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, outliers in the area column were removed and the necessary columns were encoded. The implementation of which can be found in the notebook.

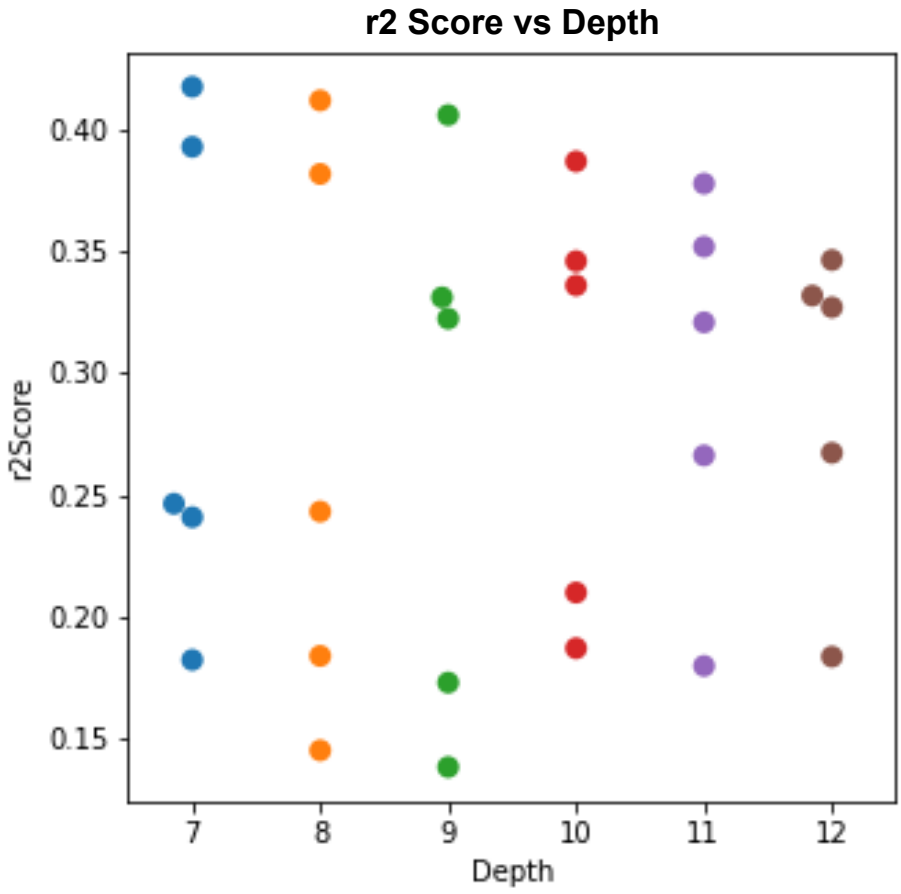
1. A simple decision tree regressor was implemented from **scratch** to predict the price of the houses without any validation and the r2 score that we got was 0.35434.
2. A k-fold function was implemented with a default value of k equal to 5. A code of which can be found below:

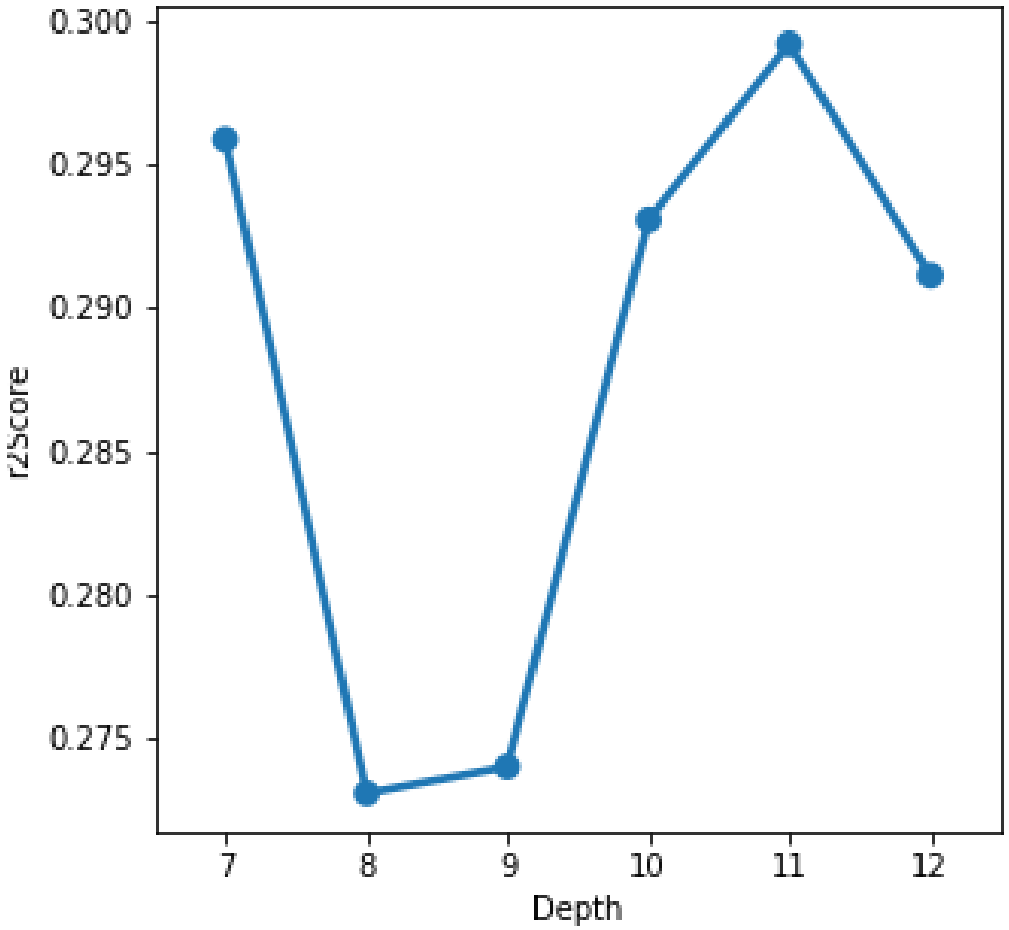
```
def kFold(x,y,model,k=5):
    xlist = []
    ylist = []
    part = int(len(x)/k)
    for i in range(k):
        tempx = [x[j] for j in range(part*i ,part*i + part)]
        tempy = [y[j] for j in range(part*i, part*i + part)]
        xlist.append(tempx)
        ylist.append(tempy)
    res = []
    for i in range(k):
        xtrain = []
        ytrain = []
        xtest = []
        ytest = []
        for j in range(k):
            if i!=j:
                xtrain.extend(xlist[j])
                ytrain.extend(ylist[j])
            else:
                xtest.extend(xlist[j])
                ytest.extend(ylist[j])
        model.fit(np.array(xtrain),np.array(ytrain))
        ypred = model.predict(np.array(xtest))
        res.append(r2(ytest, ypred))
    return res , np.mean(np.array(res))
```

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

Fold	R2 Score
1	0.240
2	0.392
3	0.246
4	0.417
5	0.295

Average of all the r2 scores is 0.295.
Max depth of 7,8,9,10,11,12 were tested to find the best fit possible.





As we can see depth 7 or 11 gives us the best r2 score. After 11 depth, the model starts to overfit. The default value of k which was chosen was equal to 5. Other values were also tested which favoured to use the max depth as 11.

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 can be found below:

```
def baggingSplit(x,y,sampleSplit):
    returnArray = []
    ix = np.random.randint(0,len(x),int(sampleSplit * len(x)))
    returnArray.append(x[ix].copy())
    returnArray.append(y[ix].copy())
    return returnArray
```

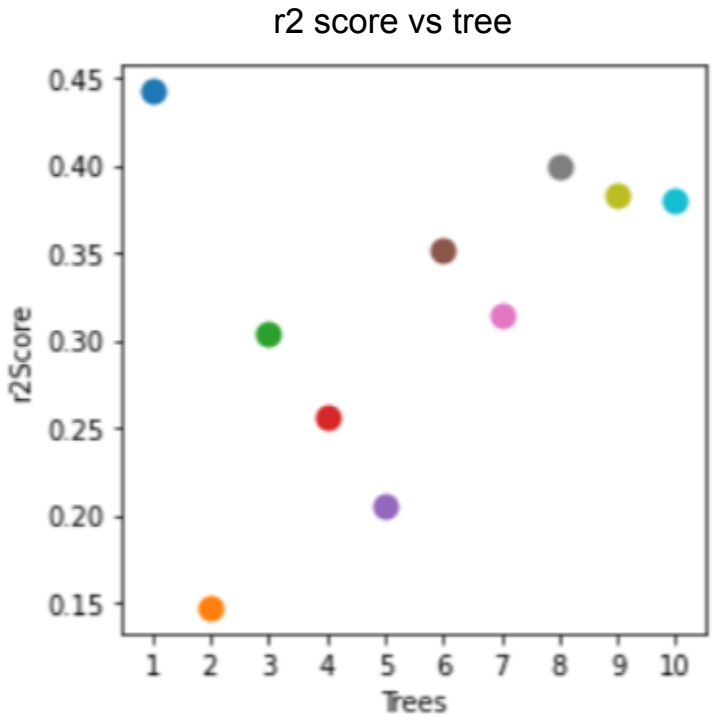
Bagging code can be found below:

```
def bagging(model,xtrain, ytrain, xtest, ytest, sampleSplit = 0.8,nEstimators = 10):
    res = {}
    count = 0
    trees = {}
    yPredict = []
    while count != nEstimators:
        array = baggingSplit(xtrain,ytrain,sampleSplit)
        xTr = array[0]
        yTr = array[1]
        model.fit(xTr,yTr)
        yPr = model.predict(xtest)
        if r2(yPr,ytest) >= -0.2:
            res[count+1] = (r2(ytest,yPr))
            trees[count+1] = model.treeNode.copy()
            yPredict.append(yPr)
            count+=1
    avgY = []
    for i in range(len(ytest)):
        sum = 0
        for j in range(nEstimators):
            sum+=yPredict[j][i]
        avgY.append(sum / nEstimators)
    return np.array(avgY), res , r2(ytest, avgY), trees
```

