Pattern Recognition and Machine Learning
Lab 2 Assignment
Decision Tree Regressor and Classifier

Jatin Lohar
B21CS091

Question 1

Part 1

I imported **‘numpy’, ‘pandas’** and **‘test_train_split’** to do the question. The read the csv file using **‘pd.read_csv( )’**. Then printed to 5 entries.

I then split the dataset into X and Y using **‘drop’**. Then split X and Y into **‘X_train’, ‘Y_train’, ‘X_test’, ‘Y_test’, ‘X_val’, ‘Y_val’** using **‘test_train_split’** in the ratio 70:20:10.

Part 2

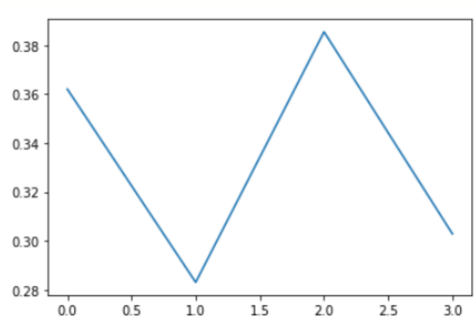
I trained 4 different Decision Trees by different hyperparametes…

- DecisionTreeRegressor(random_state=1)
  Mean squared error = 0.3620592105263157

- DecisionTreeRegressor(criterion='squared_error', max_depth = 7, min_samples_split=4)
  Mean squared error = 0.2830112724214286

- DecisionTreeRegressor(criterion='absolute_error', min_samples_split = 4, max_depth = 7)
  Mean squared error = 0.38552368421052685

- DecisionTreeRegressor(criterion='squared_error', max_depth = 8, min_samples_split=4)
  Mean squared error = 0.30285302353875765



The Decision Tree in second case performed best.

Part 3

- Hold-out Validation → MSE = 0.40884854838709683

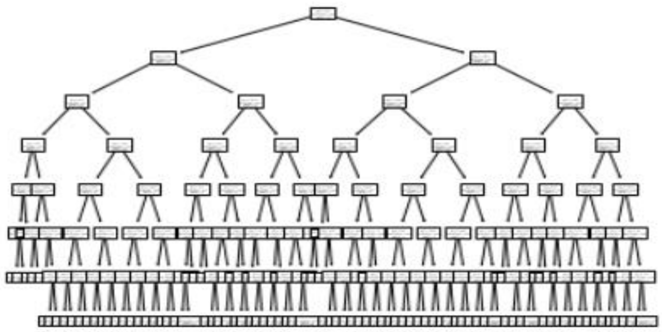
❖ 5-Fold Validation

```
5-Fold Cross-Validation without Shuffling =  2.93327659578983
5-Fold Cross-Validation with Shuffling = 0.3781359653255241
```

❖ RepeatedKFold Cross Validation – Used **‘n_splits = 5’** and **‘n_repeats = 5’**

```
MSE of Repeated KFold Cross Validation =  0.37849752699261535
```

❖ Decision Tree – plotted the Decision Tree using **plot_tree** from **‘sklearn.tree’**



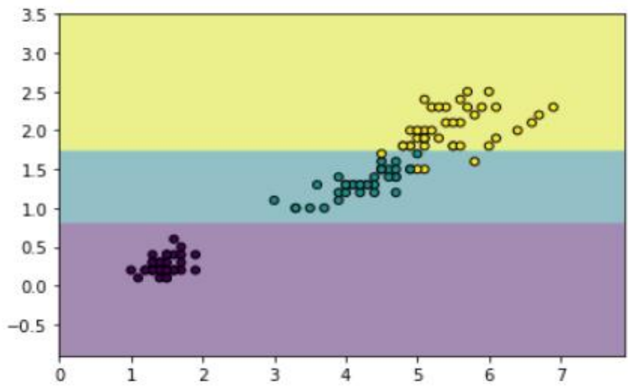
Question 2

Imported **numpy**, **pandas** and **train_test_split.** Then imported the dataset using **‘datasets’** from sklearn. And then ‘datasets.load_iris( )’. Then split it into X and Y. The shapes of X and Y matrix are:

```
(150, 2)
(150,)
```

