Decision Trees
Lab Report

Name: Palak Singh

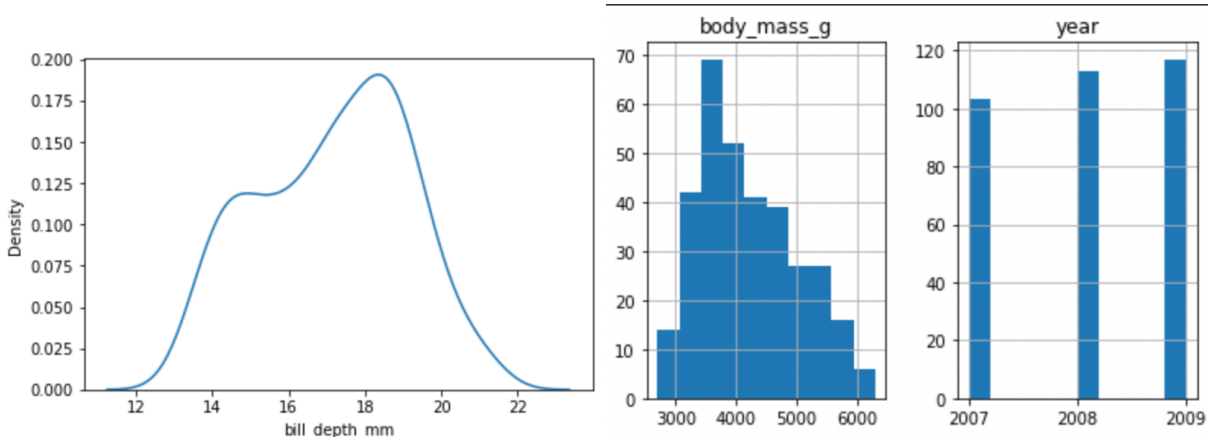
Question 1: Decision Tree- Classification Of Penguins

Step 1: Preprocessing of the Data

1. The data had some missing values which cannot be passed to the model. Hence, they either have to be dropped or replaced by some numerical data.
2. It was observed that dropping the rows with missing values would lead to a loss of about 3.19% of the total data. We observe that only 3% of data is lost so it will not have much effect on the model accuracy.

Step 2: Visualizing the Data

1. The features present in the dataset were plotted. For "`bill_length_mm`","`bill_depth_mm`" and "`flipper_length_mm`", a Kernel Density Estimate (KDE) plot was plotted as the data was of continuous nature. For all other features, histogram plots were used as discrete bands were visible in the data. The following is the plotted curves:



Step 3: Categorical Encoding

1. Label Encoding was performed on some features to convert the labels into a numeric form so as to convert them into the machine-readable form using the 'LabelEncoder' from 'sklearn.preprocessing'.

Step 4: Splitting the Training and Testing Dataset

1. The target variable in the given dataset is “species_n” was labelled as Y and all other features were labelled under X.
2. The data was then divided into training and testing data in the ratio of 80:20, the training set was used to train our model, whereas the testing set will later be used to check the credibility of the model created.

Step 5: Implementation of the cost function: Entropy

1. Here a function entropy is created with a parameter y, y is the feature of which the entropy has to be calculated. Entropy is nothing but the measure of disorder in a set of values.
2. The following formula is implemented to calculate entropy.

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Where ‘Pi’ is simply the frequentist probability of an class ‘i’ in our data.

Step 6: Implementation of cont_to_cat function for encoding the continuous data into categorical data

1. Function cont_to_cat was created to convert data from continuous to categorical form.
2. Different attributes are provided as a parameter to this function to get encoded into 2 different sets based on the threshold value, to find the threshold value, the information gain using entropy at every threshold value was calculated the threshold value which gave the maximum information gain was selected and then used for dividing the data into 2 categories.
3. For performing the above task the idea which i have followed was that i iterated over all the values present in an attribute and made these values the threshold and calculated the information gain for all these values.
4. For assigning the value bins were created where the values which were greater than the threshold were assigned 1 and the values less than the threshold value were assigned to be 0.
5. The formula implemented for information gain was

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

6. The cont_to_cat function was used to convert the values of the attribute 'bill_length_mm', 'bill_depth_mm','flipper_length_mm'.

