Pattern Recognition and Machine Learning

Lab - 2B Assignment Report

Aryan Himmatlal Prajapati (B21EE012)

Question 1.

Part 1

Pre-Processing of Dataset:

1. Checked for Null Values in Every Column of Dataset. Did not find any null values in the dataset.
2. Checked for the categorical/object type data. Again did not find any categorical/object type data.
3. Normalized dataset into range of (0,1) using Maximum-Absolute Scaling.
   Maximum Absolute Scaling:
   Maximum absolute scaling **scales the data to its maximum value**; that is, it divides every observation by the maximum value of the variable: The result of the preceding transformation is a distribution in which the values vary approximately within the range of -1 to 1.
4. Separated Features and labels.
5. Splitted Data into Training,Validation and Testing Set using a 70:10:20 ratio, which represents training:validation:testing.

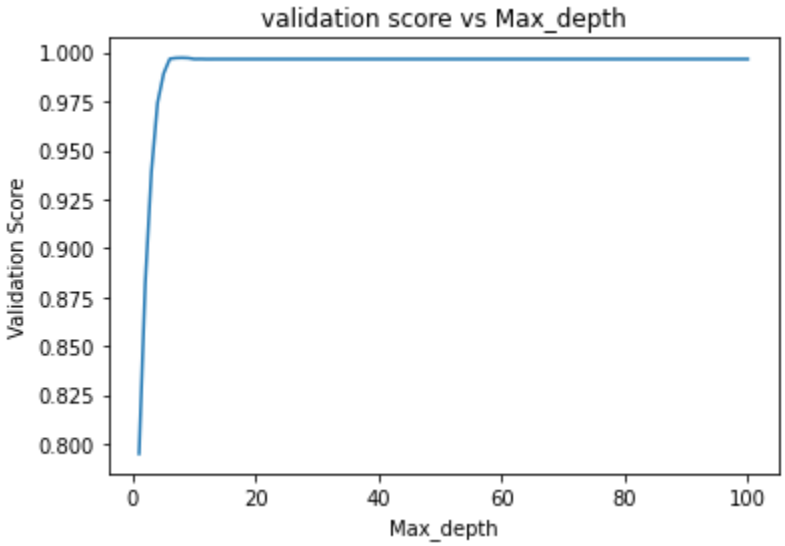
Part 2

Hyper Parameter Tuning:

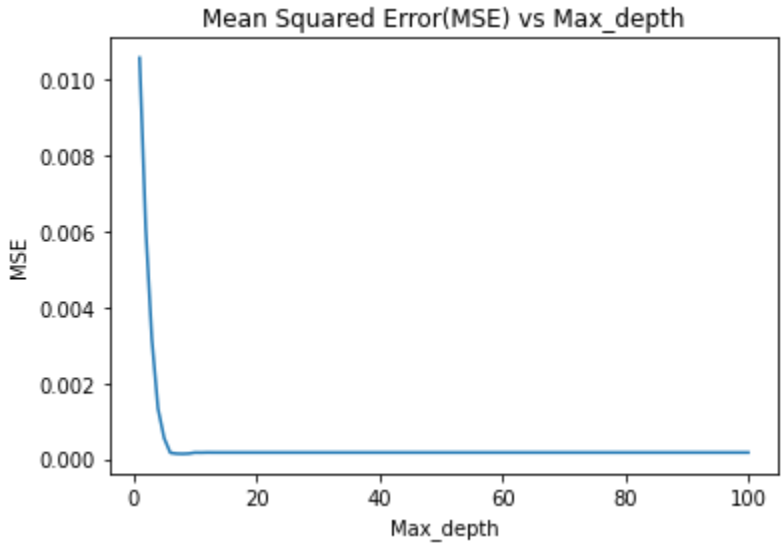
1. Max_depth:
   The maximum depth for growing each tree: an integer between 1 and 100, inclusive.
   Effect:
   In general, the deeper you allow your tree to grow, the more complex your model will become because you will have more splits and it captures more information about the data.