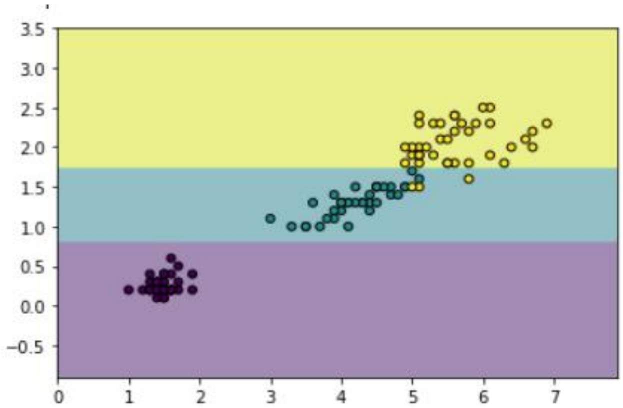
Part 1

Imported the DecisionTreeClassifer with max_depth = 2. And then fitted it on X_train and Y_train. Then used **‘DecisionBoundaryDisplay’** to plot the decision boundary.



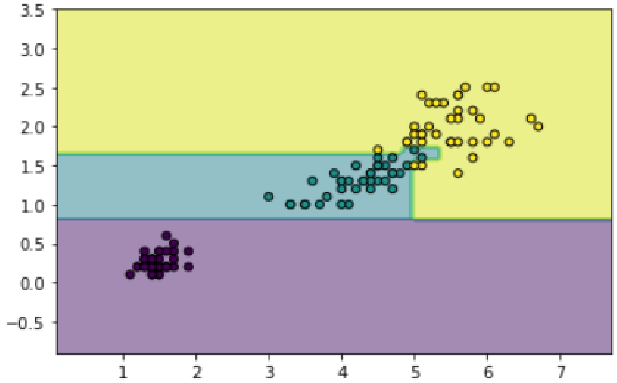
Part 2

Deleted the entries entries with petal_length = 4.8 cm and petal_width = 1.8 cm. Then train new DecisionTreeClassifier for it, and plotted the Decision Boundary.



Part 3

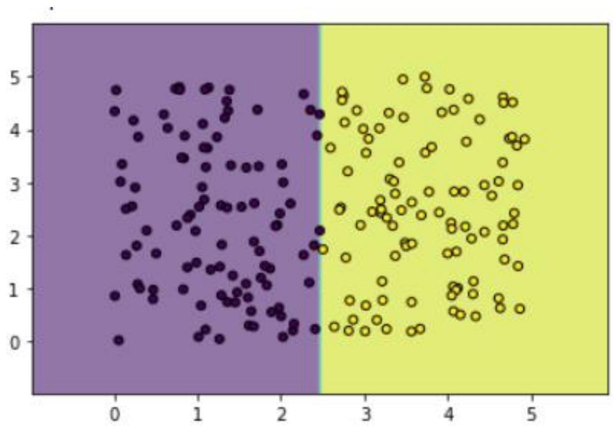
Making new Decision Tree Classifier with default entries and then plotting the Decision Boundary. We can see, that the classifier has overfitted every point.



