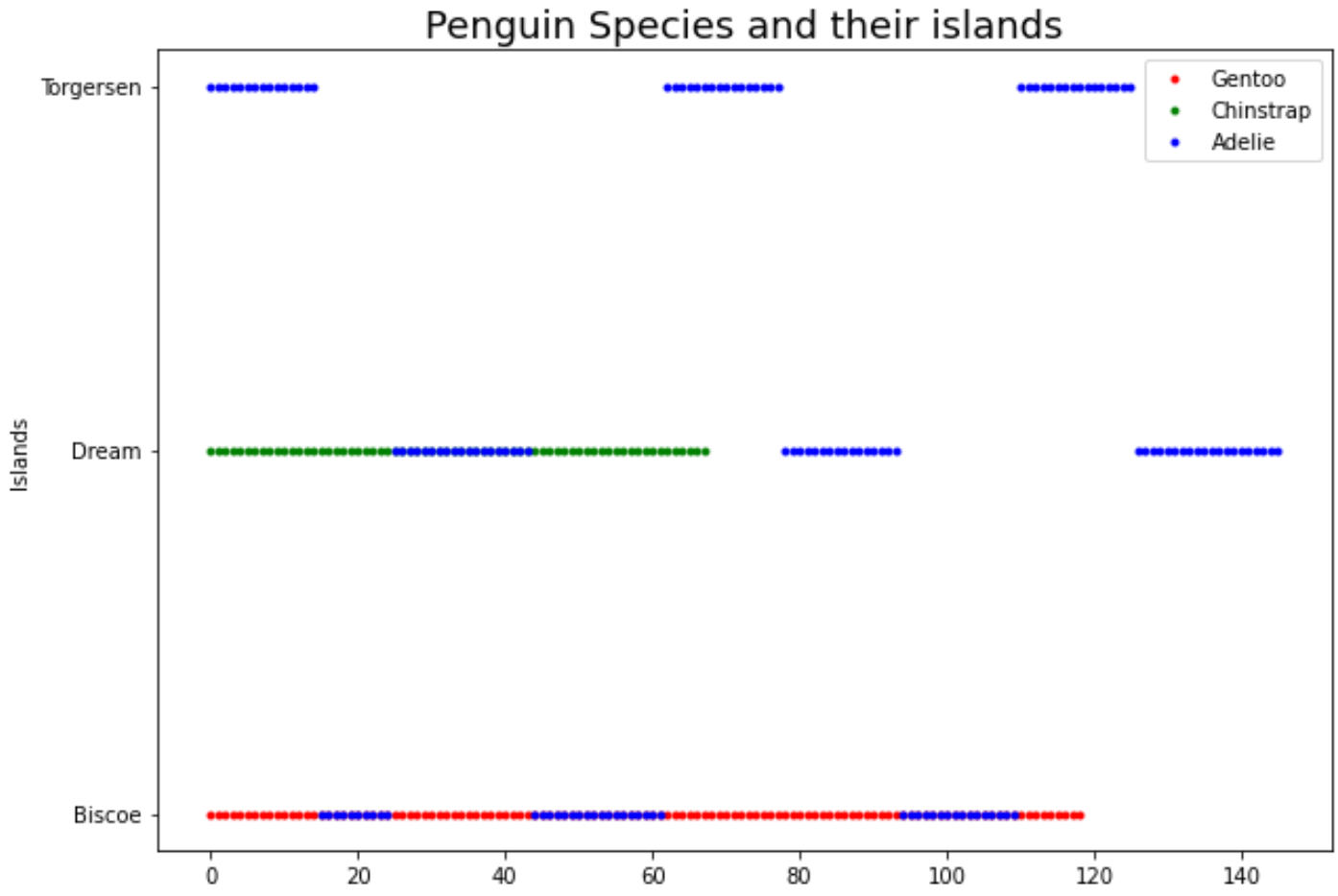
Maniya Yash Rajeshbhai
B20CS033

ML Lab 2 Report

Question 1 :

1. Preprocessing :

- There were several null values, which were eliminated first.
- Then I analysed the island and species relationships by visuals through matplotlib.pyplot. I found out that there were just 3 islands, out of which Torgersen island had just one species as habitants : Adelie. This could lead to information gain 1 of Adelie class while splitting via Island feature.
- Also, then I encoded the island feature to integers.
- After that, the data was splitted into training and testing data with a ratio of 75 : 25.



