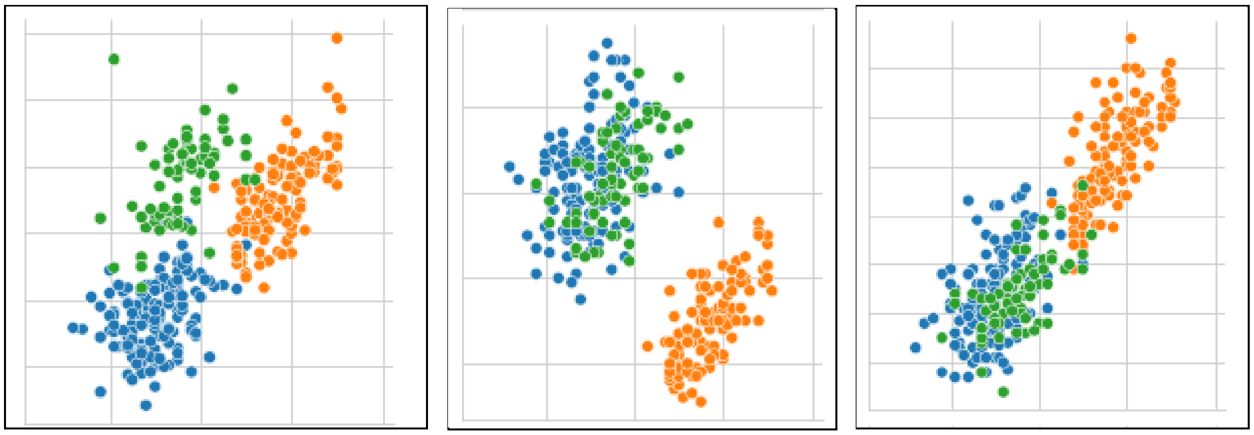
Pattern Recognition and Machine Learning
Report for : Lab 2

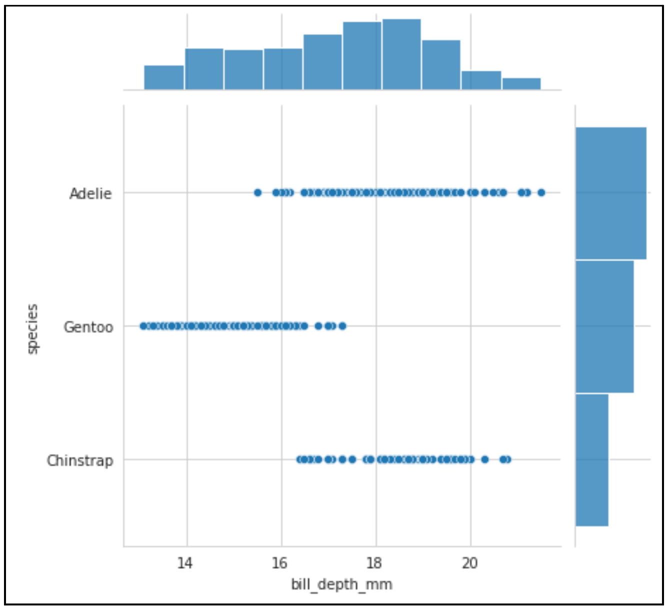
Name : Khushi Parikh
Roll No : B20EE029

Question 1:

- **(1.1A)** After importing the necessary libraries, we imported the data("penguins.csv"). We looked at the features of the data - data types, shape, head of pandas table and dropped the NaN values.
- **(1.1B)** I have plotted a pairplot of various features. If we look at the column of bill-length, we can see three **separate clusters** compared to other features and hence this is a good parameter to separate the data on.
  Diagrams - (bill_length, bill_depth, body_mass vs flipper_length respectively)



- Related plots which give distinct clusters are - (bill_length,bill_depth) (bill_length,flipper_length) (bill_length,body_mass)
- In the joint plots of each feature vs target variable(species), we see that we cannot completely differentiate using one feature. For example, bill_depth is high for both "Adilie and "Chinstrap" species.
- Our main aim is to **classify a group of test data into which species** they belong to, out of three output options and increase our accuracy.