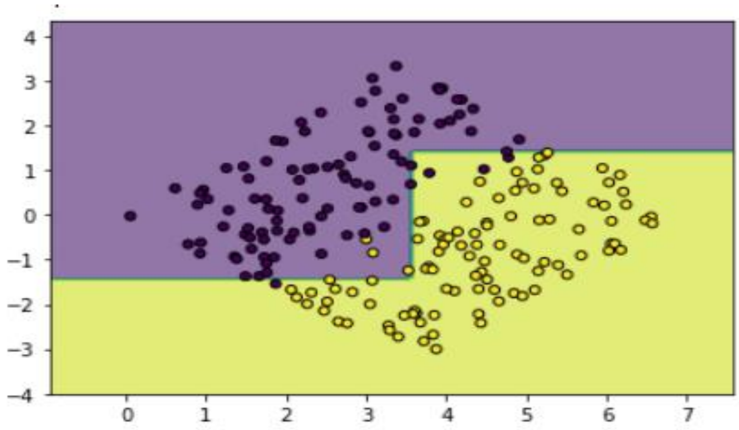
Part 4

Made a random dataset using random.uniform. I made two datasets. One with X_1 from **0 to 2.5** and other from **2.5 to 5**. So as to ensure equal distribution.

The concatenated both the datasets and shuffled it. Then, I made the Y dataset using X_1. The decision boundary of the points is…



Using mathematical formula for rotation of point about origin by 45^0. I got the new points. Again plotted the decision boundary for it with max_depth = 2. The decision boundary I got is…



Part 5

The conclusion I can make are…

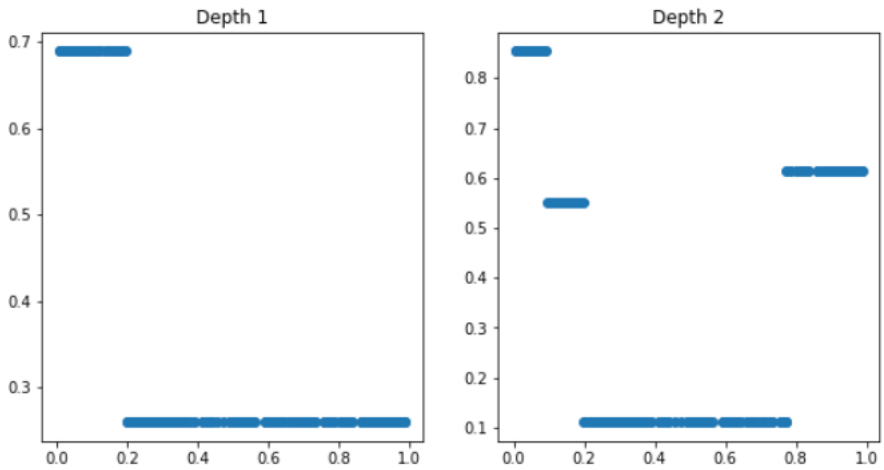
- ❖ The decision boundary of the Decision Trees and either horizontal or vertical.
- ❖ It tends to overfit in case the max_depth is not pre-defined.

Regression

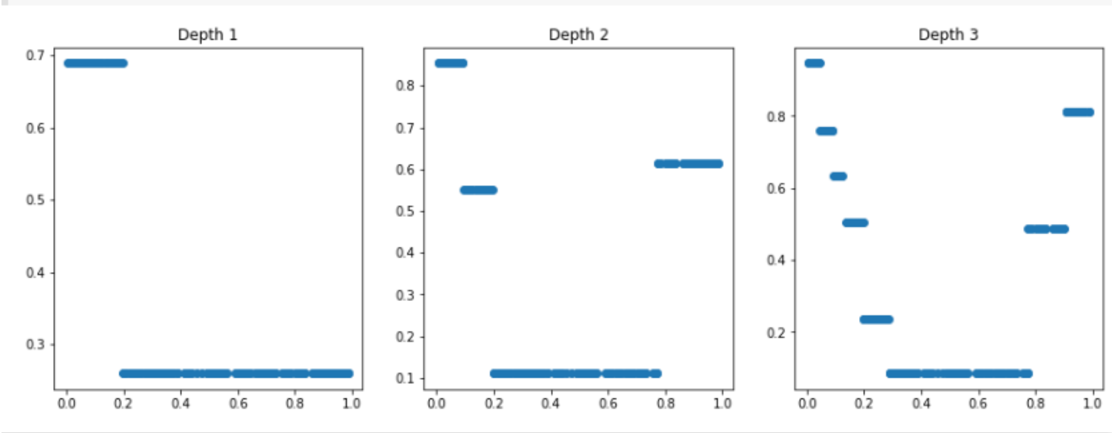
Part 1

Training model for different max_depth and ploting the plots.

