Lab 3
Abu Shahid
B20CS003

Question 1

- Data was loaded and necessary pre-processing was done.
- Ocean_proximity feature was One_hot_encoded.
- Data was split in train-validation-test set in ratio 70:10:20.
- MSE function and R2 Score function was built from scratch.

```
def mse( arr1, arr2):
  l= len(arr1)
  sum=0
  for i in range(l):
    sum+= (arr1[i]-arr2[i])**2
  sum=sum/l
  return sum
```

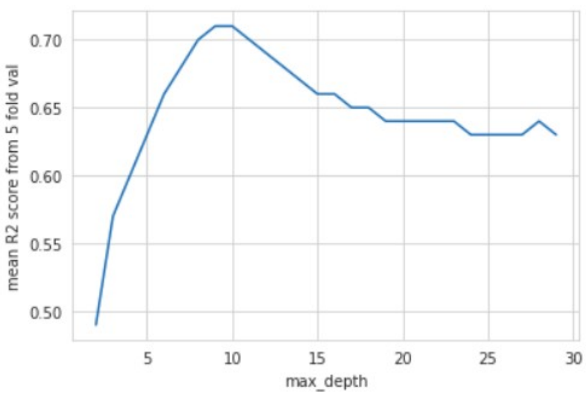
```
def r2_score(y, y_pred):
  y=np.array(y)
  y_pred=np.array(y_pred)
  # print(len(y_pred)) #<<<<<<<<<<
  mn=np.mean(y)

  var_mean=np.var(y-mn)
  var_line=np.var(y-y_pred)
  score= 1- (var_line/var_mean)
  return score
```

SubTask 1:

- Decision Tree Regressor was implemented using skleaarn
- We got an MSE of 4573453901.069
- And an R2 score of 0.6656
- No hyperparameter tuning was performed as it wasn't demanded.

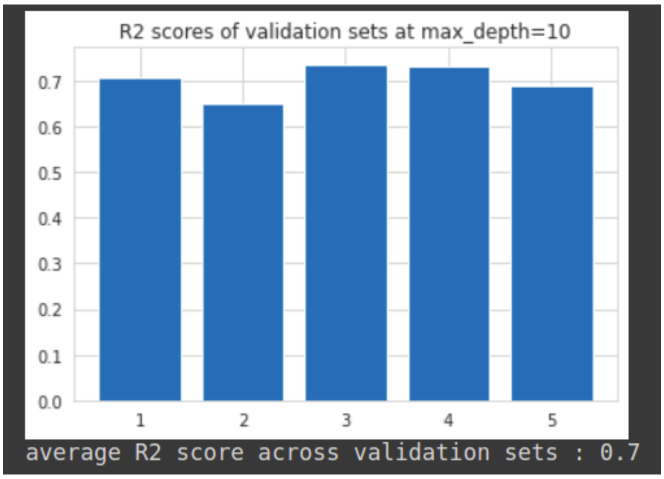
Subtask 2 and 3:

- Cross Validation function was initialized from scratch, which returns K- R2 scores where K is the number of folds(5: default)
- max_depth hyperparameter was varied from 2 to 30 and 5-fold cross validation scores was computed.
- Average validation score for each depth and variance was stored and plotted.



- Clearly 9 and 10 are best contenders of max_depth but less variance was reported for when max_depth was equal to 10.
- Therefore, max_depth=10 would be ideal choice.

- Results across validation sets has been visualized and summarized as above.



Subtask 4:

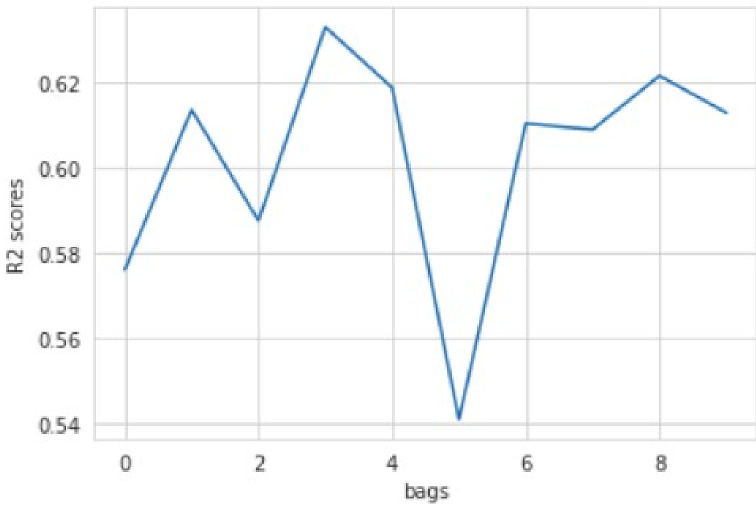
- Function to perform bagging was declared with default ratio=0.5 of original dataset.

```
def bag(X, y, count, ratio=0.5):
```

