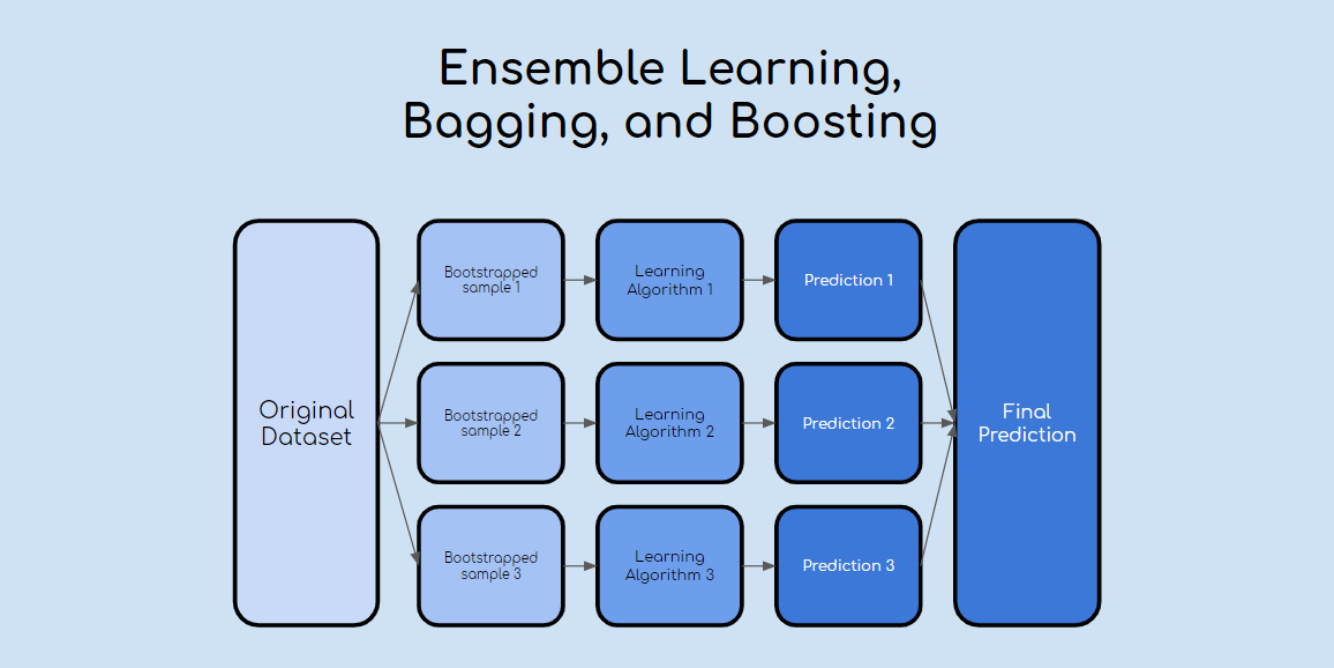
Pattern Recognition and Machine Learning

LAB3: Bagging and Boosting



Question 1.

Ensemble learning Ensemble learning, a form of meta learning, is a machine learning paradigm where multiple learners are trained to solve the same problem. The data set was preprocessed and the necessary columns were encoded. The implementation of which can be found in the notebook.

1. A simple decision tree regressor was implemented from scratch to predict the price of the houses without any validation and the r2 score that we got was

```
r2_score(Y_test, Y_pred)

0.4313391053733684
```

.