Tuning:
Trained Decision Tree Regressor with Values of Max_Depth varying from 1 to 100. Collected the accuracy score(r2 score) and Mean Squared Error(MSE) for models trained at Each value of Max_depth. Plotted Curve between Mean Squared Error vs Max_Depth and Accuracy Score vs Max_Depth.

Accuracy Score vs Max_Depth



Mean Squared Error(MSE) vs Max_Depth



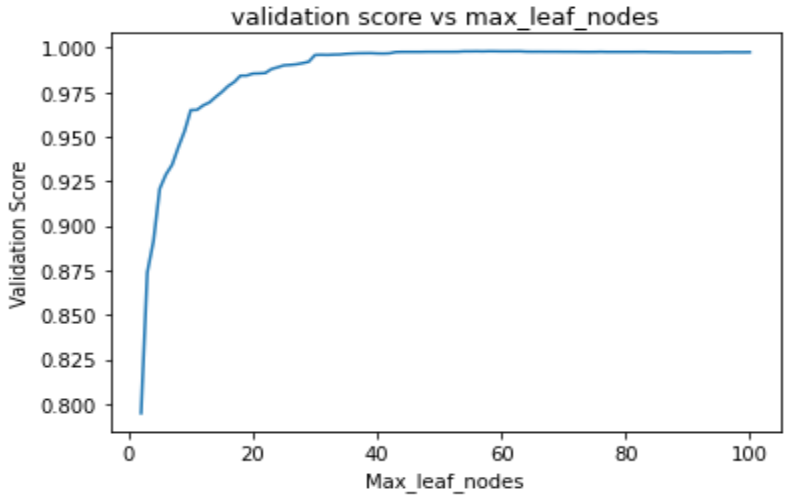
From the obtained results,

```
Validation score is maximum, when max_depth = 8.0
Mean Squared Error is minimum, when max_depth = 8.0
```

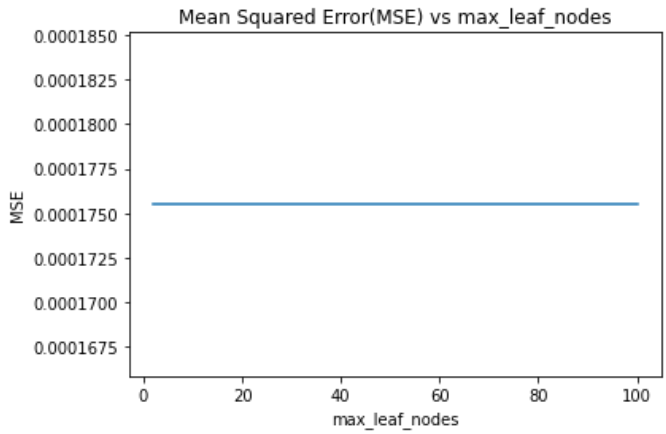
Optimum Max_Depth = 8

2. Max leaf nodes:
   Maximum number of leaf nodes a decision tree can have.
   Effect:
   Increasing no. Max_leaf_nodes increases the accuracy of the model. Not limiting the growth of a decision tree may lead to overfitting.
   Tuning:
   Trained Decision Tree Regressor with Values of Max_leaf_nodes varying from 1 to 100. Collected the accuracy score(r2 score) and Mean Squared Error(MSE) for models trained at Each value of Max_leaf_nodes. Plotted Curve between Mean Squared Error vs Max_leaf_nodes and Accuracy Score vs Max_leaf_nodes.

Accuracy Score vs Max_leaf_nodes



Mean Squared Error(MSE) vs Max_leaf_nodes



Part 3

Hold-out Cross Validation:

```
mean squared error: 0.00013865018770035017
  Accuracy Score(r2): 0.9976515080764548
```