5. Different datasets were generated with 0.8 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 parallelly.
6. This is how different trees performed in the process of bagging.

```
r2 Score for each tree:
1 ==> 0.4422121392879107
2 ==> 0.14727258937778298
3 ==> 0.30451698316595577
4 ==> 0.2566231133170297
5 ==> 0.20506185825028334
6 ==> 0.35169445107484754
7 ==> 0.31378167324732165
8 ==> 0.39915867589625 93
9 ==> 0.38274376916178987
10 ==> 0.38051741836072994
```

On average they provide a r2 Score of 0.32 which is not that good until the predictions are averaged to provide us a good model.



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

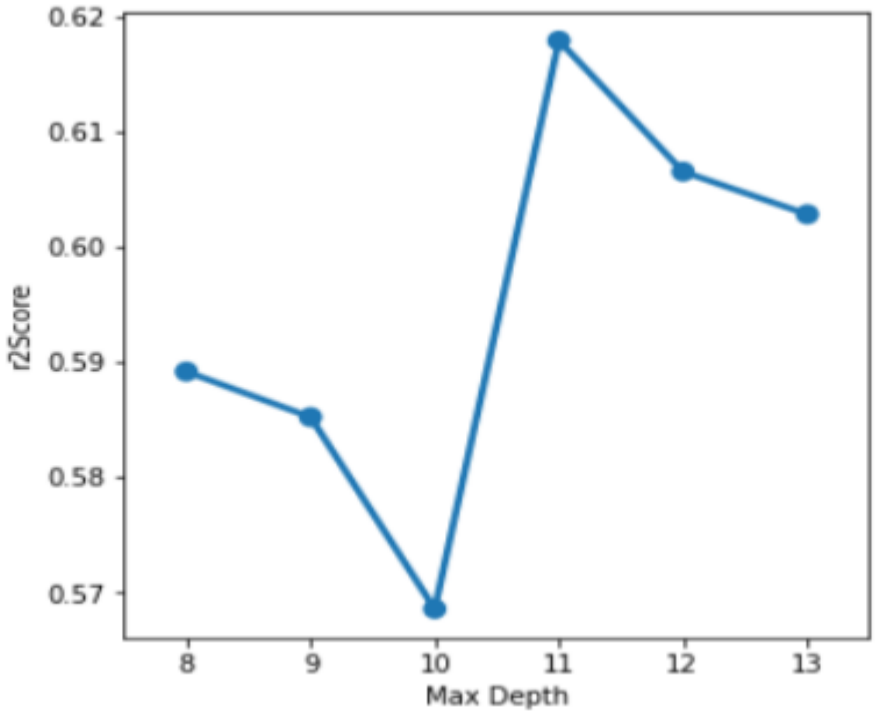
7. Trees were combined to get the average of the predictions. This method reduces the variance and provides us with better results. The predictions were combined and the results are below:

Trees combined	r2 Score
10	0.61

As we can see, averaging and bagging gave us far better results than using a single decision tree which gave only about half the r2 score. Depth was changed to see the results which are shown in the next part.