2. A k-fold function was implemented with a default value of k equal to 5. A code of which can be found below:

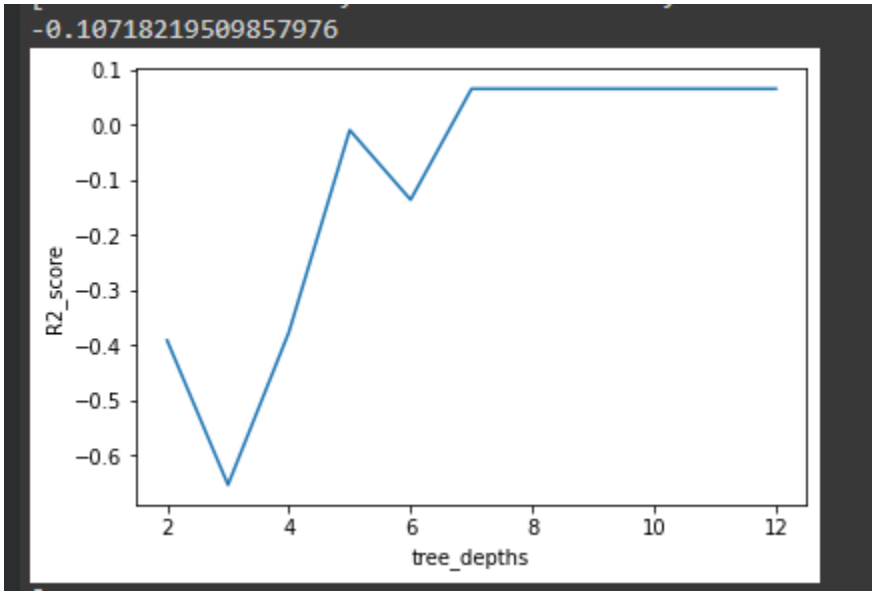
```
def KFold(X_train, Y_train, k = 5):
  for i in range(k):
    val = X_train.shape[0]//k
    x_train_split, y_train_split = X_train[i*val : (i+1)*val-2], Y_train[i*val : (i+1)*val-2]
    x_test_split, y_test_split = X_train[(i+1)*val-2 : (i+1)*val], Y_train[(i+1)*val-2 : (i+1)*val]
    tree_depths = [2,3,4,5,6,7,8,9,10,11,12]
    R2_score = []
    for j in range(len(tree_depths)):

       model = DecisionTreeRegressor(min_samples_split=3, max_depth=tree_depths[j])
       model.fit(x_train_split, y_train_split)
       y_pred_split = model.predict(x_test_split)
       R2_score.append(r2_score(y_test_split, y_pred_split))
    print(R2_score)
    print(sum(R2_score)/len(R2_score))
    plt.plot(tree_depths, R2_score)
    plt.xlabel('tree_depths')
    plt.ylabel('R2_score')
    plt.show()
```

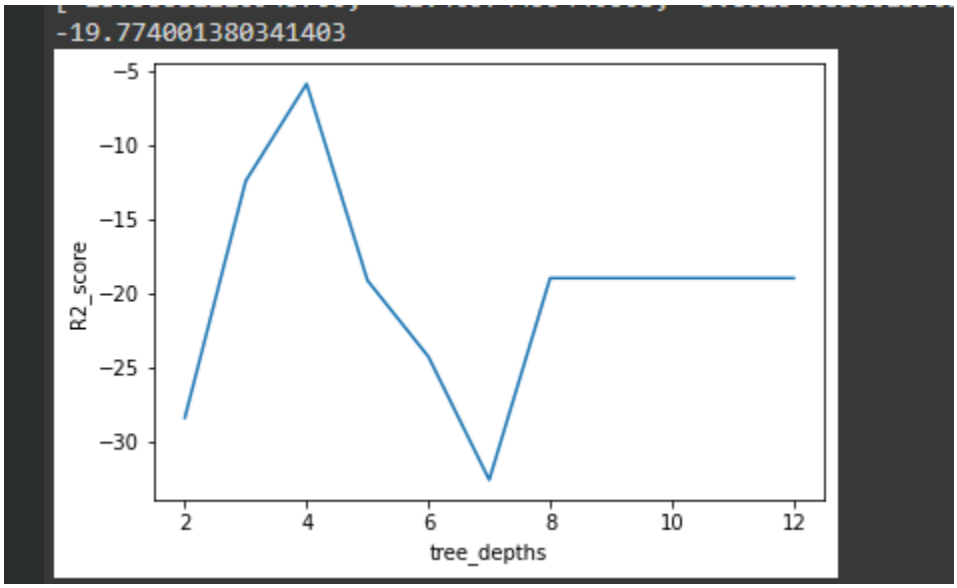
3. K fold cross validation was applied on the X train data set. The R2 scores that we reported are given below :