5-fold cross-validation:

```
mean squared errors: [0.00675959 0.00022262 0.00012667 0.00020008
                      0.0003589 ]
        average mean squared error: 0.0015335728784766091

accuracy scores: [0.84421734 0.99547333 0.99762545 0.99665116 0.99366038]
            average Accuracy score: 0.9655255297806924
```

repeated-5-fold validation:

```
mean squared errors: [1.53771666e-04 9.41602036e-05 1.41014484e-04
                      1.21345082e-04
    9.71499673e-05 1.04223826e-04 1.37274420e-04 1.04334862e-04
    1.13616911e-04 1.68068564e-04 1.38599199e-04 1.03870031e-04
    1.06022490e-04 1.27058452e-04 1.04555977e-04 1.06602126e-04
    1.04408599e-04 1.16032843e-04 1.11309621e-04 1.39366491e-04
    1.30074014e-04 1.41951264e-04 1.14893992e-04 1.75741939e-04
                      1.39139154e-04]
        average mean squared error: 0.00012378344717534645

accuracy scores: [0.99741323 0.99807012 0.99742432 0.9977345  0.99822153
                      0.99821833
 0.99749597 0.99807314 0.99774484 0.99691856 0.9975405  0.9982086
 0.99808446 0.99749629 0.99799592 0.99800006 0.99826376 0.99771738
 0.99808817 0.99726755 0.99742392 0.99710187 0.99783456 0.99690983
                      0.99777503]
            average Accuracy score: 0.9977208979409312
```

Part 4

Methods to avoid overfitting of Decision Trees:

Pre-Pruning:

The pre-pruning technique refers to the early stopping of the growth of the decision tree. The pre-pruning technique involves tuning the hyperparameters of the decision tree model prior to the training pipeline. The hyperparameters of the decision tree including max_depth, min_samples_leaf, min_samples_split can be tuned to early stop the growth of the tree and prevent the model from overfitting.

Post-Pruning:

The Post-pruning technique allows the decision tree model to grow to its full depth, then removes the tree branches to prevent the model from overfitting. Cost complexity pruning (ccp) is one type of post-pruning technique. In case of cost complexity pruning, the ccp_alpha can be tuned to get the best fit model.

Mean Squared Error after Pre-Pruning of Decision Tree: 0.013342087822081414

Mean Squared Error after Post-Pruning of Decision Tree: 0.0006425421827856934

Conclusion:

In This Case, Post-Pruning of Decision Tree Performed better than Pre-Pruning of Decision Tree(as MSE(Post-Pruning)"<"MSE(Pre-Pruning)) Because In Post-Pruning we plot decision tree and after analyzing that we do hyper-parameter tuning but in case of Pre-Pruning we set parameters before plotting of Decision Tree.

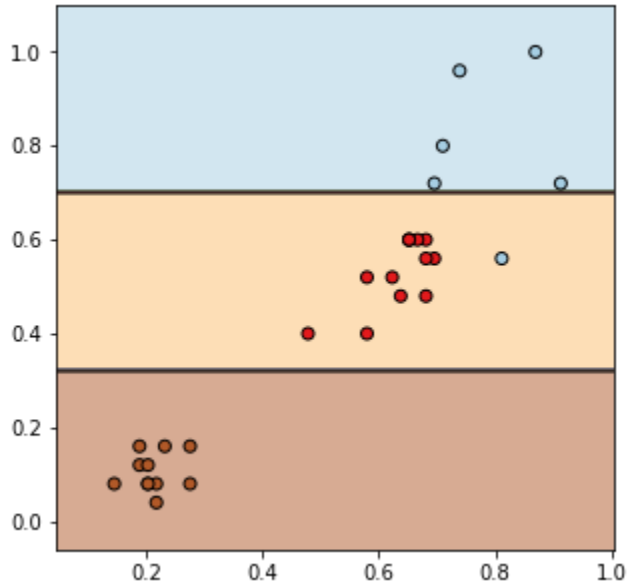
Question 2.

Classification

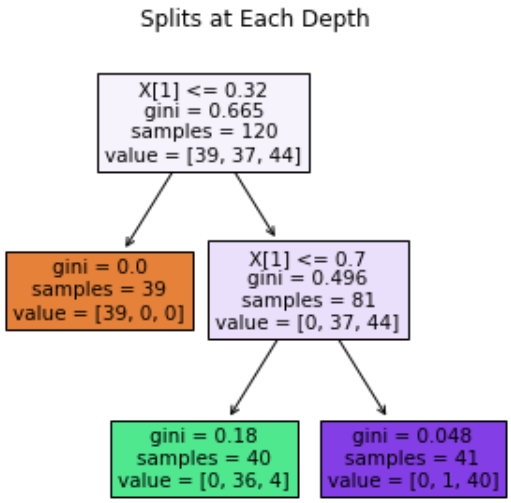
Part 1

Trained Decision Tree Classifier with max_depth = 2 on preprocessed dataset and Plotted Decision Boundary as well as indicated the depth at which each split was made.