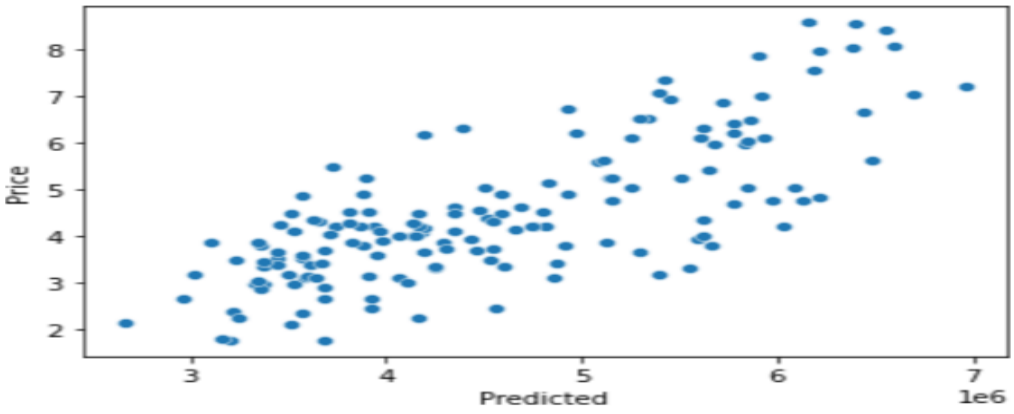
8. Depth was varied from 8-13, where depth 11 gave us the best results. Results are shown below:

r2 Score vs Depth



As we can see before depth 11, the model underfits and does not unleash its full capability. After depth 11, the model overfits as we can see slowly, the r2 score starts decreasing. So depth 11 gives us the best results possible. So, the r2 score decreases before and after depth 11.

Final results:



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

Model	MSE	MAE
Random Forest Regressor	824452182878.3085	702370.0424

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

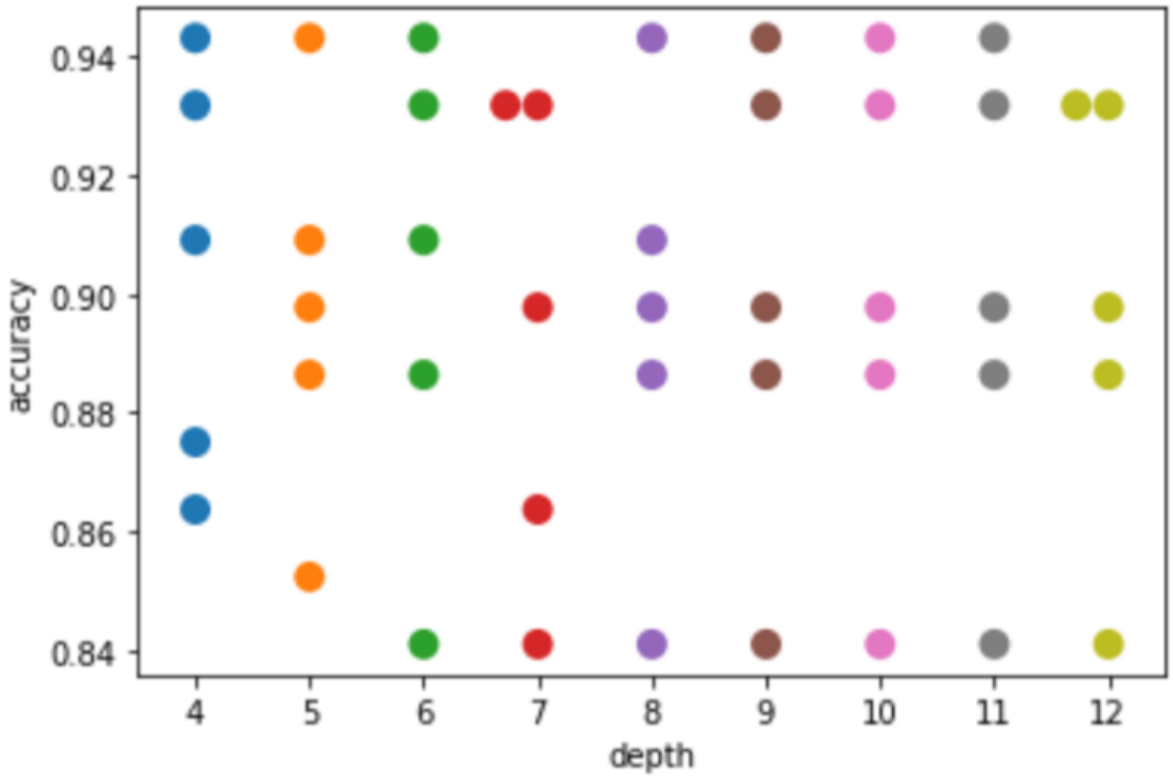
Model	MSE	MAE
AdaBoost	1068487117854.6982	803345.836189