FOLD1

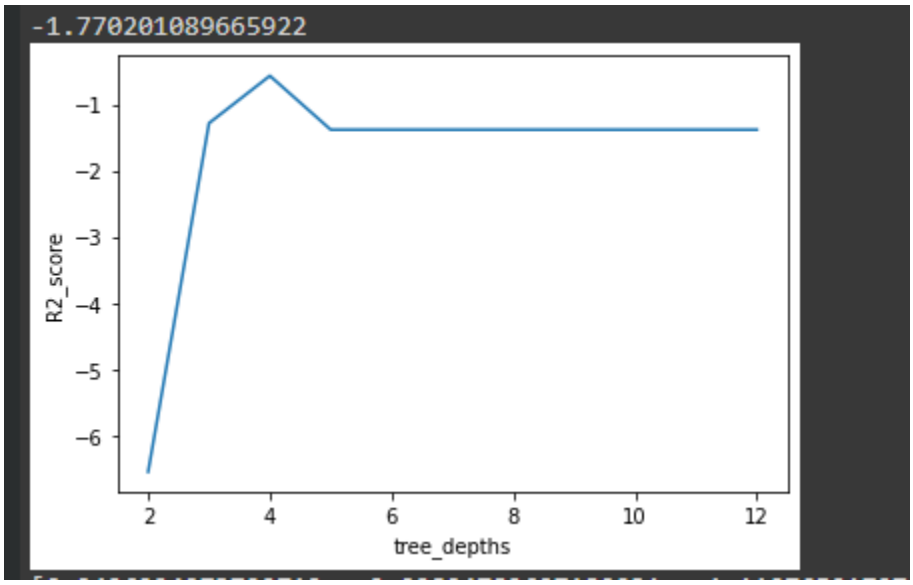
```
-0.10718219509857976
```



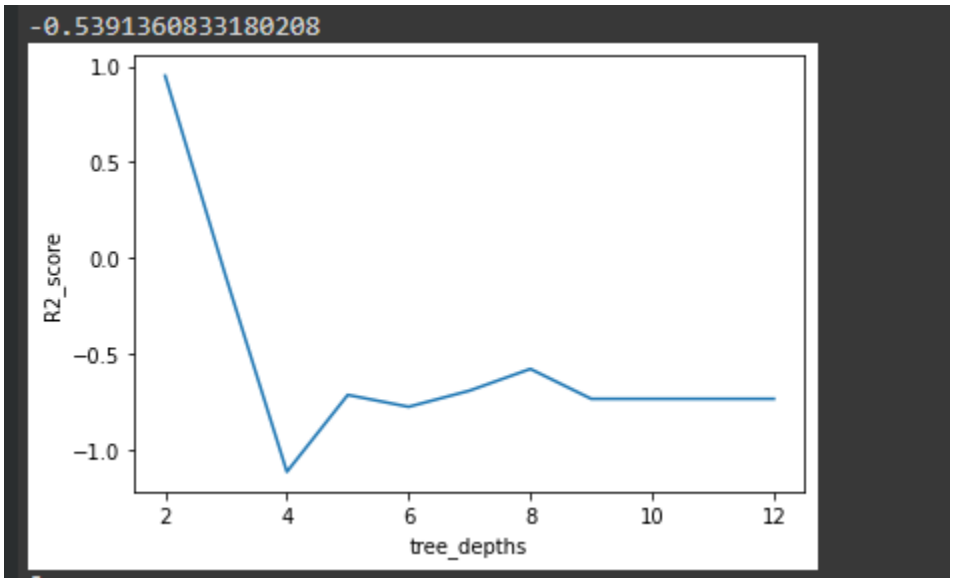
FOLD 2


-19.774001380341403
R2_score
tree_depths

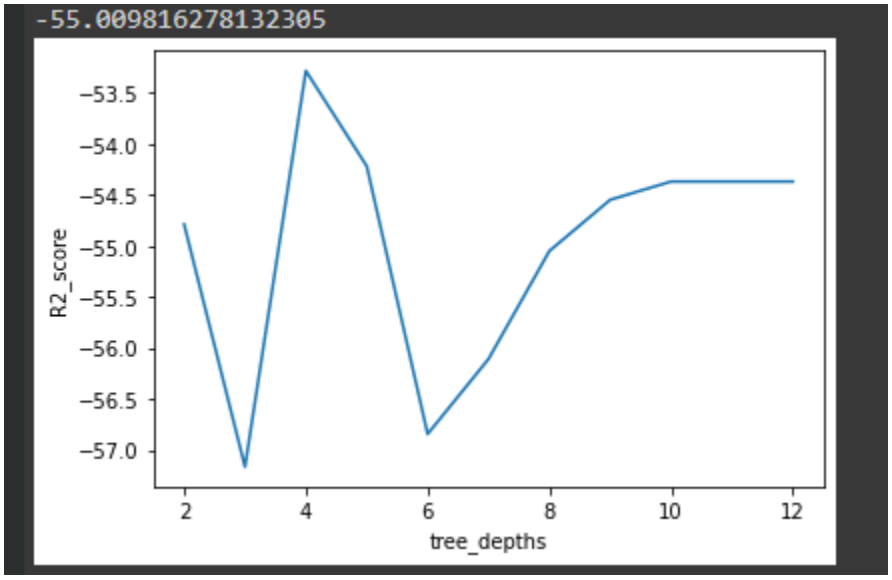
FOLD 3


-1.770201089665922
R2_score
tree_depths

FOLD 4



FOLD 5



As we can see depth 4, 5 gives the best R2 score.