2. Cost Function :

The gini_index() function takes the dataset as an argument and returns the data's gini index.

Gini index = 1 - ∑p^2

Where p = probability of finding the respective label.

3. Continuous to Categorical values conversion :

For the required purpose, cont_to_cat() function is used, which takes dataset as argument and returns best_split dictionary consisting of best split details when a particular feature is split. These details include Information Gain, Threshold, Feature index, etc. This in turn happens after the data is converted into categorical data.

4. Building the Decision Tree :

- For building a tree, there must be a class Node. So first of all I created a Node class with several attributes : left, right, feature_index, threshold, value.
- After this I made a build_decision_tree() recursive function which takes dataset and current depth as arguments and returns a decision tree model.
- Build tree creates a best split using cont_to_cat() function, and recursively traverses to its left and right subtrees.
- The recursion stops at the base case when the Node is a leaf Node or current depth gets greater than the maximum depth.
- And hence finally the tree's root node is returned. Now that Decision Tree is our training model.

5. Adding important properties to build tree function :

As mentioned above, the max_depth property is added to build tree function for termination of the recursion. Also, it will terminate when no information gain is happening, i.e. it encounters a leaf node. The model is trained using the training data using the train_model function.

6. Classifying the Testing Samples :

For the given purpose, a classify() function is used which takes trained_model and X_test as arguments and returns a list which consists of classified samples. The classify function uses another recursive function predict() for predicting the respective classes.

7. Evaluating the Trained Model :

- For evaluation, accuracy_score is used from sklearn.metrics library. The accuracy of the model turns out to be 97.62%.
- I designed a 3X3 confusion matrix for the same and evaluated class wise accuracies. Given below are the accuracies I got.

```
Overall Accuracy for trained Model : 97.62 %

Confusion Matrix :
[[32  0  0]
 [ 0 33  1]
 [ 0  1 17]]

Class wise accuracy :

Class Accuracy for Gentoo : 100.00 %
Class Accuracy for Adelie : 97.06 %
Class Accuracy for Chinstrap : 94.44 %
```

Question 2 :

1. Preprocessing :

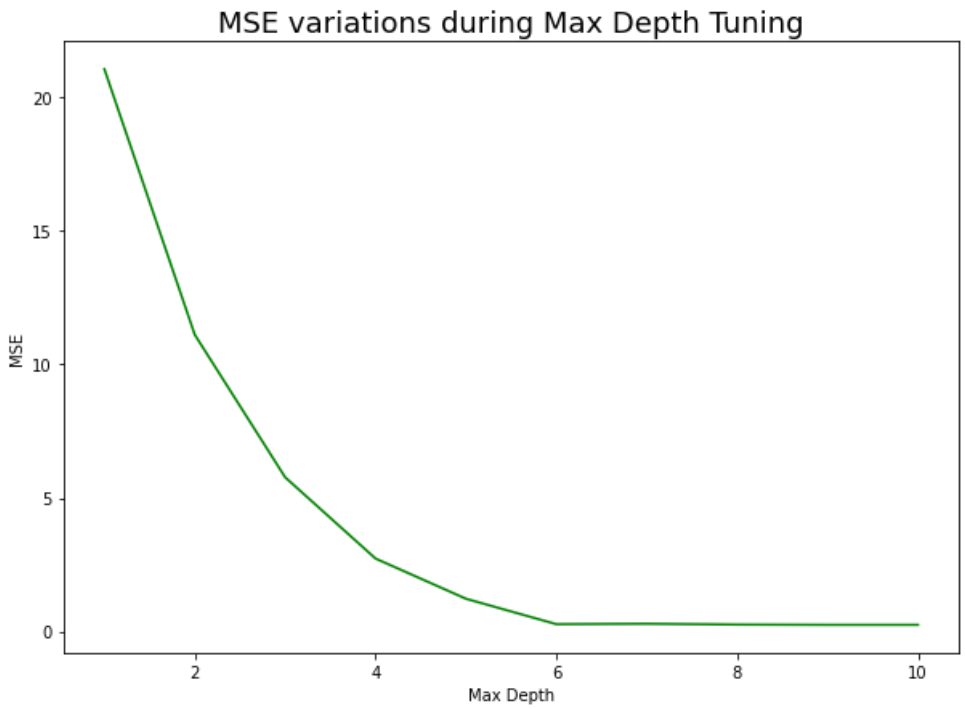
- First of all, I dropped all the NaN values. I saw that the 'X5' feature had just two values 7 and 3.5, which can be scaled to 1 and 0. So I scaled the 'X5' feature using MinMaxScaler().
- Then I splitted the data into training, validation and testing sets respectively in the ratio 70 : 10 : 20.

