CSL2050 PATTERN RECOGNITION AND MACHINE LEARNING

LAB REPORT-02

NAME:	SAMEER SHARMA
ROLL NUMBER:	B21CS066
LAB TITLE:	DECISION TREES

Q1)

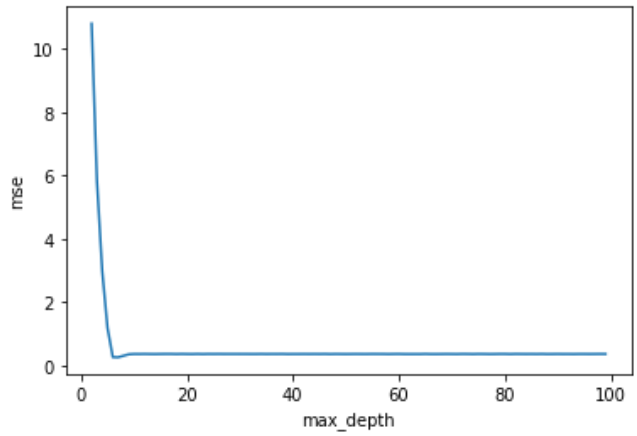
Part 1)

Basic Pre processing of data is done in this part. Dropped the rows having null values so that the “Not a Number” error does not occur while further process. Typecasted the columns into the required data types. Then, Normalized the data so that one feature does not dominate over other features. In the last, splitted the dataset into training, validation and testing datasets in the ratio 70:10:20.

Part 2)

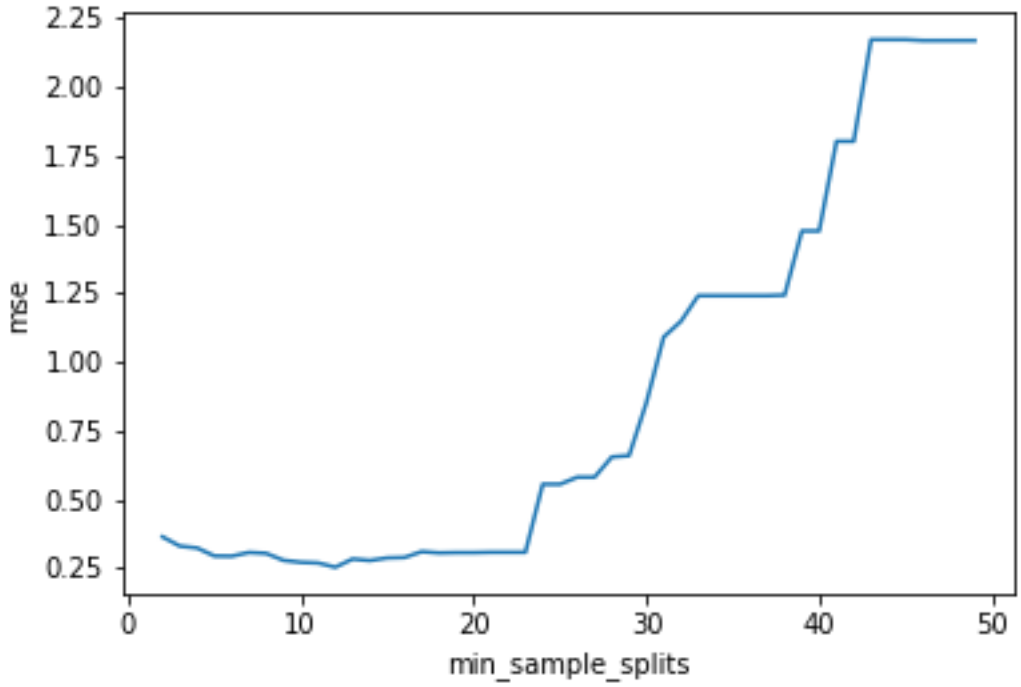
Max depth

The hyper-parameter Max-depth defines the maximum depth of our decision tree that it can reach. If its value is very less, our tree would be short and may underfit the training dataset. If its value if very high, tree would be very long and may overfit the training dataset and would not perform good on testing dataset.



Min Sample Split

It determines the minimum number of samples a node should contain so that it can split further. Its high value may cause underfitting since the impurity of the leaf nodes can be high. Its low value may cause overfitting on training dataset if training dataset is not representing each class equally.