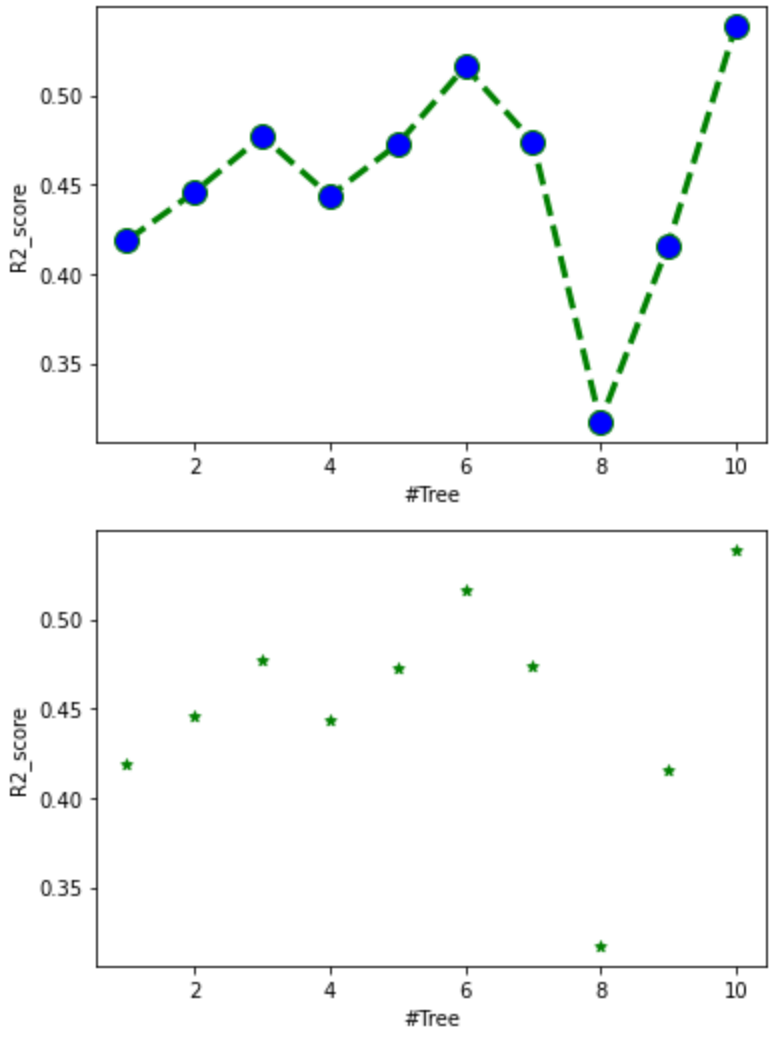
```
For depth2 The R2 Score is 0.34515230435879896
For depth3 The R2 Score is 0.4313391053733684
For depth4 The R2 Score is 0.45345505725968316
For depth5 The R2 Score is 0.45369457187961737
For depth6 The R2 Score is 0.05798955464838118
For depth7 The R2 Score is -0.03797708574601266
For depth8 The R2 Score is -0.10582545675596755
For depth9 The R2 Score is -0.11614668788866478
For depth10 The R2 Score is -0.12989377073004094
For depth11 The R2 Score is -0.12658366302998658
For depth12 The R2 Score is -0.13209401859208847
For depth13 The R2 Score is -0.12834022721055494
For depth14 The R2 Score is -0.1301425243969756
For depth15 The R2 Score is -0.1301425243969756
```

4. Bagging is a technique in which multiple weak models run in parallel and the average of the result from each model is returned. This reduces the variance in the predictions. Code for bagging Split and code can be found below

```
def scratch_bagging(df, n_estimators = 10, max_depth = 3):
  l = []
  trees = [int(x) for x in range(1, 11)]
  for i in range(n_estimators):
    df_bagging_train = sample_rows(df, 0.5)
    df_bagging_test = sample_rows(df, 0.5)
    x_bagging_train = (df_bagging_train.iloc[:, 1:])
    y_bagging_train = (df_bagging_train.iloc[:, 0])
    x_bagging_test = (df_bagging_test.iloc[:, 1:])
    y_bagging_test = (df_bagging_test.iloc[:, 0])
    t = DTR(max_depth = max_depth)
    t.fit(x_bagging_train, y_bagging_train)
    t_pred = t.predict(x_bagging_test)
```

5. Different datasets were generated with 0.5 times the size of the original data. Rows were selected from the data with replacement. Different trees (=10) were used to train on the data set parallely.
6. This is how different trees performed in the process of bagging:

```
The R2 Score for estimator1 is 0.41856407009172003
The R2 Score for estimator2 is 0.44575855894336025
The R2 Score for estimator3 is 0.476995133308615
The R2 Score for estimator4 is 0.4435885855103364
The R2 Score for estimator5 is 0.47240323810090434
The R2 Score for estimator6 is 0.515843767492056
The R2 Score for estimator7 is 0.4741467562850573
The R2 Score for estimator8 is 0.31706918644901483
The R2 Score for estimator9 is 0.4158544671464731
The R2 Score for estimator10 is 0.5383948013468953
The Average R2 score is 0.45186185646744326
```