Question 2 - Boosting

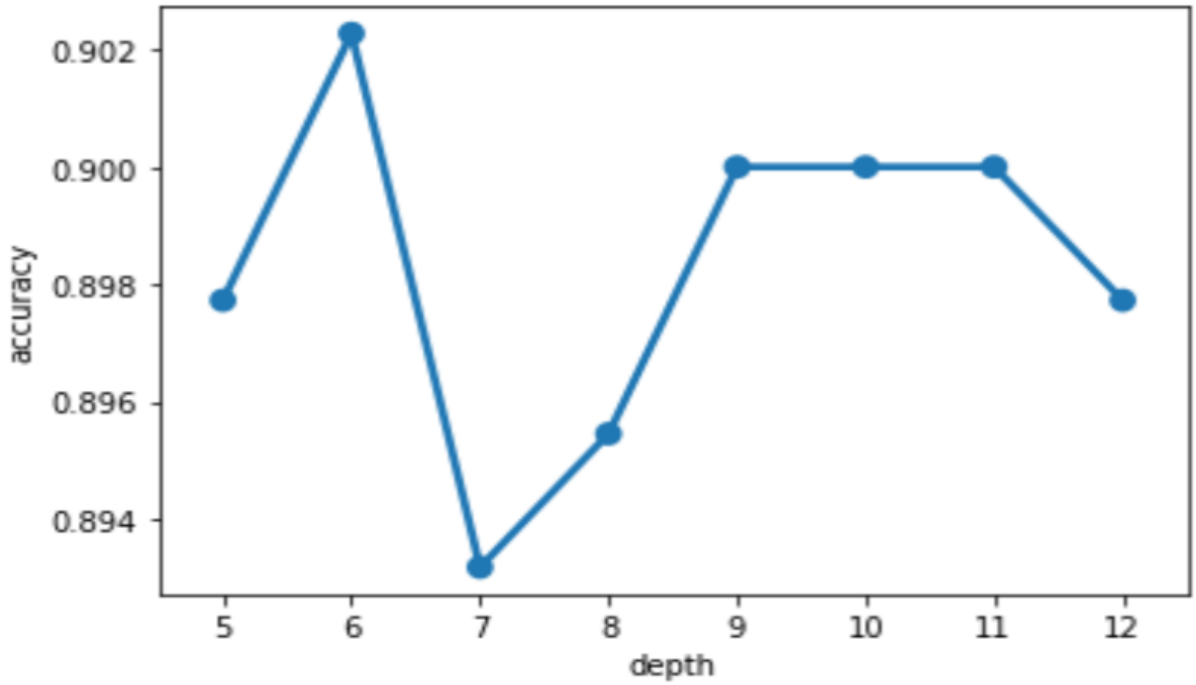
Boosting is an ensemble modeling technique which attempts to build a strong classifier from the number of weak classifiers. It is done by building a model using weak models in series.
Simple preprocessing techniques like removing outliers was done on the dataset. Visualisation and reasoning of which 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: Accuracy : 90 percent
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 6 gave us the best results and after it the model overfits. Below 6 the model underfits and the results are not that great. Minimum accuracy was seen at depth 7.
   Results are shown below through a pointplot between Accuracy and Max depth.