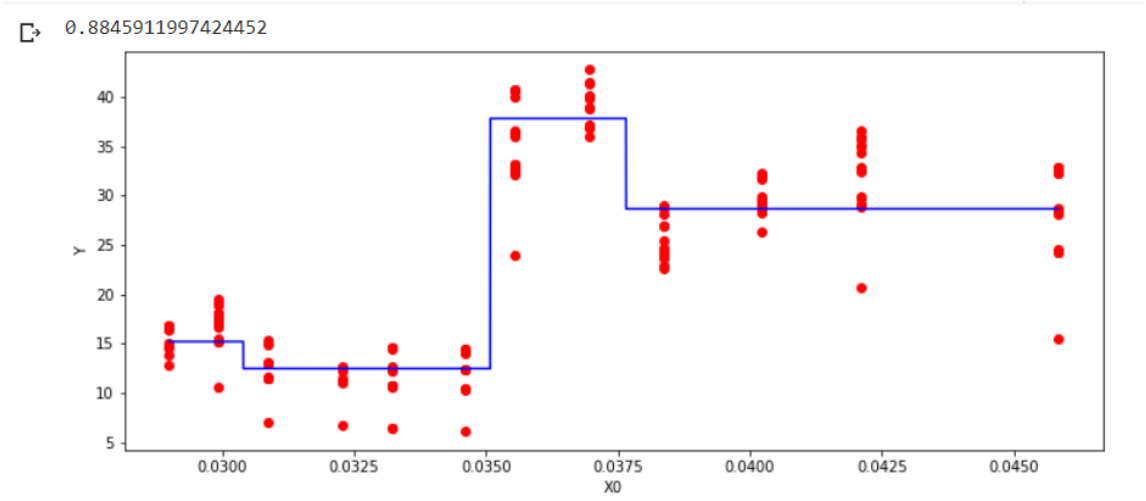
Part 3)

In this part different types of cross validations are done on the model to ensure that it is not underfit as well as not overfit. The different types of cross validations applied are as follows – hold out cross validation, k-fold validation and repeated k-fold validation. In the last, mse calculation for the testing data is done.

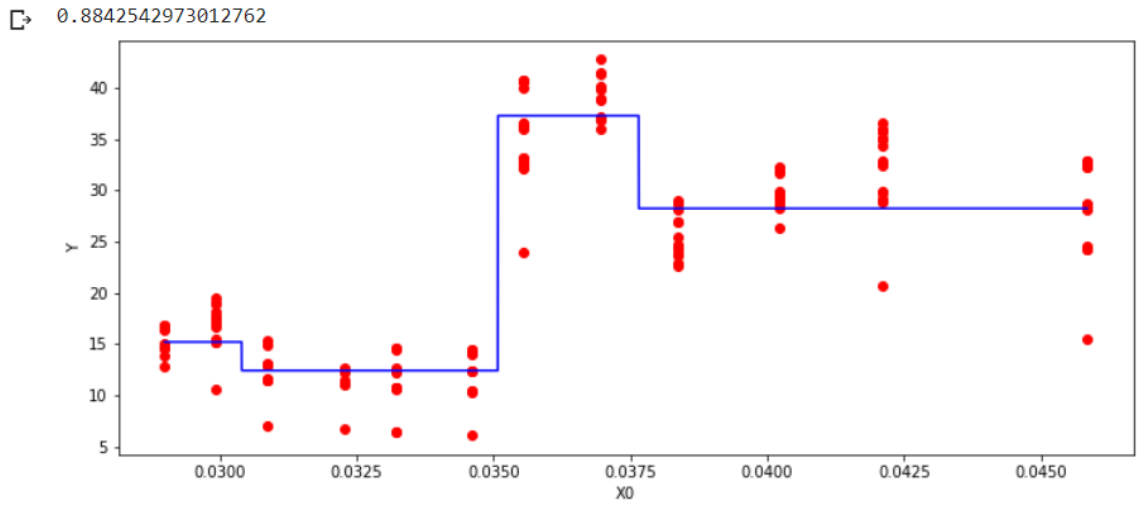
Part 4)

Here I have plotted the decision boundary by considering only one feature (because it is not possible to plot for more than one feature for a decision tree regressor) with max depth = 2. As one can see that the two graphs are not much different and accuracies are also nearly equal. This is because at the time of preprocessing the data we have normalized the data. If it was not normalized it may happen absolute error or L1 norm perform better since squared error does

not perform better when we have outliers in our data (data with high magnitude compared to remaining dataset)