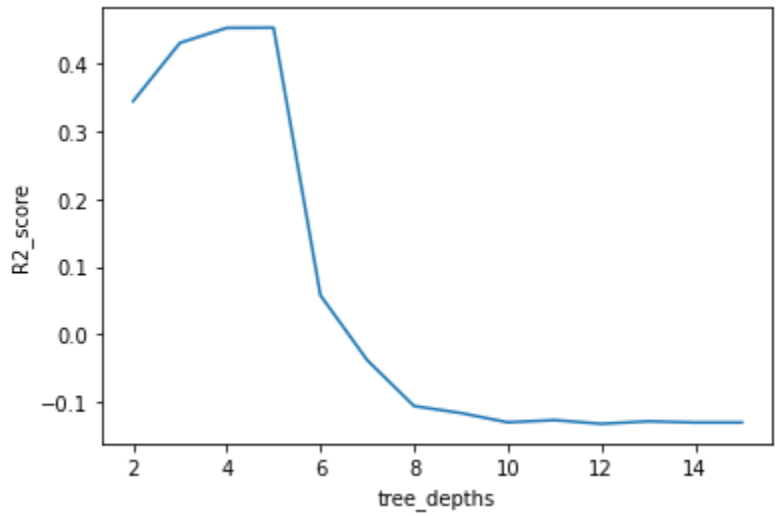
These are the different r2 scores achieved by different trees. Each tree's structure was returned by the bagging function

7. Trees were combined to get the average of the predictions. This method reduces the variance and provides us with better results. The predictions were combined and the results are below:

   TREES = 10 AVERAGE R2_Score = 0.452

8. Depth was varied from 2-15, where depth 4 gave us the best results.

```
For depth2 The R2 Score is 0.34515230435879896
For depth3 The R2 Score is 0.4313391053733684
For depth4 The R2 Score is 0.45345505725968316
For depth5 The R2 Score is 0.45369457187961737
For depth6 The R2 Score is 0.05798955464838118
For depth7 The R2 Score is -0.03797708574601266
For depth8 The R2 Score is -0.10582545675596755
For depth9 The R2 Score is -0.11614668788866478
For depth10 The R2 Score is -0.12989377073004094
For depth11 The R2 Score is -0.12658366302998658
For depth12 The R2 Score is -0.13209401859208847
For depth13 The R2 Score is -0.12834022721055494
For depth14 The R2 Score is -0.1301425243969756
For depth15 The R2 Score is -0.1301425243969756
```



As we can see before depth 4, the model underfits and does not unleash its full capability. After depth 6, the model overfits as we can see slowly, the r2 score starts decreasing. So depth 4-6 gives us the best results possible.

9. Random forest regressor was imported and then trained on the data to get the results.

```
print(mse(Y_test,y_pred))
print(mean_absolute_error(Y_test,y_pred))
```

```
1508097463032.7095
861623.7064220184
```

10. AdaBoost Regressor was imported and trained on the data to get the results

```
print(mse(Y_test,ypred))
print(mean_absolute_error(Y_test,ypred))
```

```
1752381229681.3813
990349.6212921754
```

Question 2.

Boosting is an ensemble modeling technique which attempts to build a strong classifier from the number of weak classifiers. It is done by building a model using weak models in series. Simple preprocessing techniques like removing outliers was done on the dataset. Visualization and reasoning of which can be found in the notebook itself.

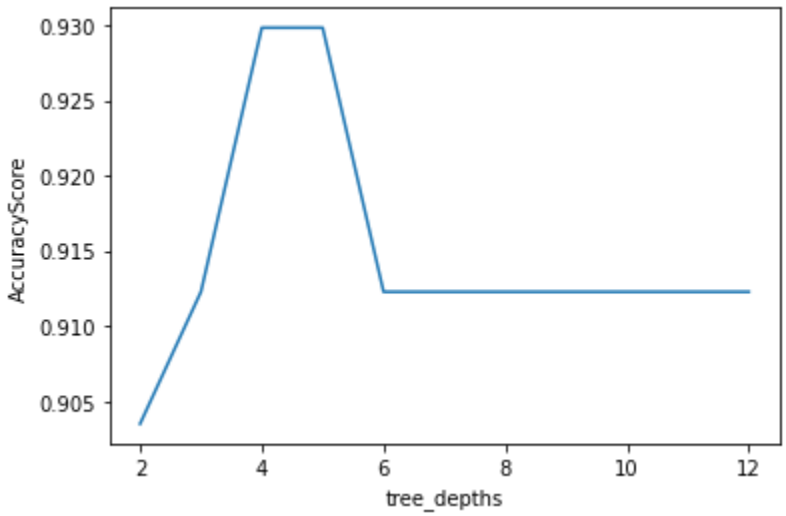
1. A simple Decision tree Classifier was implemented from scratch and the model was trained on X_train and y_train data. Results are given below