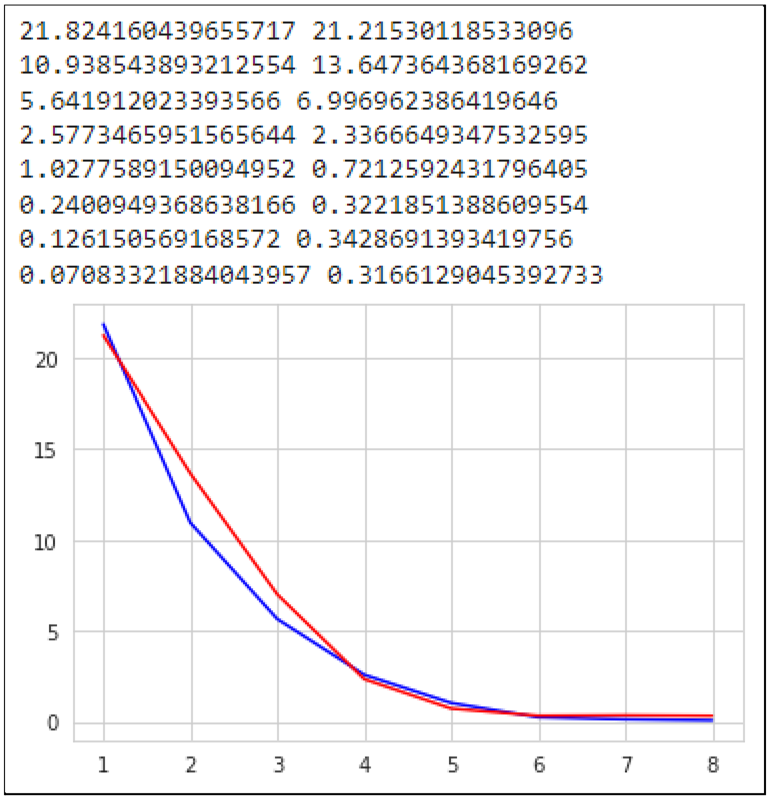


- **(1.1C)** The categorical variables, namely sex, island, species, and year were encoded into 0’s, 1’s and 2’s based on the number of unique values. The data was then split into the training set(0.8) and testing set(0.2).
- **(1.2)** The functions of gini_index and information gain(gini gain) were calculated. More the gini gain, more the reduction in impurity and the feature with the maximum gini gain is chosen for splitting. Gini index is calculated from the formula :
$$Gini = 1 - \sum_j p_j^2$$
- **(1.3)** The continuous data needs to be converted to categorical data in order to be processed. This includes ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g']. For each feature, we first find the unique values and consider each of them as a threshold for splitting the data. The split that **leads to the maximum information gain** will be the values that the data of the column is split on.
- **(1.4)** Firstly, I have defined two classes, one for each node and another one for the decision tree. The node class has attributes such as which feature it is split on, left subtree, right subtree, mid subtree, information gain and the feature value predicted if it is a leaf node. The tree class has methods such as getting the best split, building the tree, printing the tree,etc.
- Looking at the method to get the best split, I have firstly created a dictionary to store the values I get.The maximum information gain is initialised to a large negative number. For **each feature**, I have found the information gain and if it is greater than the maximum information gain, I have replaced the values of the dictionary with these ones. We then return the dictionary which has the same attributes as the node.
- In order to build the tree, we call the method to get the best split **recursively** on the current dataset. In case it is a leaf node, information gain will be zero. If we are able to split the data(i.e. Information gain>0), we call the best split method recursively on the left, right and middle subtrees. If information gain is zero, it is a leaf node and we store the maximum of the values of the target variable in the current dataset in that node.
- **(1.5)** The maximum depth attribute is added when the tree is initialized. When we are recursively calling the left and right subtrees, we make sure that the current depth is less than the maximum depth. The recursion will only be done when the information gain is greater than zero. In a case where it is always zero, we will come out of the loop and an output feature value will be assigned to the node by **majority value.**

- **(1.6)** The fit method in the class of the tree concatenates the x and y values passed as a parameter and passes them to the function that builds the tree. In order to predict a certain input, we pass the feature values of that input to another function which **traverses through the tree** until it reaches a leaf node. According to my definition, if the value is 0 it will go to the left subtree, 1 will go to right and if it is 2 it will go to the middle subtree.
- **(1.7)** The overall accuracy is calculated by checking the number of right predictions divided by the number of total predictions. The average class wise accuracy involves a 3x3 matrix, calculating the correctly predicted samples divided by the number of predicted samples of their class and then taking their average.