L1



L2

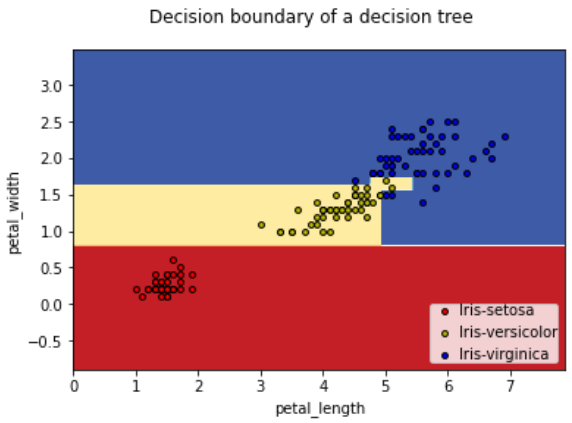
Q 2) Classification

Part 1)

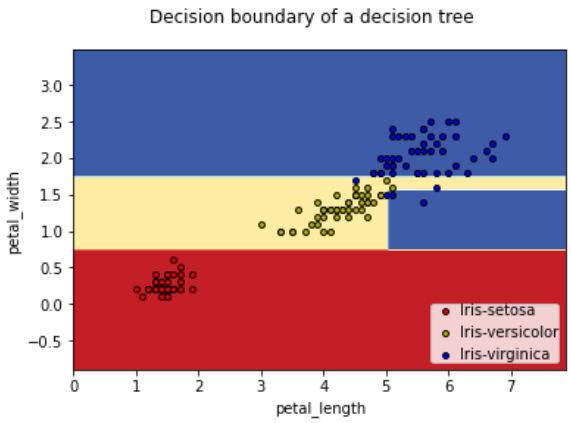
Basic Preprocessing on iris data is done in this part. A decision tree classifier is modeled using the sklearn library and decision boundary is plotted using the counterf function.

Part 2)

On removing the widest Iris-Versicolor (petal length = 4.8cm, petal width = 1.8cm), the decision boundary got modified and we got an extra rectangular elongation for Iris-Versicolor (yellow) in the region (5->), (1.5-1.8). Here both graphs has max depth = none.



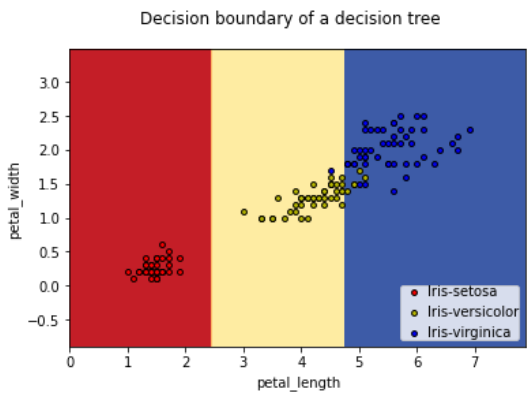
Without removing the

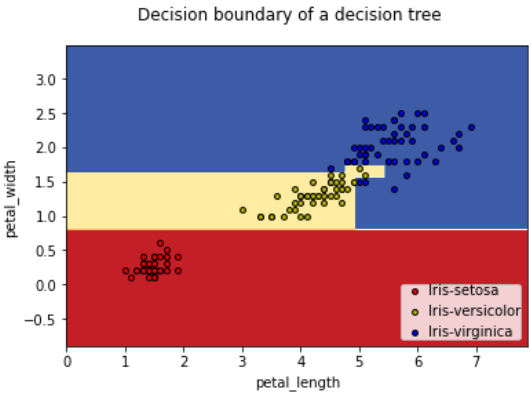


After removing the sample

Part 3)

On removing the constraints on max depth of the tree, the depth of the tree increased as we can see these step increases in the graph. This model is giving 1.0 accuracy on the training dataset but may not perform good on test data.



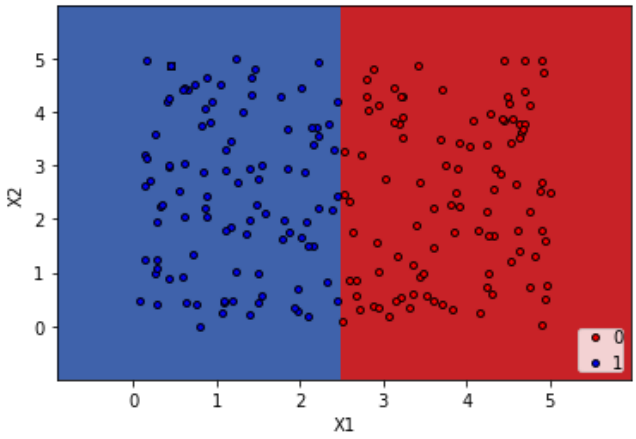


Max depth = 2 Max depth = None

Part 4)