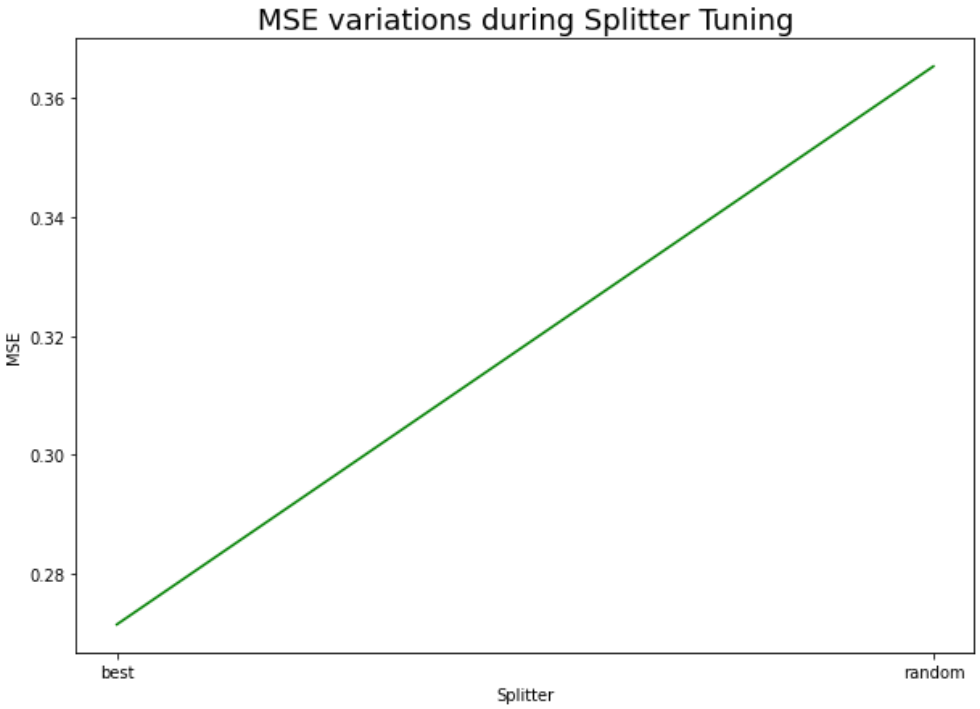
2. Training the Model using Hyper-Parameters :