Step 7: Training the Model - Constructing the decision tree

1. After converting the data to the categorical form, a fraction of the data obtained was used to train the model.
2. For training the data, construction of the decision tree was done using the class DTC.
3. As the first step of construction on a decision tree a class- node was created that stored all the necessary information for every leaf which were as follows:
   1. fti: feature index: stores the index of the feature stored in that node
   2. Thres: threshold: threshold value, which forms the basis of split of the left and the right sub tree
   3. left: the left subtree
   4. Right: the right subtree
   5. i_g: information gain corresponding to that node, calculated by using entropy
   6. val: Value: has the respective output value for a leaf node
4. The class DTC consists of a constructor which defines the root node, the other function called build_tree which was used for building the decision tree, this function states the stopping condition of the decision tree, it checks that the tree doesn't exceed the max depth stated in the parameters of DTC and it also stop the division when the total number of splits of node exceeds the minimum sample split.
5. Another function called get_best_split which is called by the build_tree function uses information gain calculated using entropy to divide the tree into the most optimum branches and thus generating an efficient split.
6. Further, these functions were called again and again until any one of the stopping conditions stated in the build_tree function was attained.
7. In the given example of execution of code, a decision tree with a max depth of 5 was created.

Step 8: predict Function for classification

1. A function called predict is created in the class DTC, which is used for the prediction of data in the final step of testing, this function consisted of a

parameter for passing the dataset to the function, this function called another function called prediction which classified the given data based on the information provided by the user and then analysing the decision tree.
2. Therefore, the array y_pred was obtained which was the result obtained from applying the model to the testing data.
3. Further, a function accuracy_score from sklearn.metrics was used to find the final accuracy of the model.

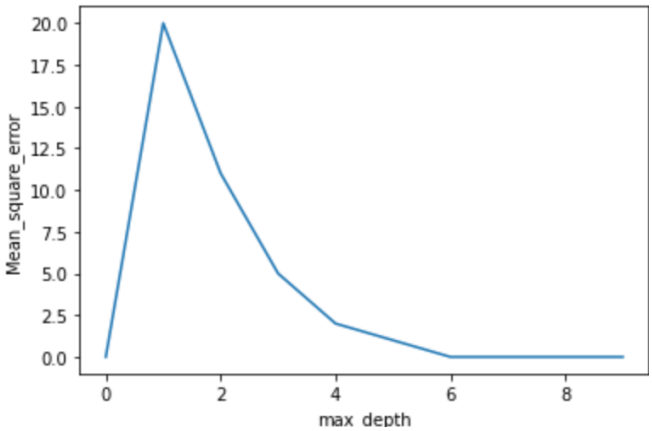
Question 2: Regression using decision trees

Step 1: Preprocessing the dataset

1. The data was divided into train and test data in the ratio of 4:1, further the training data was divided into the validation data and the training data which was used for training the model, as the training data was 80% of the total data, now on again division the new ratio was 12.5:87.5. Therefore the resulting division of the data set was training:validation:testing in the ratio 70:10:20 ratio.

Step 2: Training the data using a regression decision tree.

1. 'DecisionTreeRegressor' from 'sklearn.tree' was used to implement the regression decision tree for training the data.
2. The hyperparameter chosen by me were, max_depth and min_samples_leaf.
3. Max_depth indicated how deep the tree can be formed, and we know that the deeper the tree the greater the amount of detailing we get.



Here in this regression model we observe that the mse starts to become constant after maximum depth of 6, so we find that 6 is the optimum max depth of the model.

Step 3: 5-fold cross validation and plotting the graph

1.
```
from sklearn.model_selection import cross_val_score
```
   The above mentioned library function was used to do k fold validation of the model with the optimum value of the hyperparameters max_depth and min_samples_leaf found in the above part of the question.
2. Mean squared error was calculated for every part of the 5-folds which were made for cross validation.
3. Using the sklearn.tree library the tree was plotted which can be seen in the colab file attached with the report.