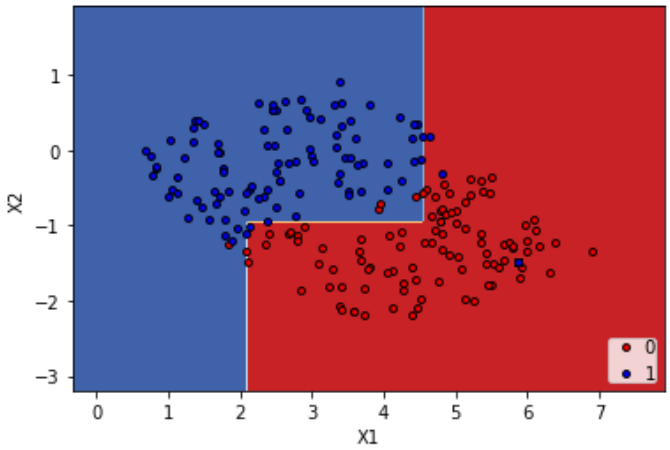
Before Rotation

200 random samples are uniformly generated in the range (0,5) and thus for nearly half samples, value of X1 is greater than 2.5 and for others is less than 2.5. Since both the classes are equally likely and data points are randomly distributed with nearly same variance, the decision boundary is x = 2.5. Also the other feature has no effect on the decision boundary.



After Rotation

After the rotating the data points by 45 degree angle clockwise, the second feature also comes into action and affects the decision boundary. The depth of the tree also increased since we can observe splits of depth = 0,1. Therefore, we get the decision boundary as shown in the figure.



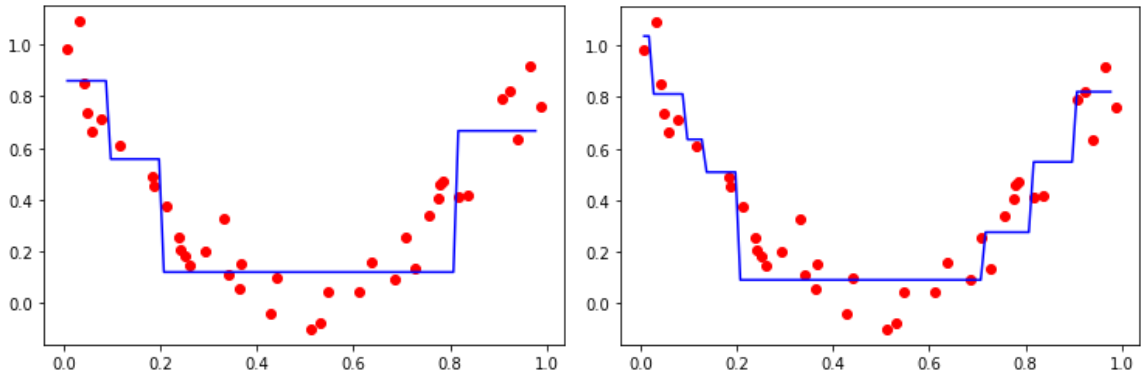
Part 5)

All the points have been discussed in the above three parts.

Q2) Regression

Part 1)

We can observe that the model with max depth = 3 is fitting more perfectly on the data than one with max depth = 2. Therefore, the former will perform better on the data than the latter as we can also see by evaluating the accuracy in the colab file.

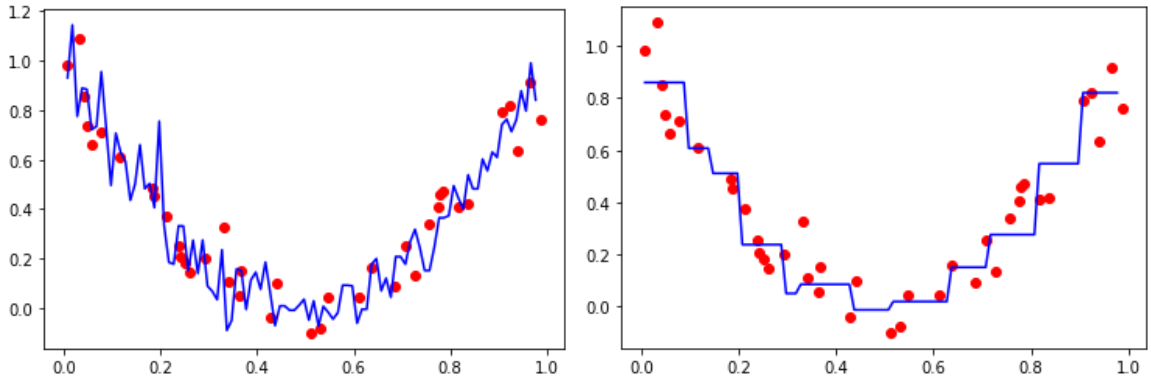


Max depth = 2 Max depth = 3

Part 2)