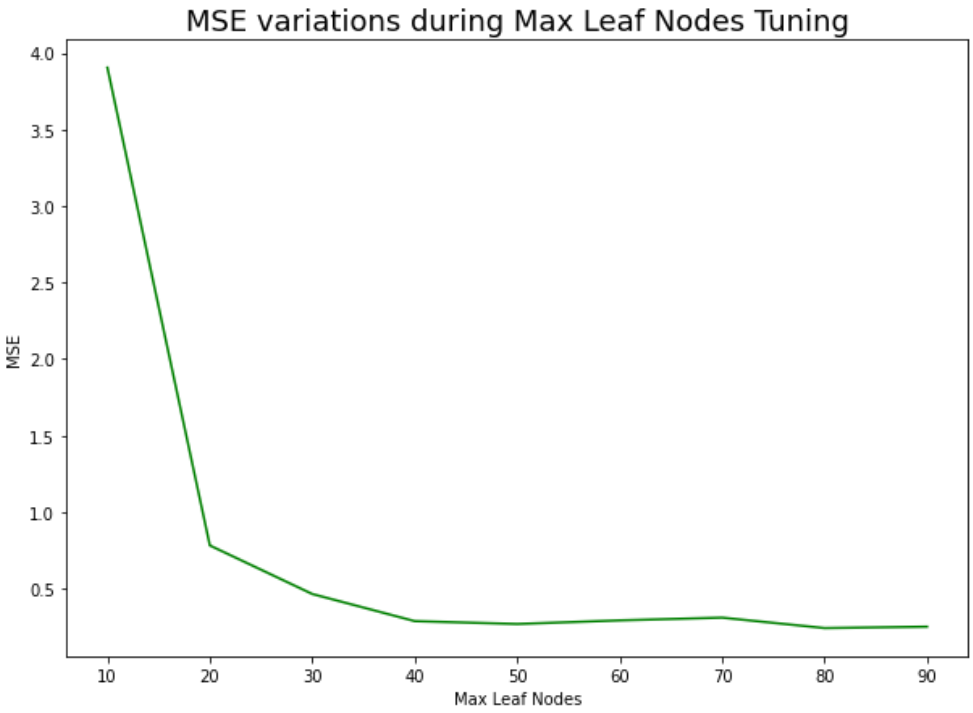
- I used sklearn DecisionTreeRegressor() model for the training and validation purpose. Initially I trained the data using a default DecisionTreeRegressor without any setting parameters. With all the models, I used one common parameter criterion as 'squared_error' as our need was to test the model using MSE.
- For the default model, I got an MSE of 0.2834 (may vary when run again).
- Then I tried setting different parameters to the DecisionTreeRegressor. I selected a couple of parameters : Splitter, Max Depth, Max Leaf Nodes and Min Samples Leaf.

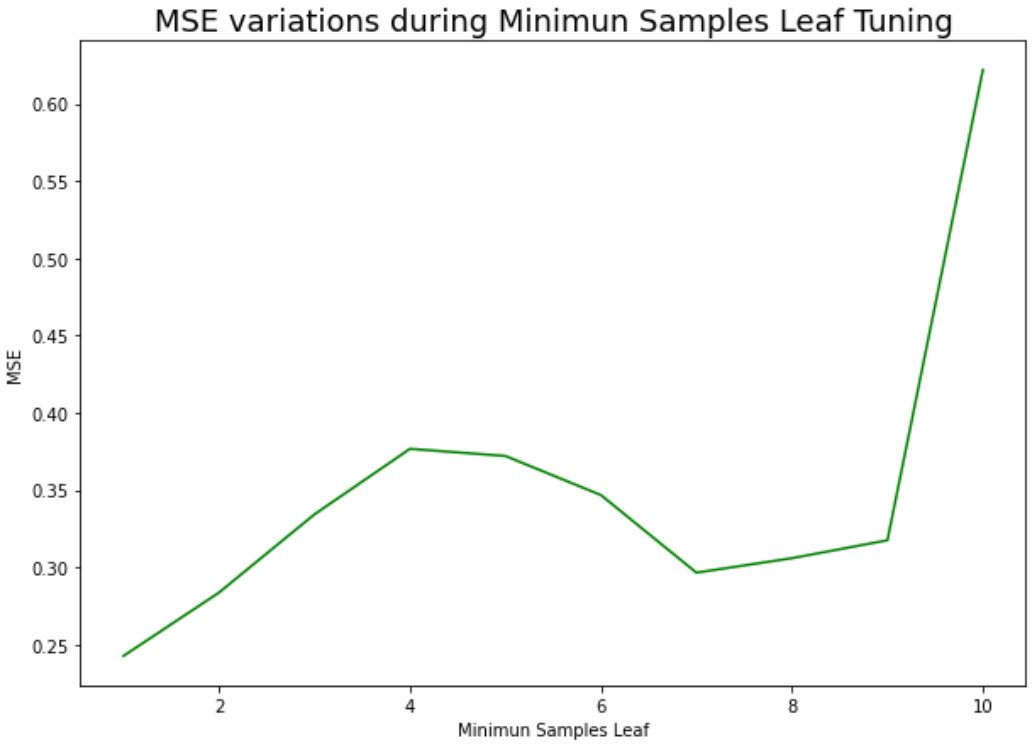
- Understanding behind the selected Hyper-parameters :
    - **splitter** : This attribute has 2 values : [‘best’ , ‘random’]. The strategy used to choose the split at each node. Supported strategies are “best” to choose the best split and “random” to choose the best random split.
    - **max_depth**: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
    - **min_samples_split** : It is the minimum number of samples required to split an internal node. As this value increases, usually the MSE value increases.
    - **min_samples_leaf** : The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