Obtained Decision Boundary:



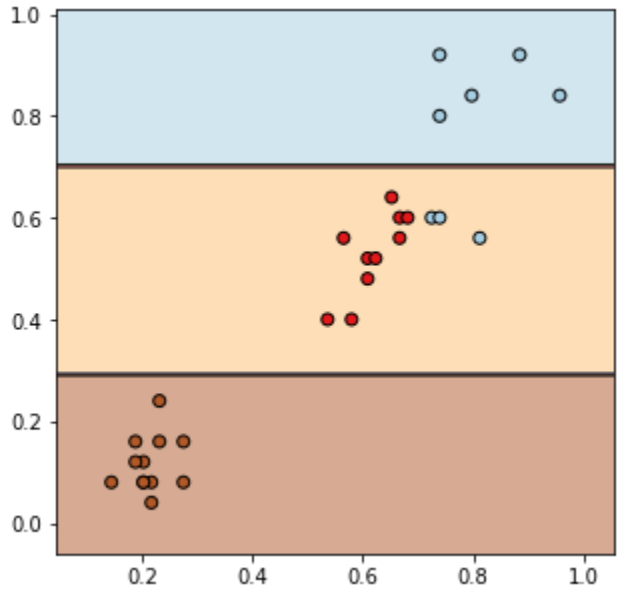
Split at each depth:



Part 2

Remove the widest Iris-Versicolor from the iris training set (the one with petals 4.8 cm long and 1.8 cm wide) and trained a new Decision Tree Classifier with max_depth = 2 and Plotted Decision Boundary.

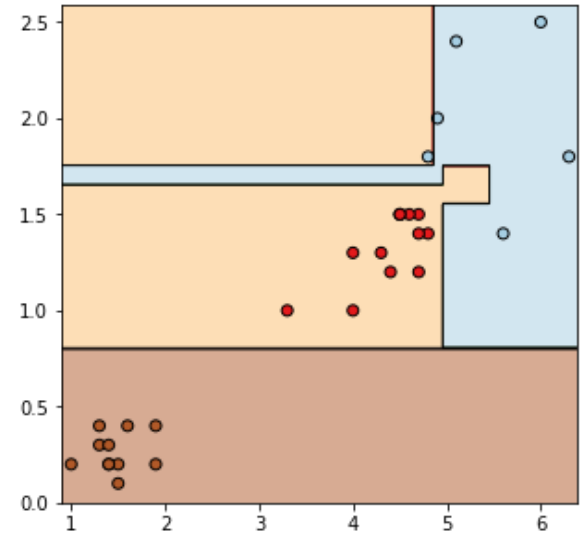
Obtained Decision Boundary:



Part 3

Trained a Decision Tree Classifier with (max-depth = None) on the preprocessed dataset. Plot the Decision boundary for the same.

Obtained Decision Boundary:

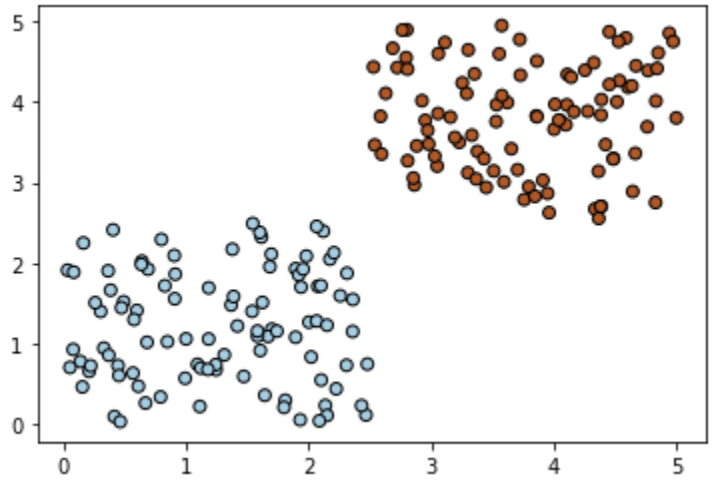


Comparison:
In part 1 , decision boundaries are straight horizontal lines but this is not the case with part 3. In part 3, the decision boundary is more towards overfitting which is pretty obvious because In part 1 max_depth is limited to 2 which makes it less complex compared to part 3 where max_depth = None(nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples).