Accuracy on different folds vs depth



Average accuracy vs depth

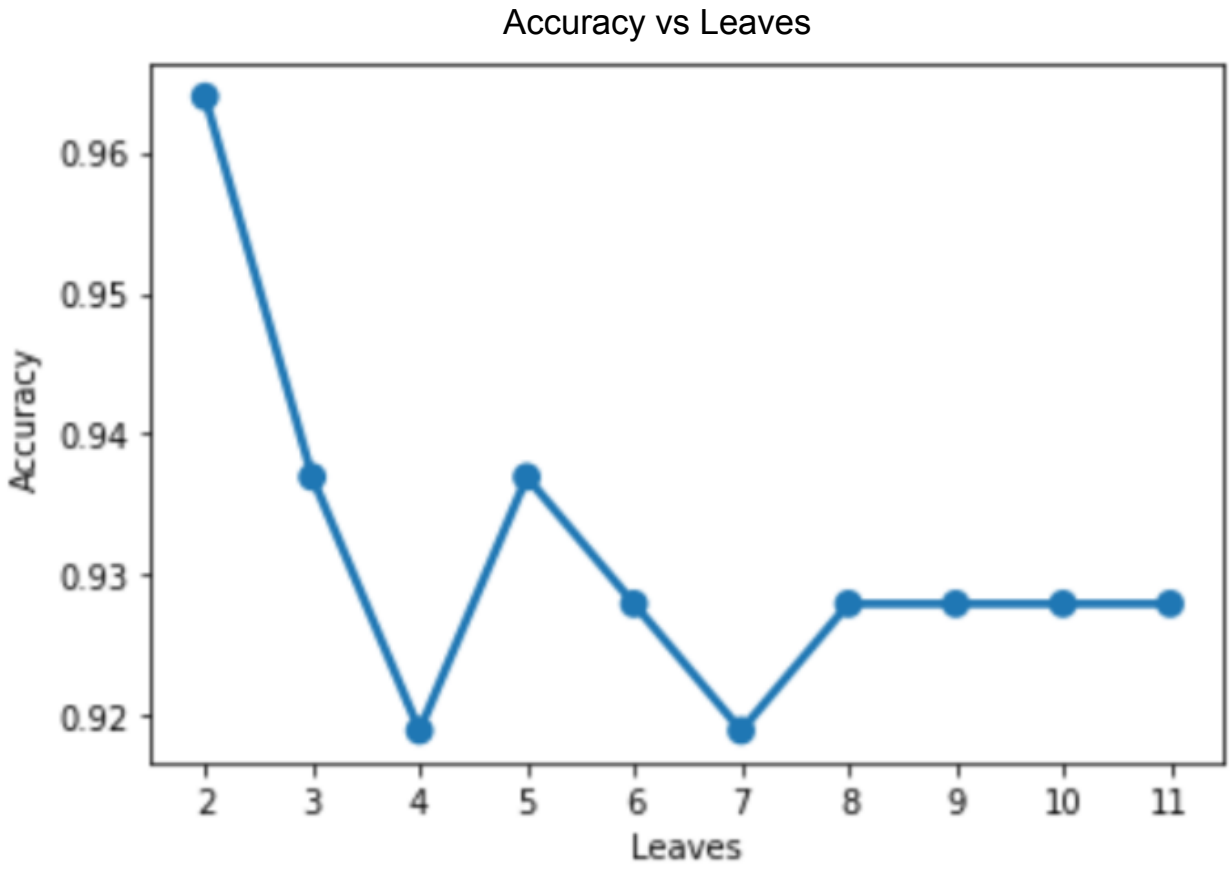


3. The best results were found for max depth equal to 6 as after that model overfits and below that model underfits. The plots above justify the statement.
4. XGBClassifier was implemented with max_depth = 4 and subsample = 0.7. Accuracy was calculated and printed. Results are shown below.
5. Confusion Matrix

41	2
5	63

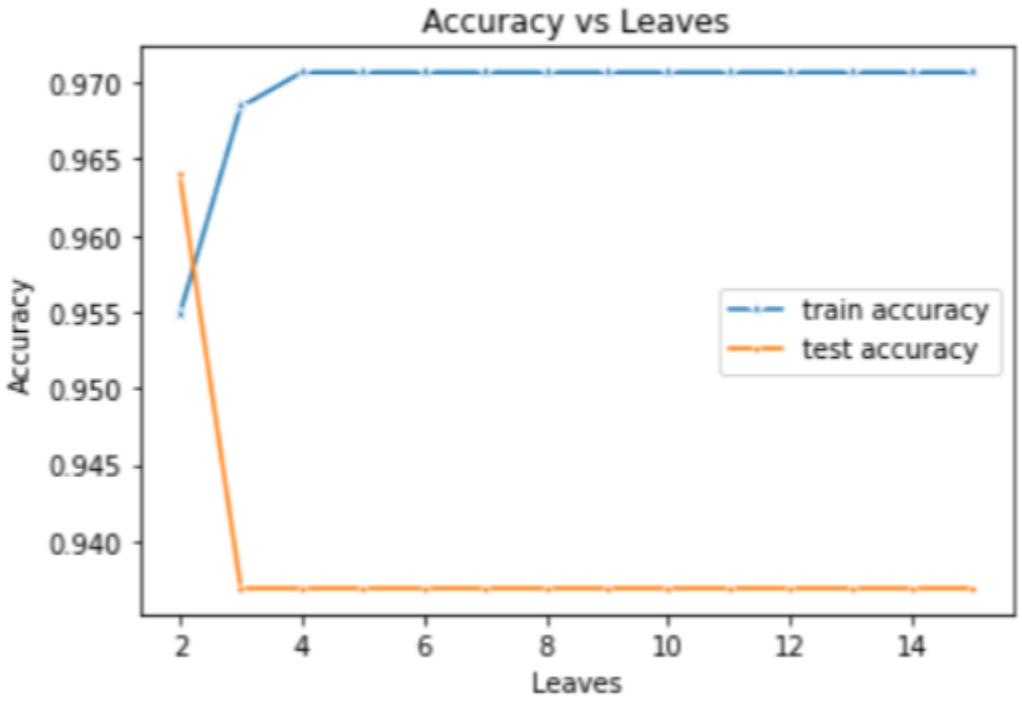
Accuracy reported on X_test = 93.69 %
Accuracy reported on X_train = 98.8 %

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 11 and a line plot was plotted between the accuracy and the number of leaves :