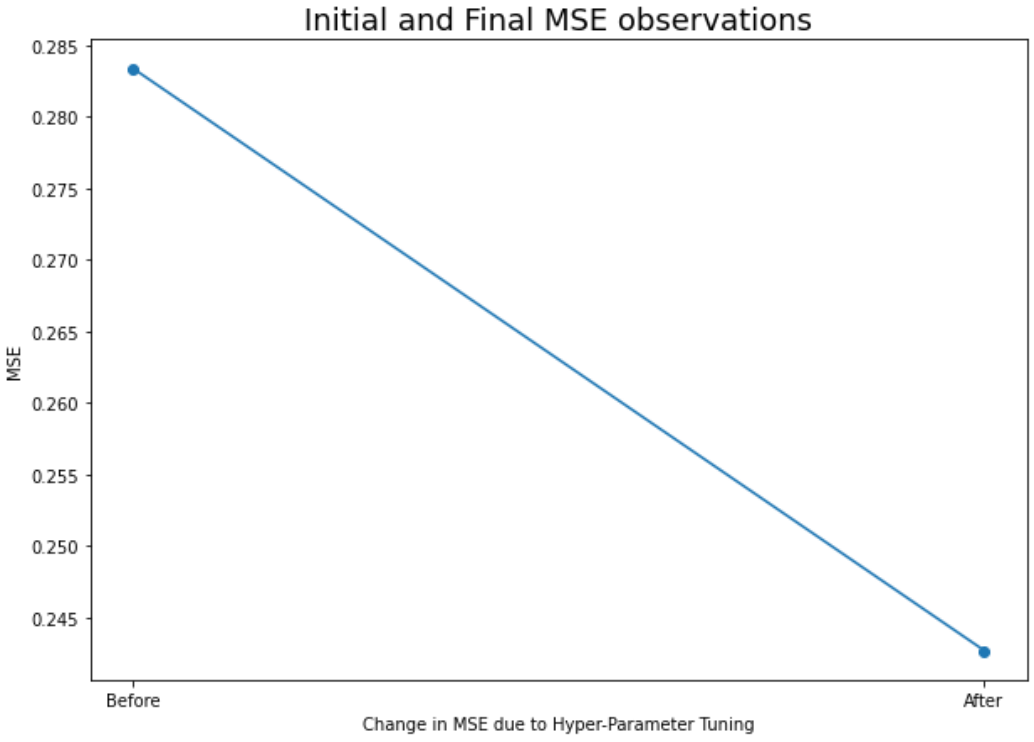
The corresponding changes in MSE on changing the above hyper-parameters are described in the below plot




MSE variations during Splitter Tuning
MSE
0.36
0.34
0.32
0.30
0.28
best
random
Splitter


MSE variations during Max Leaf Nodes Tuning
MSE
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
10
20
30
40
50
60
70
80
90
Max Leaf Nodes





After tuning all the hyper-parameters, we got the following MSE value on the validation set.
MSE = 0.2427. (it may vary when the code is run again)

So, here are the observations:

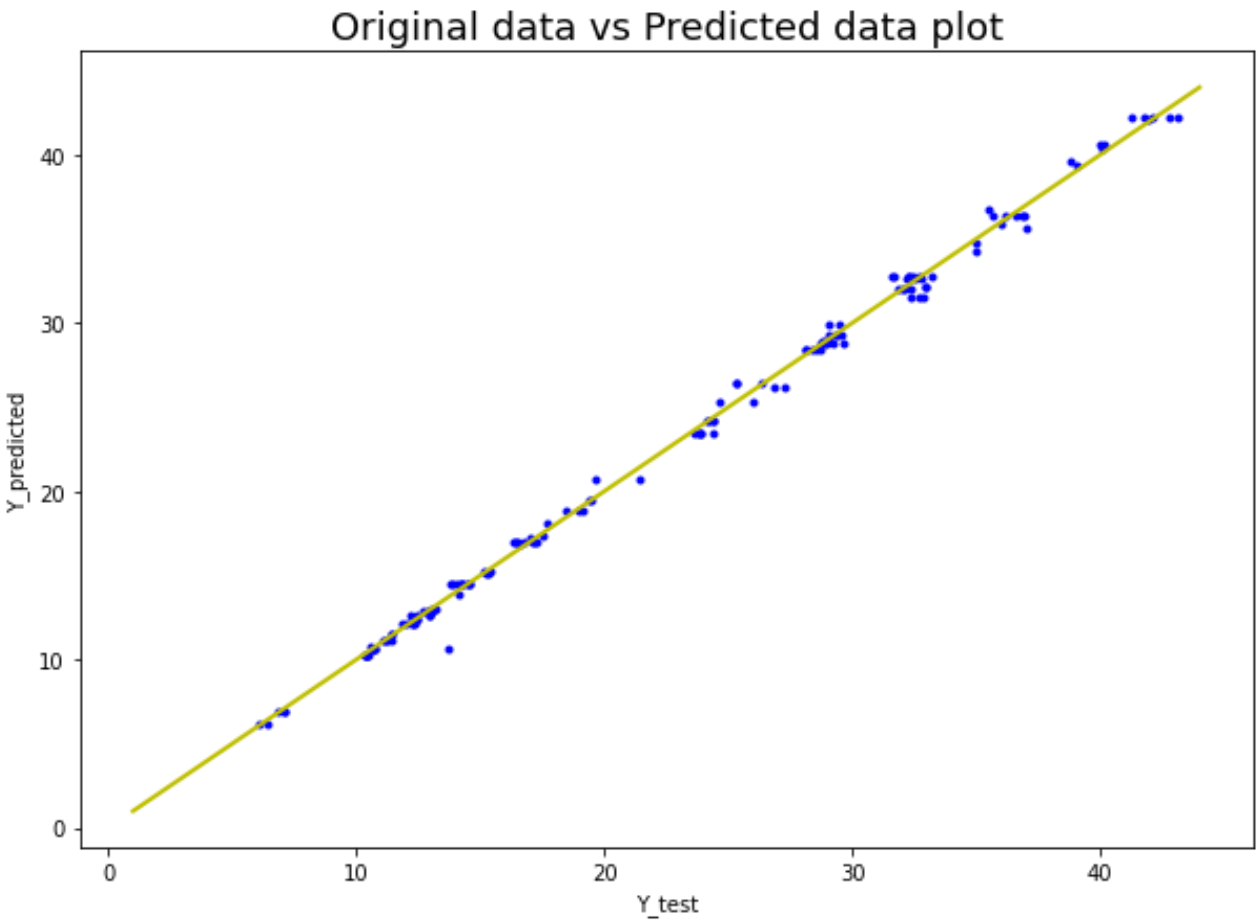
MSE before tuning	=	0.2834
MSE after tuning	=	0.2427