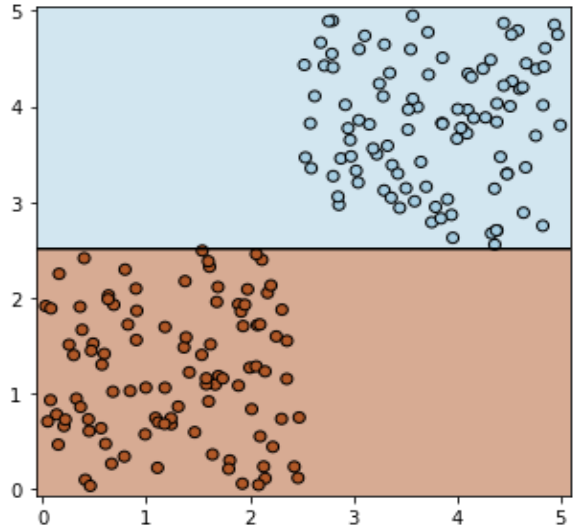
Part 4

Created a random dataset having 2 attributes(X1 and X2), and 2 classes (y=0 and y=1). X1,X2 are randomly sampled from the range (0,5). y=0 when X1<2.5, and y=1 when X1>2.5. The dataset has 100 data points for both the classes. Trained a decision tree for such a dataset(max-depth=2). Plotted the obtained decision boundaries.

Distribution of Data Points:

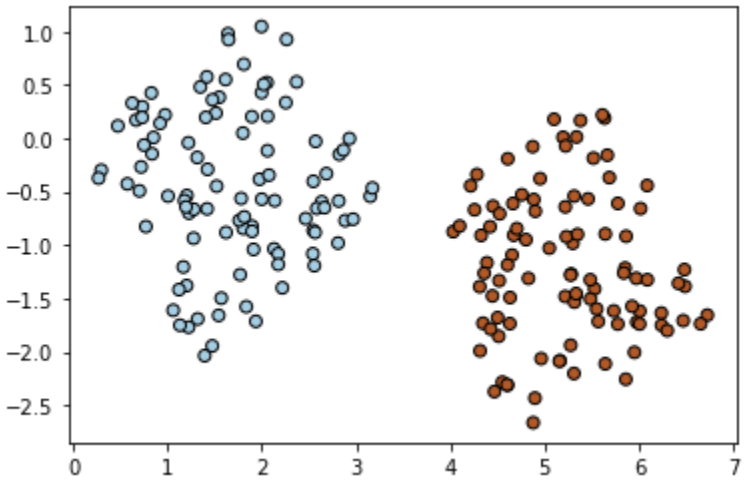


Decision Boundary:

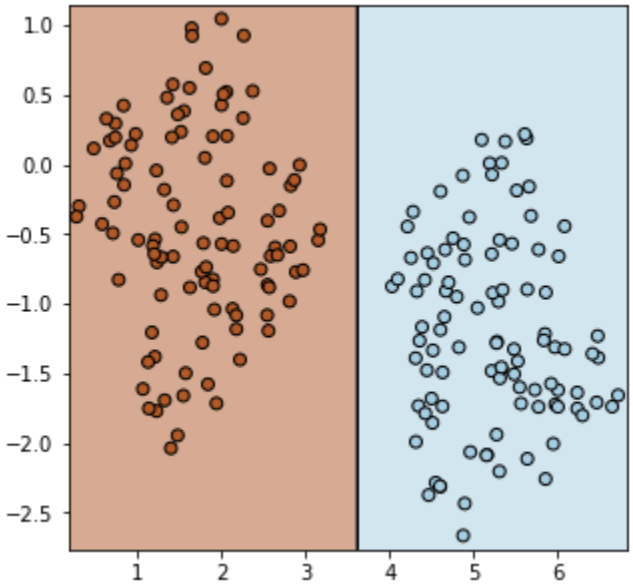


Rotated the data points by 45 degrees in clockwise direction about the origin (X1=0,X2=0). Trained another decision tree classifier using sklearn.