For max_depth = 2

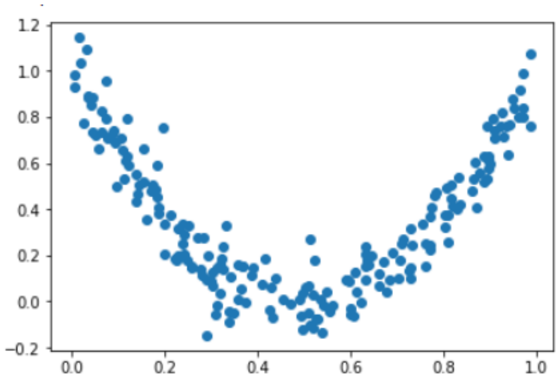


For max_depth = 3

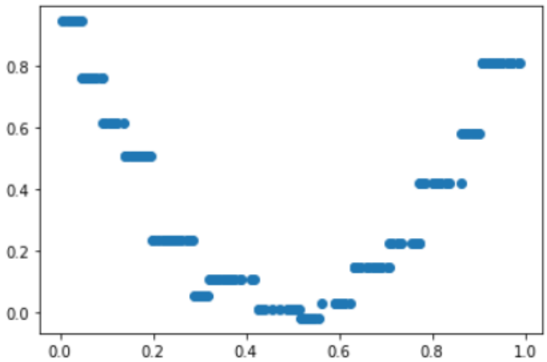


Part 2

For min_samples_leaf = 1, the Decision Tree overfits the dataset.



For min_samples_leaf = 10, the Decision Tree generalises the dataset very well.



Question 3

Part 1

I imported the data using online GitHub link, then kept the data in variable **'data'**. Then checked for null values and found the following classes has null values

```
species              0
island               0
bill_length_mm       2
bill_depth_mm        2
flipper_length_mm    2
body_mass_g          2
sex                 11
year                 0
dtype: int64
```

Then I dropped the values using **'dropna( )'.** Then I used **'LabelEncoder'** to encode the categorical features using fit_transform. I then split the function into X and Y respectively.

Part 2

I firstly calculated the unique values in X and Y. Then partitioned the data according to X_values. And found the probabilities of respective Y_values in that dataset. The by Gini index formula, I calculated the gini index for that group. Later, finding the gini index for entire column, I multiplied the gini index with the respective class probabilities. To find the final **'gini_index'**.

```
gini_index(X[:, 1], Y)

0.39703404111897544
```

Part 3

To make the function count_to_cat, I used the thinking that gini index is minimum if the randomness in the dataset is minimum. Hence it allotted the values of X according to the Y_values. Later, I converted all the continuous dataset into binary dataset.

Part 4

Made a function best_split( ), that finds the gini index of each column and then return the gini indexes found and the index with minimum gini index.

Part 5

Firstly made a TreeNode that will hold the values. The made a class DecisionTree with parameter as value, children, index. Then made the function train, which makes the decision tree using another function **fit**. Here **fit** function runs recursively until the max_depth has not reached to all the elements have the same class. I the chose the best split using the function made above and ran the fit function again with neglecting the function found here.

Part 6

In the **predict** function, I simply traversed the tree, according to the index of the TreeNode until a leaf node is been achieved. The printing value of the leaf node.

Part 7

Trained a new DecisionTree using X_train and Y_train. The made a list that carries the predicted values of the decision tree. Later found the accuracy by comparing the predicted value and original value and finding number of correct prediction.

```
Correctly predicted elements  [20. 11. 22.]
Original number of elements  [32. 13. 22.]

Overall Accuracy of the Model is 0.7910447761194029:
Classwise Accuracy of the Model is 0.8237179487179488:
```