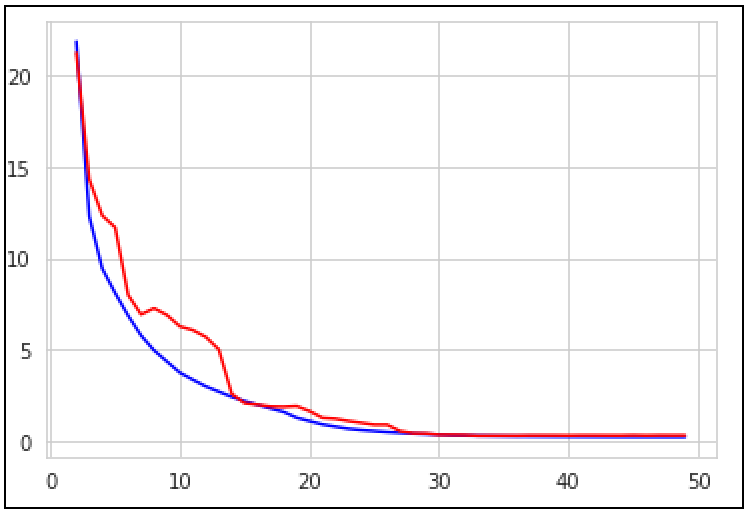
Question 2:

- **(2.1)** The data was imported from a file named “Lab2_Q2.csv”. Here the data does not need to be encoded as we are using a regression decision tree. The data was studied and no NaN values were found. The data was split into three parts, training, testing and validation, 70:20:10 respectively.
- **(2.2)** First, I plotted the tree in order to get a rough idea of how it looked. The tree was quite overfitted. To tune the max depth parameter, I plotted the **Mean Square Error(MSE)** of each max depth in the range 1 to 8. The error seems to be quite less for a max depth of around 5 and 6,hence this range was chosen. The **red line represents the mse on the validation** set whereas the **blue line is the mse on the training set.** In the beginning there is underfitting and hence error on both sets is high. Towards the end, there is overfitting then resulting in minimal error.