- Given the parameters, bag() returns a list consisting of count number of datasets, each of which is half the size of our original dataset.
- This function was then called as
    - bags= bag(X_train_set, y_train_set, 10)

Subtask 5 and 6:

- A decision tree regressor was then trained on these datasets.
- The R2 scores we got across them are follows:
    - [0.5760751154243102, 0.613609535061817, 0.5876067967115476, 0.6330449136537518, 0.6188192303380817, 0.5410437584026953, 0.6104321143966763, 0.6089858393537038, 0.6216001830988395, 0.6129454639056615]
- Average R2 score across these 10 datasets is: 0.602416

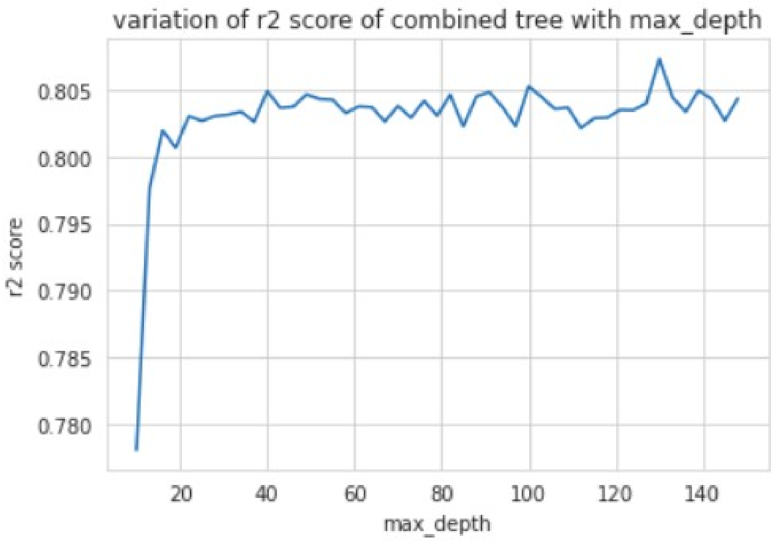


Subtask 7:

- Next we combine the trees we got from 10 datasets.
- This was done as follows
  - The predictions were made using all the 10 trees.
  - The prediciton for each instance was added across all the trees and then divided by 10 and stored as final predicitions.
- The R2 score of combined tree is: 0.8043, which is better than any individual tree’s R2 score.

Subtask 8:

- The R2 score of the combined ensemble does not improve much on increasing the max_depth.
- Rather, it fluctuates on increasing the max_depth.
- On decreasing the max_depth, R2 scores takes a dip on reducing the max_depth below 24,



Subtask 9:

- Next, RandomForestRegressor was trained and the result of the precition can be summarised as follows.

```
MSE from  RandomForestRegressor : 2365355065.343219
MAE from  RandomForestRegressor : 31604.338507462686
R2 from  RandomForestRegressor : 0.8270329948576377
```

Subtask 10:

- Next, AdaBoostRegressor was trained and the result of the precition can be summarised as follows.

```
MSE from  AdaBoostRegressor : 7428371909.992746
MAE from  AdaBoostRegressor : 72803.85299212305
R2 from  AdaBoostRegressor : 0.45679899767240306
```

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. First, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

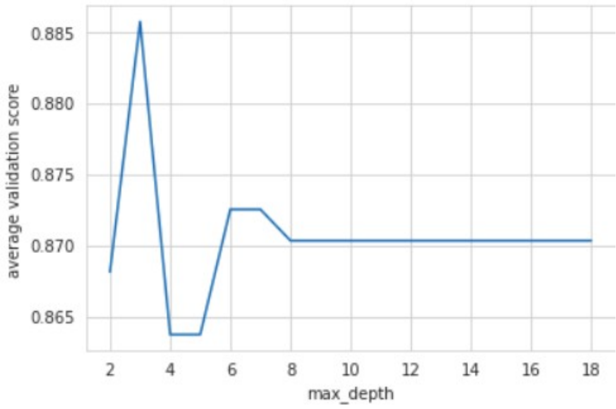
- Data was loaded and necessary pre-processing was done.
- Necessary data visualization was done to get insight of the data.
- Data was split in train-validation-test set in ratio 70:10:20.
- gini_index and Decision Tree Clasifier was initialized from scratch.
- Function to accuracy was built from scratch too.
- Cross validation function based on accuracy was also defined from scratch.

