leaves :
   We should choose a value which is sufficiently less than 2^(max_depth) for num_leaves. For greater values, it will lead to overfitting as the tree grows complex.

3. min_data_in_leaf :
   Its optimal value depends on the number of training samples and num_leaves. Setting it to a large value can avoid growing too deep a tree, but may cause under-fitting.

End of the Report !