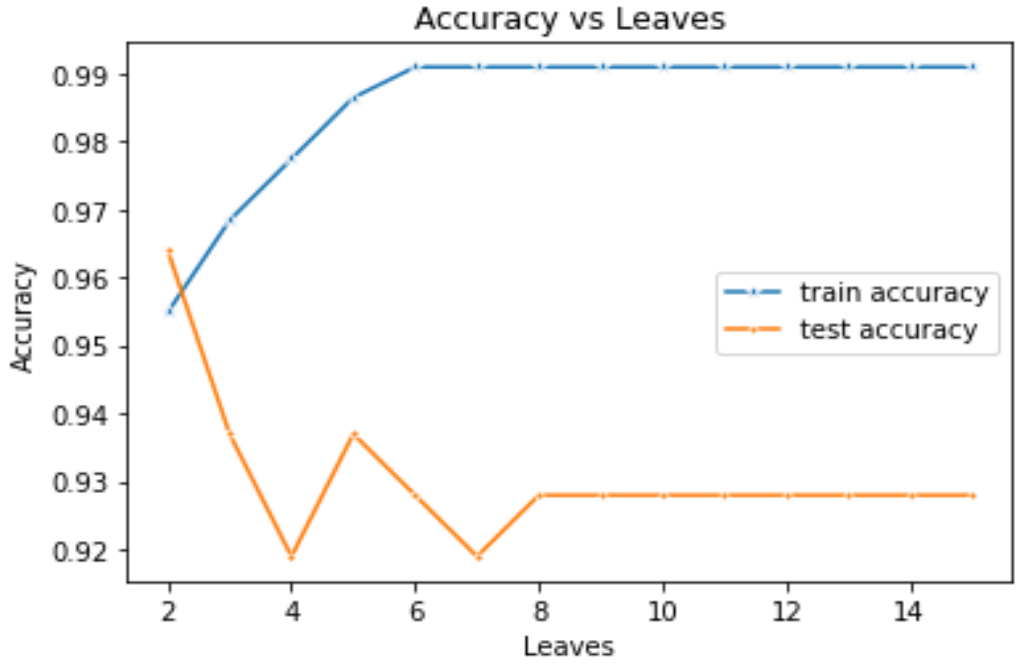


As we can see from 8 we get the same result on the X_train and highest accuracy was achieved for num_leaves equal to 2.

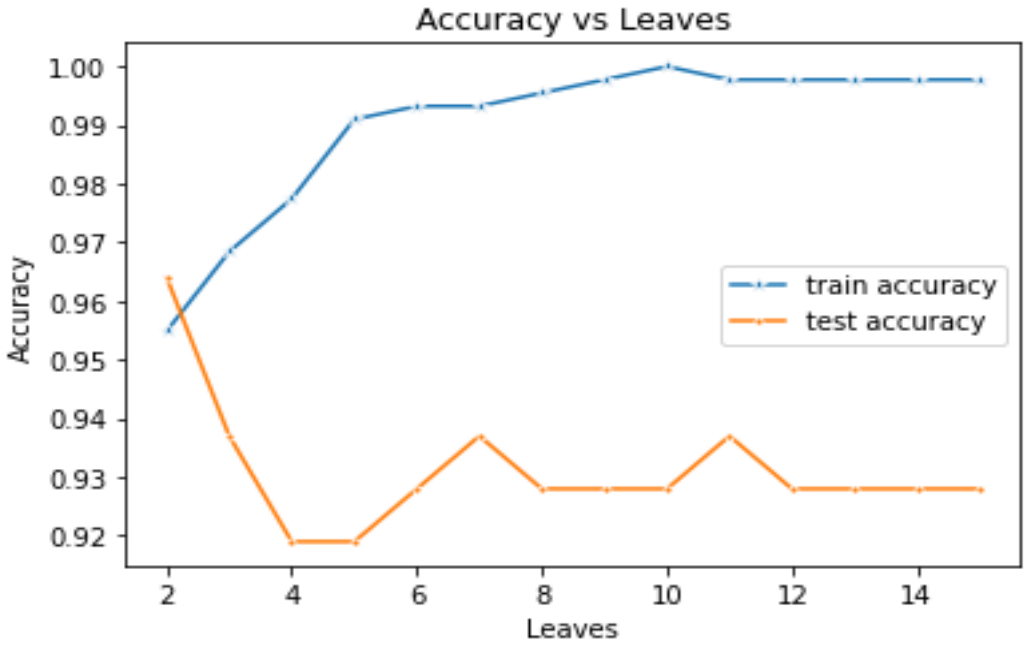
7. Training and testing accuracies were plotted for different leaves and varying depth.
   Graph for depth = 2