Distribution of Data Points:



Decision Boundary:



Comparison:
Before rotating the data points, the decision boundary was a horizontal straight line which is clear from the distribution of the dataset. After rotating the data points by 45 degrees in clockwise direction, the decision boundary becomes a vertical straight line because we can see from the distribution that data points are rotated by 45 degrees in clockwise direction. The obtained decision boundary in both the cases is straight line because data points are linearly separable.

Part 5

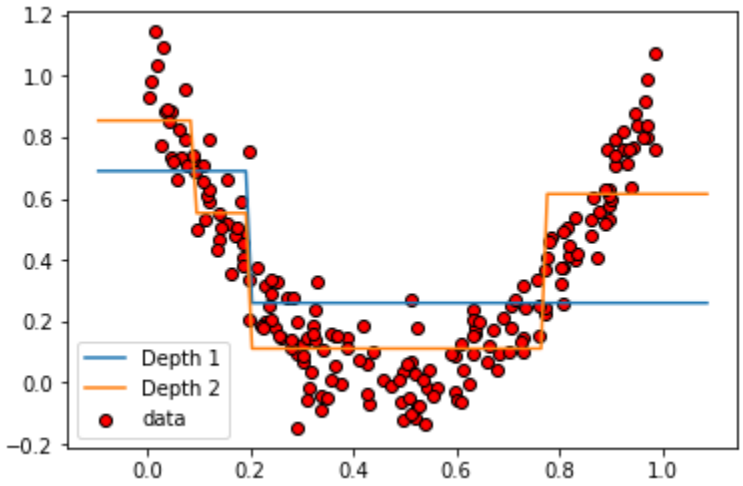
In part 2, The obtained decision boundaries are horizontal straight lines because max_depth is limited to 2 which makes it a less complex model, while in part 3 the obtained decision boundaries are not completely linear but overfitted because the tree is grown until every leaf node is pure. Again in Part 4 obtained decision boundaries are straight lines because data points are linearly separable which is also the case in part 2.

Regression

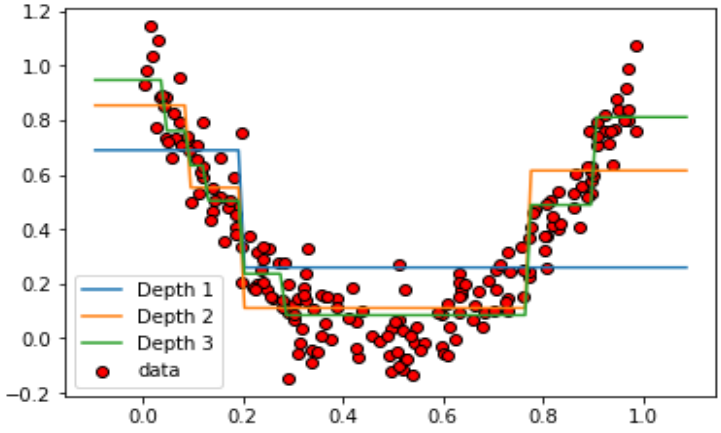
Part 1

Trained two decision tree models, one with max_depth = 2 and another with max_depth = 3. Plotted the regression predictions at each depth for each max_depth(for e.g., at depths 0,1 for max_depth = 2) using a line plot. made a scatter plot of the data points on the same plots.