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.9122807017543859
```

2. K Fold validation with k =5 was performed to decide the best depth for the data. Depths ranging from 5 to 12 were chosen.Depth 4, 5 gave us the best results and after it the model overfits. Below 4 the model underfits and the results are not that great.Minimum accuracy was seen at depth 2.

3. The best results were found for max depth equal to 4, 5 as after that model overfits and below that model underfits. The plots above justify the statement.

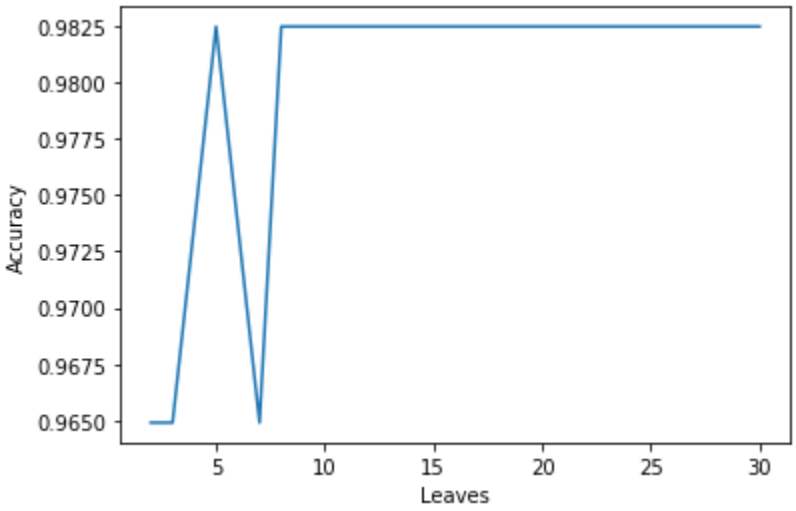


4. XGBClassifier was implemented with max_depth = 4 and subsample = 0.7. Accuracy was calculated and printed. Results are shown below.
5.

```
Accuracy over Training Data is : 98.46%
```

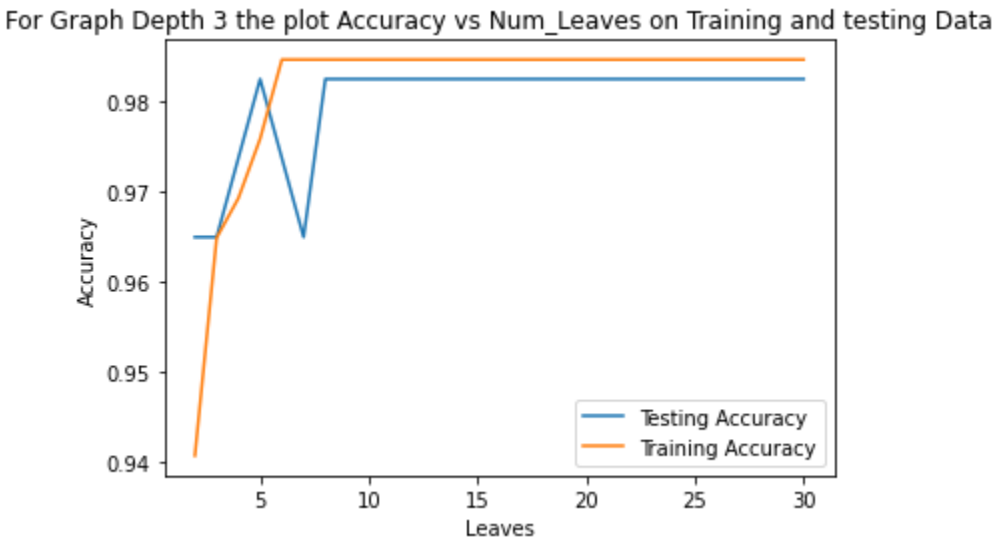
```
Accuracy over Testing Data is : 95.61%
```

6. LBGMClassifier was imported and implemented with a max_depth = 3. Different values for the num_leaves were chosen to check which one would give the best accuracy. num_leaves var from 2 to 30 and a line plot was plotted between the accuracy and the number of leaves :

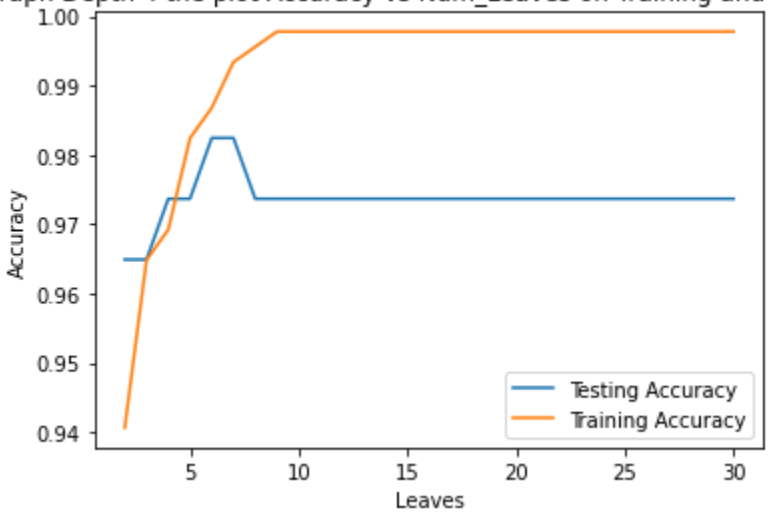