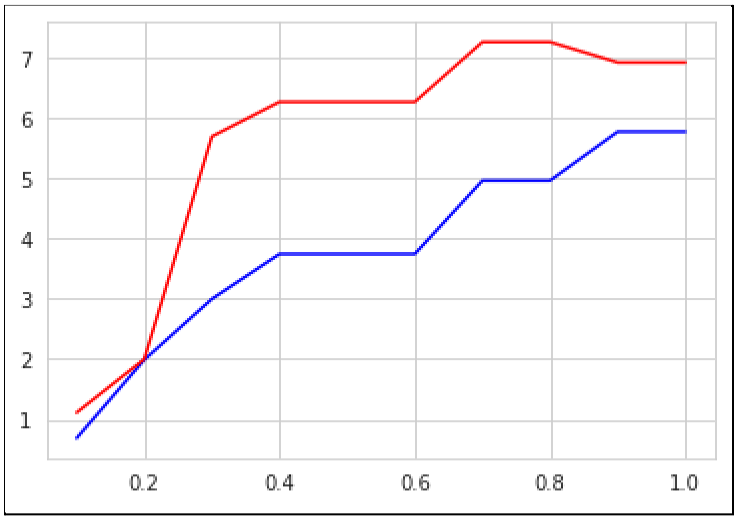


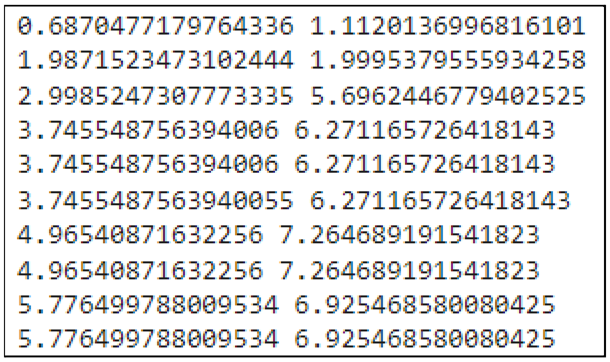
```
21.824160439655717 21.21530118533096
10.938543893212554 13.647364368169262
5.641912023393566 6.996962386419646
2.5773465951565644 2.3366649347532595
1.0277589150094952 0.7212592431796405
0.2400949368638166 0.3221851388609554
0.126150569168572 0.3428691393419756
0.07083321884043957 0.3166129045392733
```

- Similarly, I checked for maximum leaf nodes in range 2 to 50.



- Lastly, for minimum impurity decrease in range 0 to 1, with a step value of 0.1. Plotted values are shown alongside.



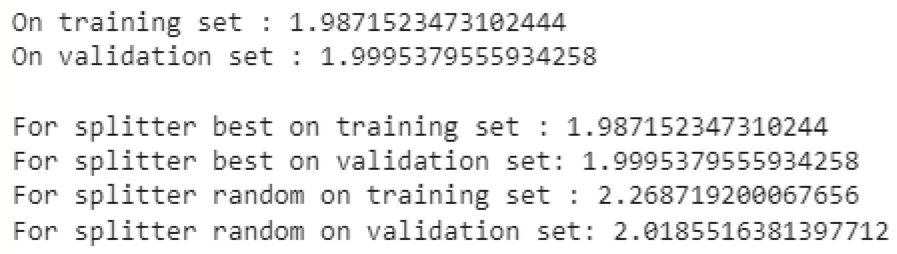


```
0.6870477179764336 1.1120136996816101
1.9871523473102444 1.9995379555934258
2.9985247307773335 5.6962446779402525
3.745548756394006 6.271165726418143
3.745548756394006 6.271165726418143
3.7455487563940055 6.271165726418143
4.96540871632256 7.264689191541823
4.96540871632256 7.264689191541823
5.776499788009534 6.925468580080425
5.776499788009534 6.925468580080425
```

- Checking for the type of splitting, I called a function to calculate mse on training and validation set using the above parameters, and passed splitting for each. Results -

```
# splitter 'best' has equal mse on both sets and is consistent
# splitter 'random' has high error difference between and on  validation
set and training set comparatively, is unpredictable as it changes on
every run
```