3. Performing 5-fold Cross Validation :

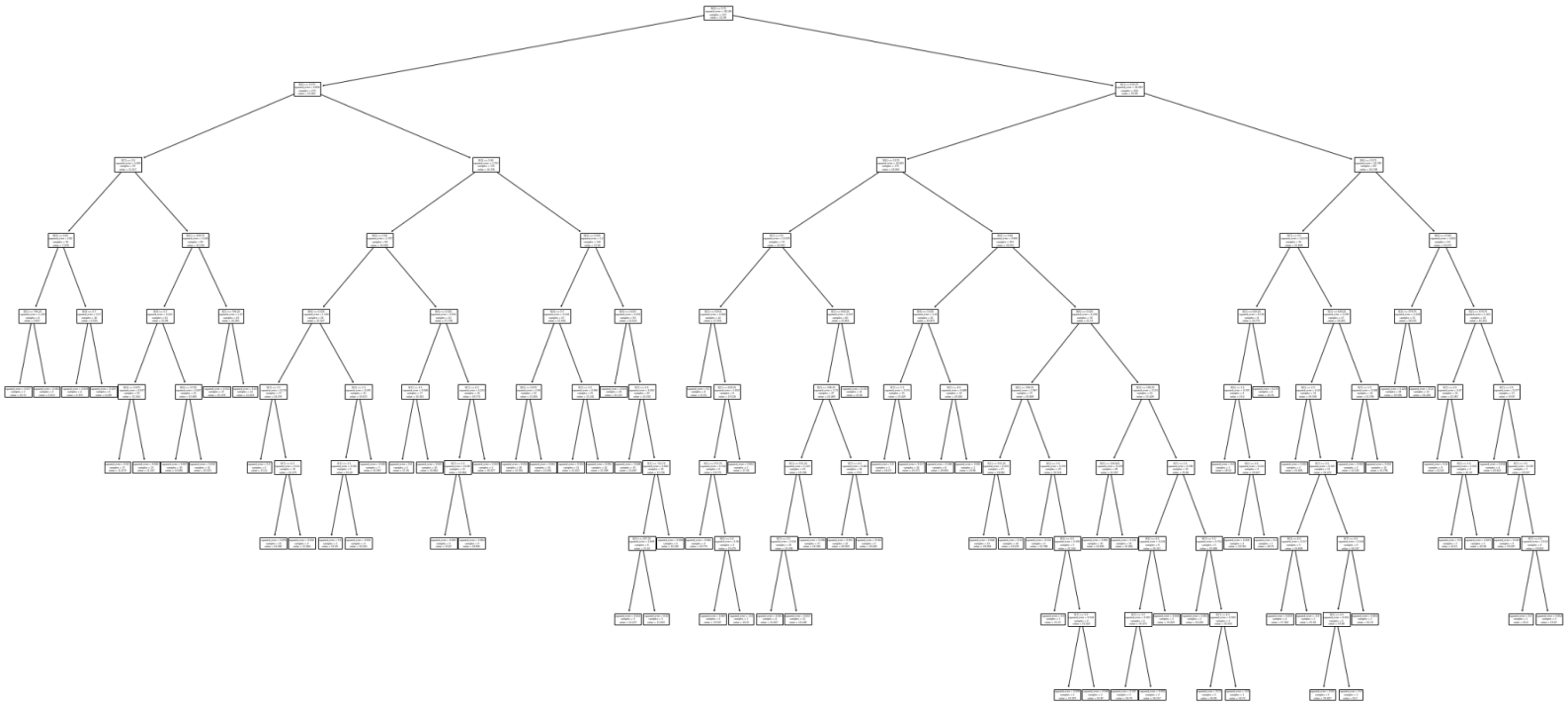
- After getting the optimal hyper-parameters, we build an optimal Decision Tree Regressor using those parameters.
- We apply 5-fold cross validation on the training dataset and observe the scores. We also calculate the average MSE score for the same.
- After that the built optimum model tested on the testing dataset and MSE is calculated.
- These are the observations for MSEs :

```
5-fold cross validation Mean Square Errors :
[0.40475494 0.32017093 0.40428392 0.29759929 0.48563558]

Average MSE for 5-fold cross validation : 0.3825

Final MSE for predictions on testing data : 0.2672
```

Also, a plot of original testing data vs predicted testing data is created for visuals.



Following is the Decision Tree Regressor plot of the trained optimal model.



End of the Report !