Regression Prediction at Each Depth for Max_Depth=2:



Regression Prediction at Each Depth for Max_Depth=3:



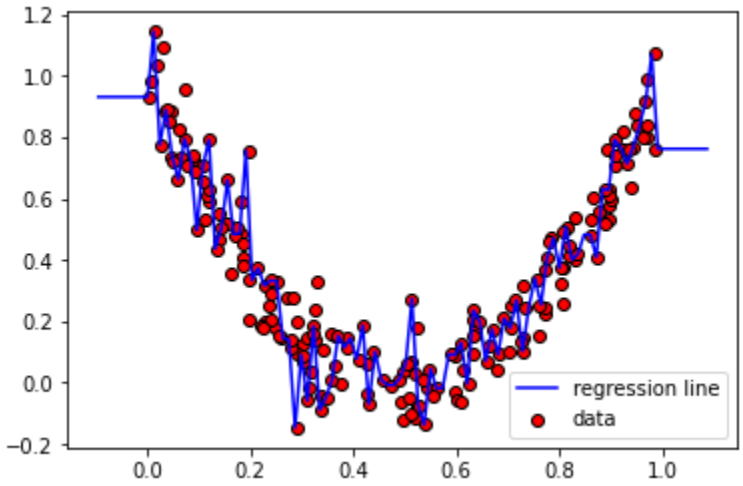
Analysis:
Here from the obtained plots we can clearly see that as the depth increases the regression predictions are getting better fit with data points.

Part 2

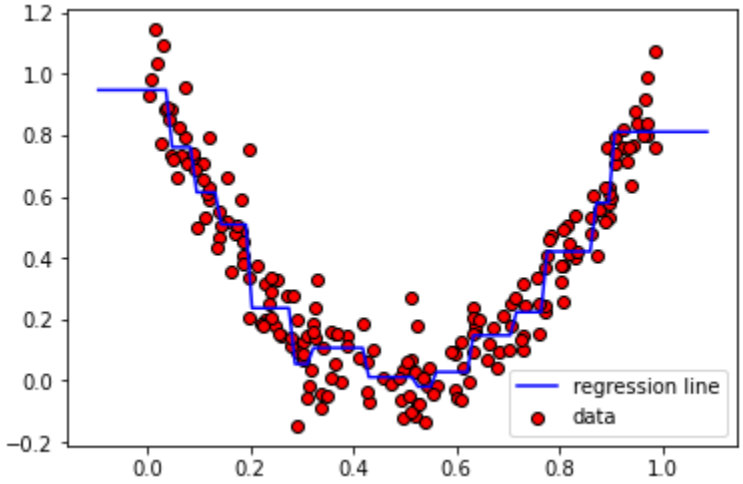
min_samples_leaf is The minimum number of samples required to be at a leaf node.

Plotted the data points and made a line graph to show the decision tree fits on the dataset in two cases: min_samples_leaf = 0 and min_samples_leaf = 10.

When Min_samples_leaf = 0:



When Min_samples_leaf = 10:



Analysis:
When Min_samples_leaf = 0, the regression line seems overfitted and when min_samples_lead = 10 the regression line seems not so good fit of data points because increasing the number of min_samples_leaf leads to underfitting.

Question 3.

Preprocessed the dataset. Implemented decision tree using Node class and DecisionTree class. The Implemented Decision tree contains methods such as fit, predict etc. incorporating all the task from part 2 to part 7(ie. Entropy calculation, Information gain, deciding best split features and thresholds, deciding leaf node value, Accuracy etc.)

Created a class Node having attributes like right, left, threshold, feature and values. It indicates a node of a decision tree. Class Node has a helper function `is_leaf_node` which helps in deciding whether the given node is leaf node or not.

Created a class Decision tree which is initialized with attributes max_depth, n_features. It indicates the decision tree. It has a method fit which makes a decision tree from a dataset using some helper functions. In the helper function _build_tree there is a stopping condition that when the decision tree must stop growing. It returns nodes with the best feature and best threshold value. Helper function best_split return splitting index and threshold value determining the best information gain. Helper function information_gain returns information gain using parent and child entropy. Helper function split returns left and right indexes by separating them using threshold value. Helper function _entropy calculates entropy and returns it. Helper function most_common_value returns the most common label to determine leaf node value.

Method predicts returns predicted labels by traversing the decision tree and traversal of the decision tree is done using the helper function traverse_tree.

```
Overall Accuracy: 0.9850746268656716
```

Thank You