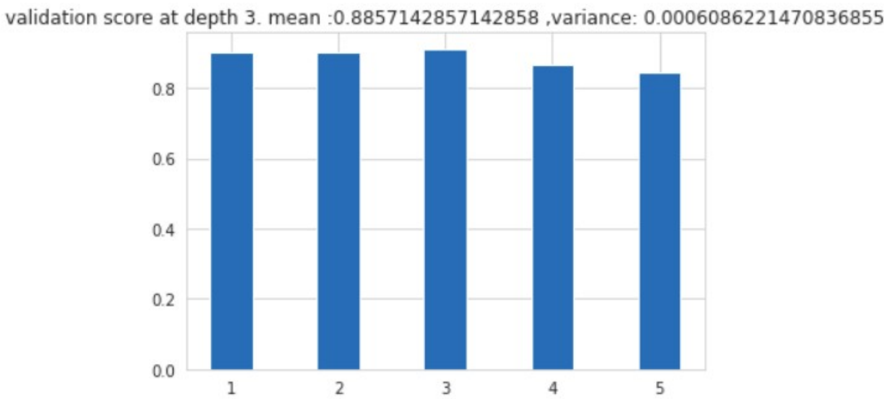
Subtask 1:

- Our Decision Tree was then trained on training data and we got an accuracy of 0.9035 off our testing data.

Subtask 2 and 3:

- Next, our DT was trained by varying max_depth from 2 to 20 and average of cross validation scores was plotted to select best value of max_depth.
- The result of above exercise can be visualized and summarized as following plot.





- The validation scores quickly reach a maxima at max_depth=3 with steep fall on both sides.
- However when max_depth increases, our validation scores stagnate and become constant indicating the limit of our model and that it is not learning anymore.

Subtask 4 and 5:

- XGBoost was implemented using sklearn with subsample-0.7 and max_depth=4.
- Accuracy on training set and testing set can be summarize as follows:

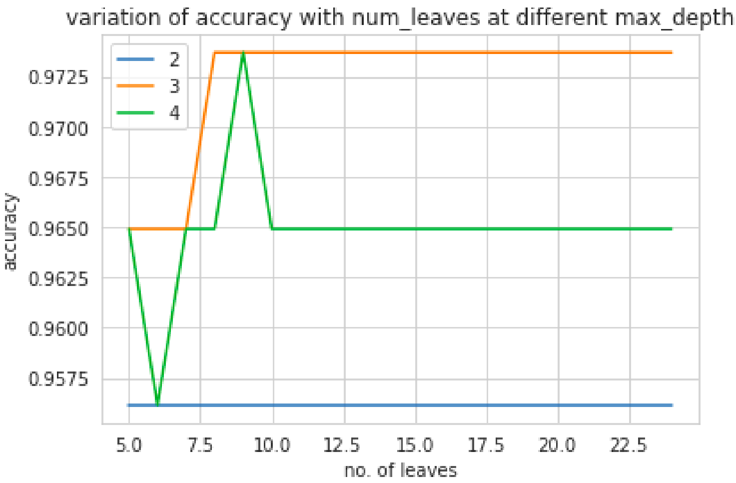
```
accuracy of training data:  0.9874371859296482
accuracy of testing data:  0.9385964912280702
```

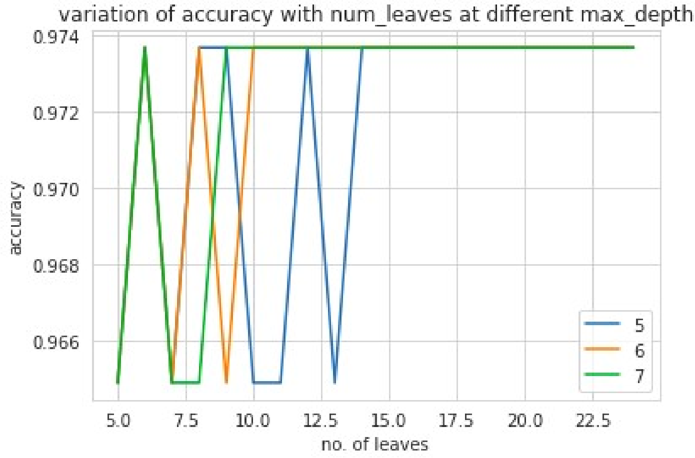
Subtask 6, 7 and 8:

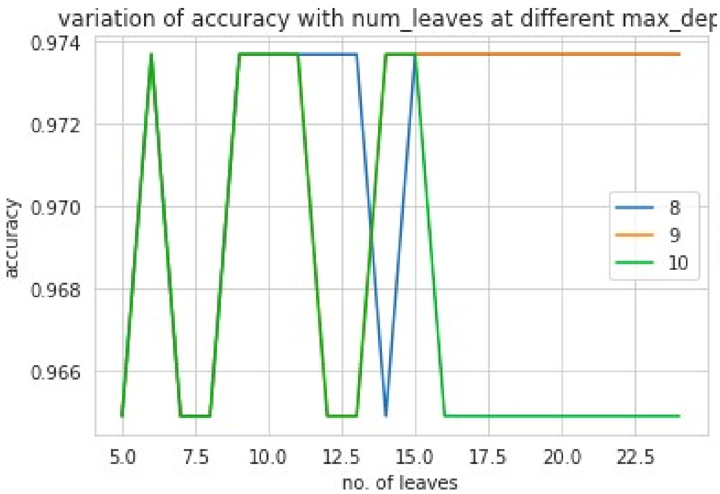
- Next we implement LightGBM with max_depth=3 and num_leaves were varied.
- Model was fitted with training data and predictions weer made.
- Results can be summarized as follows:

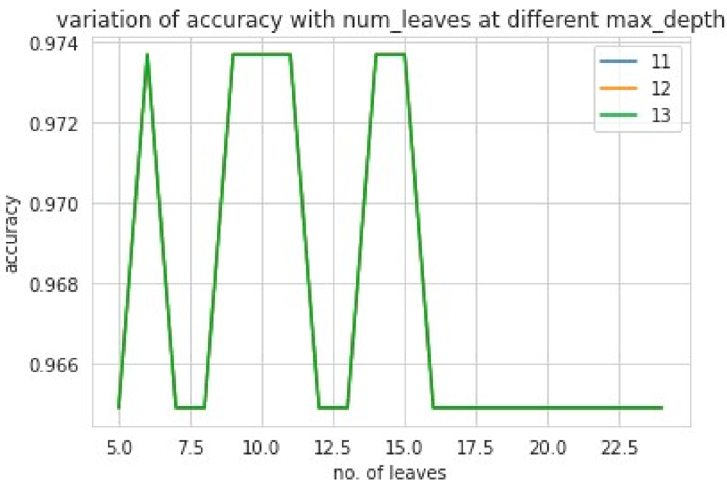
```
model performance at depth 4 and num of leaves : 10
              precision    recall  f1-score   support

           0       0.95      0.98      0.97        43
           1       0.99      0.97      0.98        71

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114



model performance at depth 4 and num of leaves : 15
              precision    recall  f1-score   support

           0       0.95      0.98      0.97        43
           1       0.99      0.97      0.98        71

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114



model performance at depth 4 and num of leaves : 20
              precision    recall  f1-score   support

           0       0.95      0.98      0.97        43
           1       0.99      0.97      0.98        71

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114



model performance at depth 4 and num of leaves : 25
              precision    recall  f1-score   support

           0       0.95      0.98      0.97        43
           1       0.99      0.97      0.98        71

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

- Next we analyze the relation between max_depth and num_leaves.
- Same can be visualized as:









- Analysis:
  - With enough number of leaves, accuracy stops fluctuating and our model stops learning any new feature, for a given max_depth.
  - When max_depth is low(less than 5), variation of accuracies with increase in num_leaves is rather small and is consistent.
  - However, if max_depth is high, our accuracies first increase, reach their maxima and then take a local maximum.
- From the above analysis, max_depth hyperparameter is best for controlling and tweaking accuracy while,
- num_leaves hyperparameter is more suited for controlling ovrfitting of our model.