The one with min samples leaf = 1 has a very Zig Zag boundary which indicates it has a much higher depth which may lead to overfitting. It has such decision

boundary because the minimum number of samples on a leaf node is one (which is ofc the least) and thus increased the depth of the tree to a higher extent. Now in the case of one with min samples leaf = 10, the depth of the tree is lesser than the former one (since it has clear splits of depth). So it is not overfitting the data and will perform better than the former one.

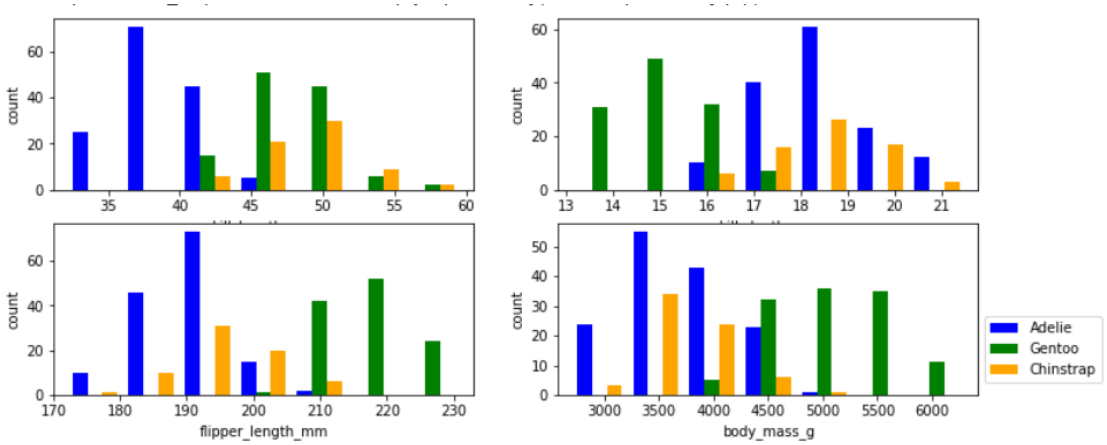


Min sample leaf = 1 Min sample leaf = 10

Q3)

Part 1)

In this part, preprocessing is done on data by dropping the null values and encoding the categorical data (sex, island etc). Data visualization of the penguin data is done by plotting bar graph for categorical features and histograms for continuous features.



Part 2)

Made a function to calculate the entropy of a given sample by using a helper function to calculate the probability of different classes in the sample. Formula used to calculate the entropy is as follows

$$Entropy(S) \equiv \sum_{i=1}^{n} -p_i \log_2 p_i$$

Using the entropy function I made a function to calculate the entropy for a feature.

$$\sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Part 3)

Cont_to_cat function is made to convert the continuous data into categorical data so that we can use the continuous data to train our decision train classifier. This is done by considering a sample from the feature column and splitting the feature column into two bins, one containing values less than the sample and other containing more than the sample. Then, Entropy of this split is calculated using the entropy function. Sample with least entropy is the best split and then encoded the feature column according to that sample (if <=sample then 0 else 1)

Part 4)

Function named find_best_split is made to find the best split feature. By considering each feature at a time, total entropy for the feature is calculated. One with the least entropy is the best split feature. Splitting of data and recursive calls to build the decision tree are done in the class Decision_tree_classifier_scratch. Data splitting is done on the basis of discrete values in the feature column (that's why we converted continuous data to categorical to perform splitting and calculate entropy)

Part 5)

In a conditional statement in the __grow_tree() function we are checking whether the information gain is not equal to zero and depth of the tree does not exceed the max_depth. After each level we are increasing the depth by 1 as you can see here

```
childs.append(self.__grow_tree(X[idx,:],y[idx],depth+1,available_feat))
```

If these conditions are not fulfilled, we are declaring a leaf node there with the most probable class of the parent as class decision. This function will also create a leaf node if there is no data in the split (empty condition) or we have no features left to continue the splitting.

Part 6)

Test function is made which takes test features as input on which prediction is to be made and outputs the predicted classes.

Part 7)

Made a function named accuracy to calculate the overall accuracy and class-wise accuracy of the decision tree. Calculations for accuracies are done as same as we did in our lab 1.

```
(0.97, [0.95, 1.0, 0.972972972972973])
```

Here, max_depth=2