```
Highest accuracy of 0.9824561403508771 is achieved for numleaves = 5
```

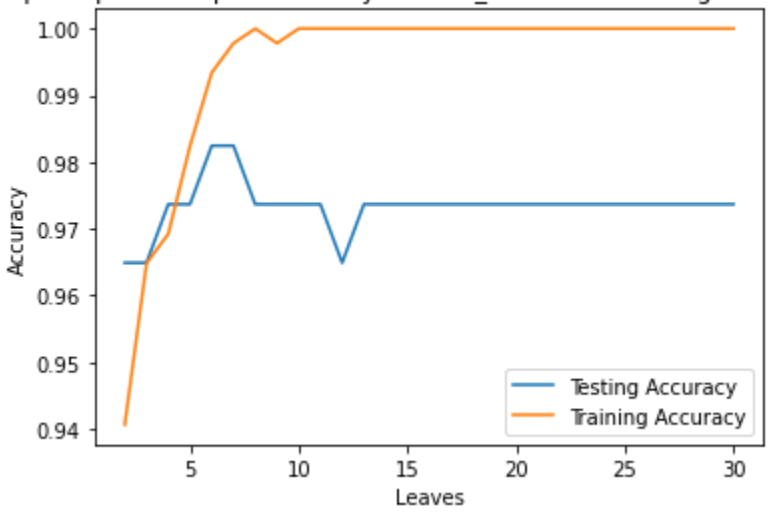
7. Training and testing accuracies were plotted for different leaves and varying depth.



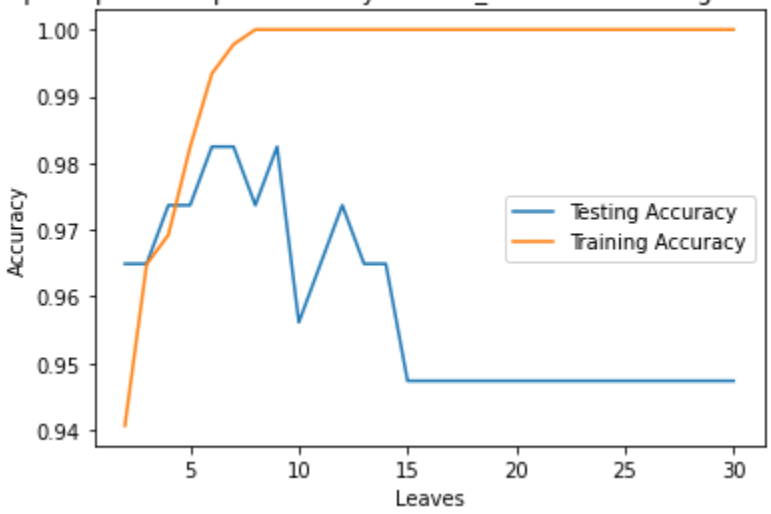
For Graph Depth 4 the plot Accuracy vs Num_Leaves on Training and testing Data



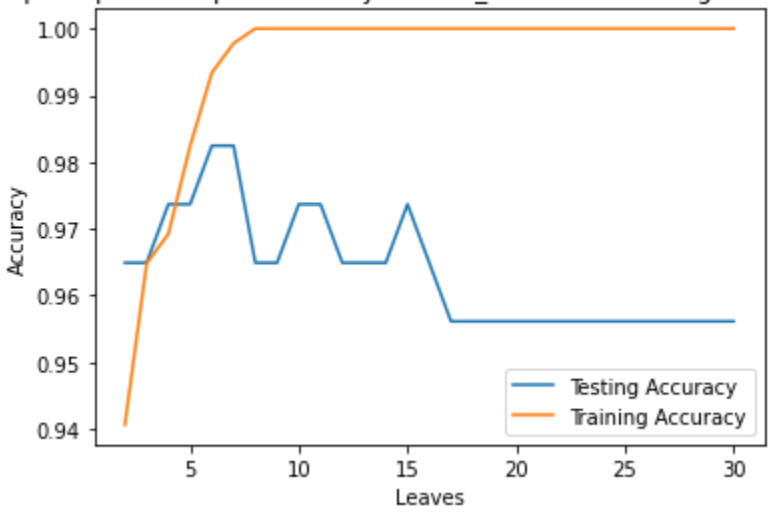
For Graph Depth 5 the plot Accuracy vs Num_Leaves on Training and testing Data



For Graph Depth 6 the plot Accuracy vs Num_Leaves on Training and testing Data



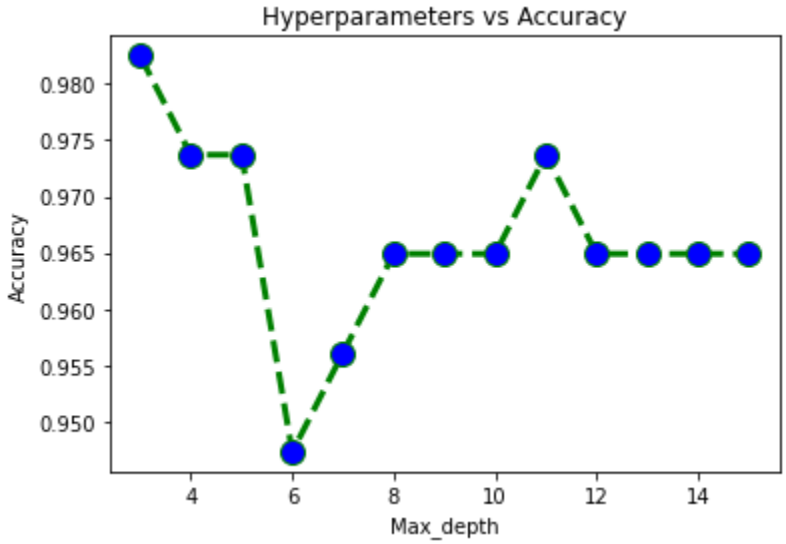
For Graph Depth 7 the plot Accuracy vs Num_Leaves on Training and testing Data

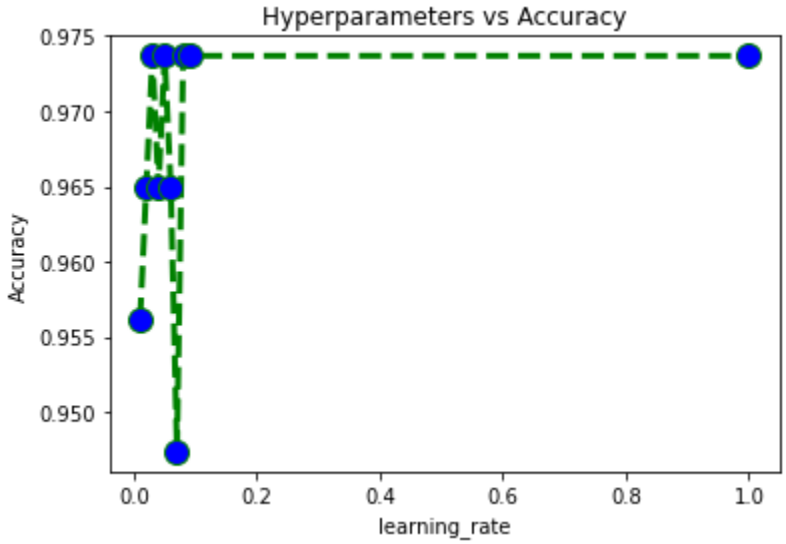