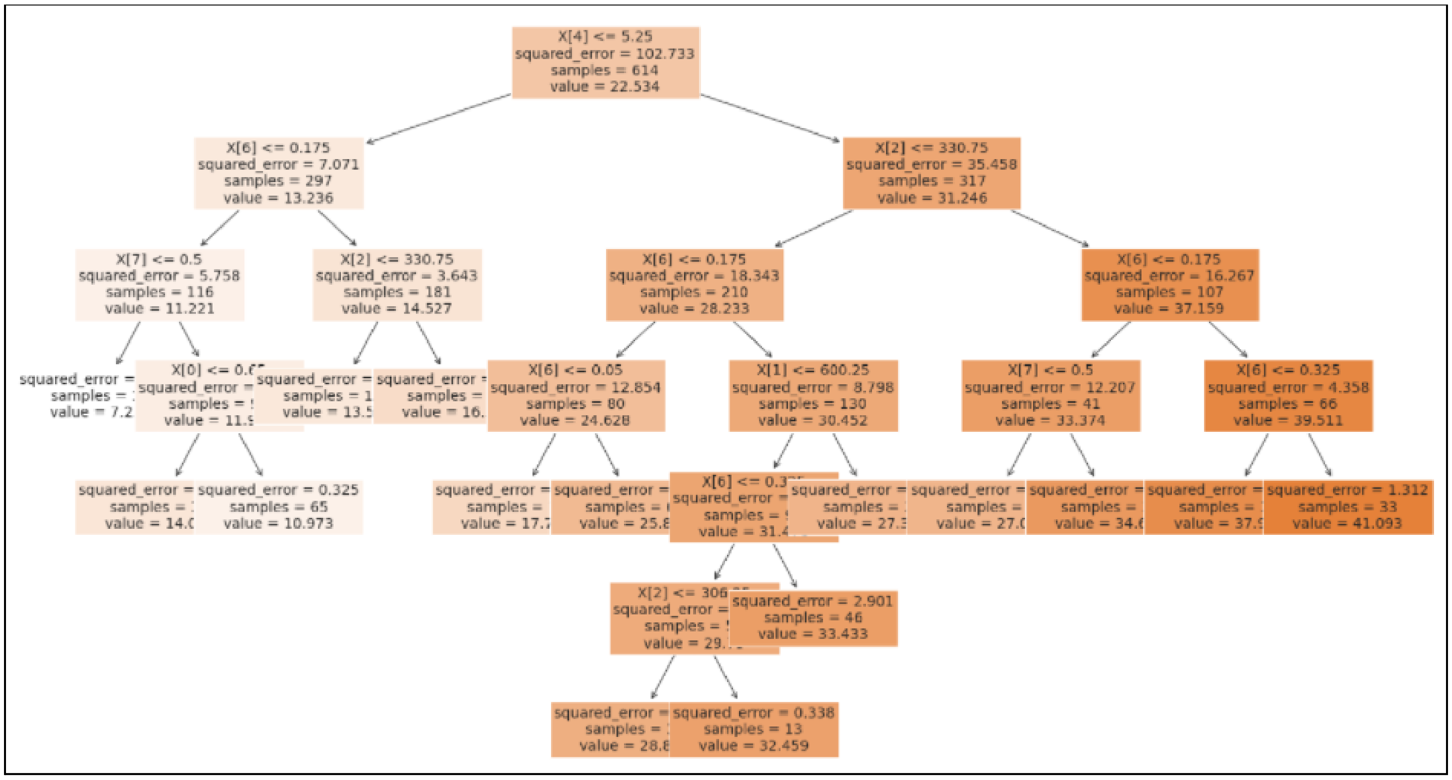
MSE values -



```
On training set : 1.9871523473102444
On validation set : 1.9995379555934258

For splitter best on training set : 1.987152347310244
For splitter best on validation set: 1.9995379555934258
For splitter random on training set : 2.268719200067656
For splitter random on validation set: 2.0185516381397712
```

- For the best performance,
  splitter='best'
  max_depth=6
  max_leaf_nodes=25
  min_impurity_decrease=0.2
- **(2.3)** Since this is a regression problem, we cannot use accuracy score to predict the accuracy and need to use MSE instead. I have performed 5 fold cross validation using the (training set+validation set) in the previous split as the training set. For each split, I have calculated the MSE and taken an average of them in the end to find the MSE for the overall model.
- The final decision tree looks like (plotted using graph) -



X[4] <= 5.25
squared_error = 102.733
samples = 614
value = 22.534
X[6] <= 0.175
squared_error = 7.071
samples = 297
value = 13.236
X[2] <= 330.75
squared_error = 35.458
samples = 317
value = 31.246
X[7] <= 0.5
squared_error = 5.758
samples = 116
value = 11.221
X[2] <= 330.75
squared_error = 3.643
samples = 181
value = 14.527
X[6] <= 0.175
squared_error = 18.343
samples = 210
value = 28.233
X[6] <= 0.175
squared_error = 16.267
samples = 107
value = 37.159
X[6] <= 0.05
squared_error = 12.854
samples = 80
value = 24.628
X[1] <= 600.25
squared_error = 8.798
samples = 130
value = 30.452
X[7] <= 0.5
squared_error = 12.207
samples = 41
value = 33.374
X[6] <= 0.325
squared_error = 4.358
samples = 66
value = 39.511
squared_error = 0.325
samples = 65
value = 10.973
squared_error = 1.312
samples = 33
value = 41.093
squared_error = 2.901
samples = 46
value = 33.433
squared_error = 0.338
samples = 13
value = 32.459