   Graph for depth = 3



Graph for depth = 4



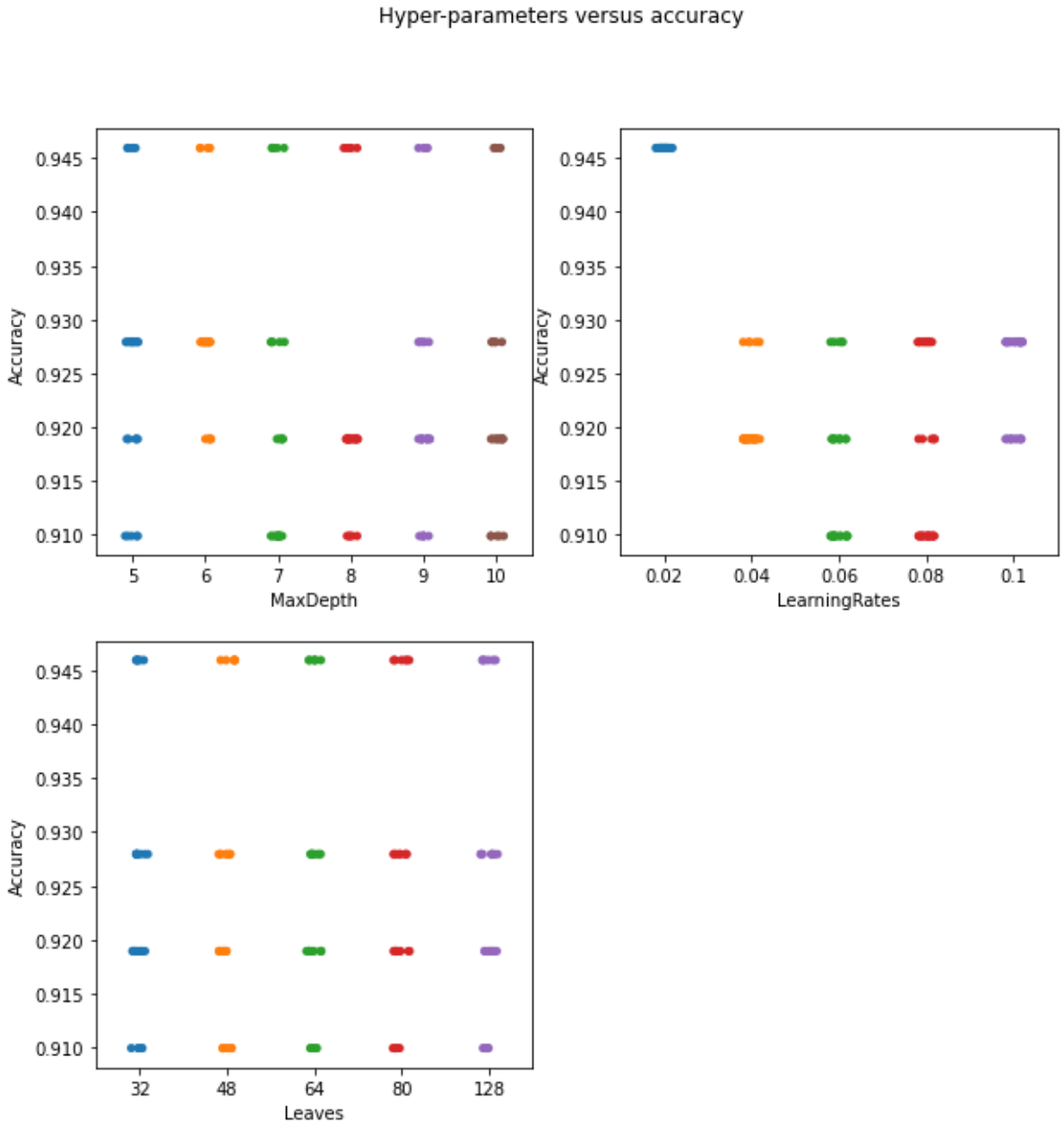
As we can see after a certain amount of leaves the model saturates and provides us with the same result. As we can see after num_leaves equal to two, the model starts to overfit as the training accuracy increases and the testing accuracy decreases. The model starts to overfit after num_leaves equal to 2 and gives lower accuracy on the test data.

This indicates that the model is overfitting.

8. Parameters which were varied to find the best accuracy were learning rate, num_leaves and max_depth. Maximum accuracy was achieved for the parameters:

Parameter	Best value
Learning Rate	0.02
MaxDepth	10
Number of Leaves	128

Graphs have been attached below:



Overfitting can be reduced by introducing some new parameters like subsample in the model and also reducing the num_leaves in the model. Best results were found for `[0.02, 128, 10, 1.0]` parameters where 1 is the subsample. 94 percent accuracy was found on the dataset which was way better than the 90 percent accuracy which was calculated before optimising.