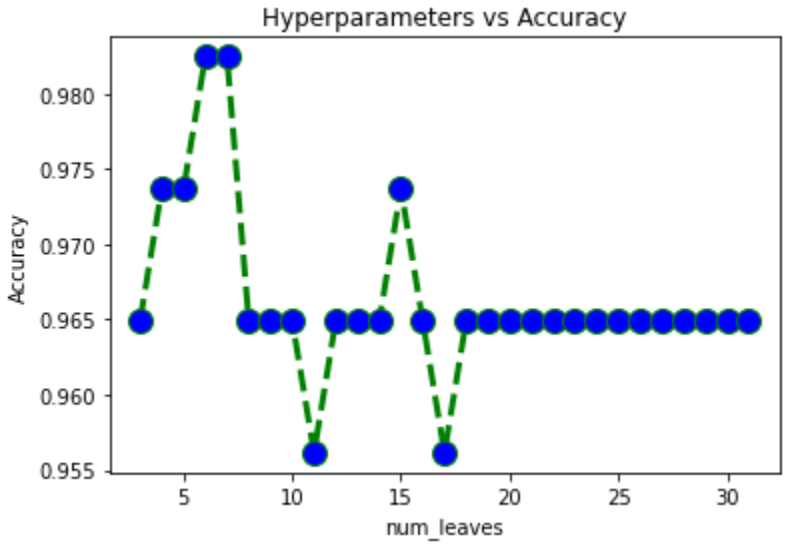
As we can see after a certain amount of leaves the model saturates and provides us with the same result. As we can see after num_leaves equal to 6, 7, the model starts to overfit as the training accuracy increases and the testing accuracy decreases. The model starts to overfit after num_leaves equal to 6, 7 and gives lower accuracy on the test data.This indicates that the model is overfitting

8. Parameters can be used for better accuracy:
    a. Using small `learning_rate` with `large num_iterations`
    b. Using large num_leaves

c. Using large max_bin







Parameters which can be used to avoid overfitting:

a. Using small max_bin
b. Using small num_leaves