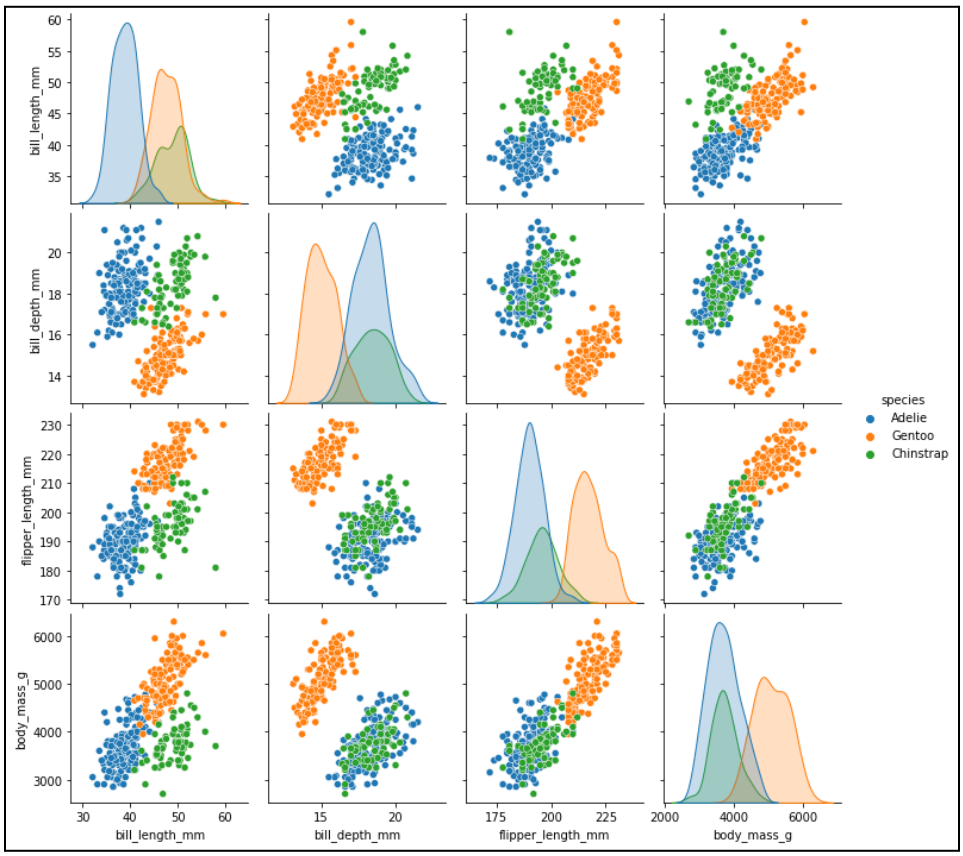
PRML LAB 2 REPORT

Q1



Part 1) Pre-processing, encoding and splitting data

1. Data rows with NaN values are dropped.
2. Column ‘Year’ is dropped.
3. Data is visualized by plotting.
4. Categorical columns in data are encoded .
5. Given data is split in a 75:25 ratio for training and testing.
6. I have used custom made functions (encode_data, split_data) for encoding categorical data and splitting data (in a ratio 75:25) into training and testing sets respectively.
7. Inbuilt function used : seaborn for visualizing input data, LabelEncoder for encoding input data.

Part 2) Cost Function

1. The cost function is implemented by calculating entropy.
2. Formula for entropy for a data is:

$$Entropy = \sum_{i=1}^{n} -p(c_i)log_2(p(c_i))$$

3. I have used custom made functions : info_gain, cost_data, entropy_data for calculating information gain for data against a particular feature, calculating weighted entropy of data against all values of the given feature, calculating entropy/impurity for the given sub-data.

Part 3) Converting continuous feature to categorical and encode them

1. Continuous feature of training data is converted into categorical and encoded by making optimal splits by selecting a threshold such that the optimal split leads to minimum weighted entropy. Also a threshold list is made containing the thresholds for all continuous variables that were used to convert them into categorical.
2. The threshold list is used to convert the continuous feature of testing data into categorical.
3. Custom made functions used: categorical , cont_to_cat , test_cont_to_cat for converting a particular continuous feature of training data into categorical, converting all continuous feature of training data into categorical, converting all continuous feature of testing data into categorical based on threshold obtained while converting training data.

Part 4) Implementing training function and its helper functions

1. Decision tree is constructed based on training.
2. For constructing the decision tree , at each stage we find the best possible attribute/feature that has the maximum information gain and make it the node and the data is split into multiple parts based on the value of the feature in each row. This best attribute is then dropped from the feature space for further construction .
3. The above process is repeated until we reach a point where there is no data, or all data belongs to the same class/species or the feature space is empty.
4. For the construction of the decision tree, the training function is paired with custom made helper function (find_best_attribute and split_data_on_feature) to find attribute with maximum information gain and splitting data based on a feature respectively.

Part 5) Constructing Decision tree

1. The training function has the property to stop growing if it crosses given maximum depth or there is no more information to gain. This ensures that the decision tree is not overfitted to training data.
2. Custom function used : depth_check, check_info_gain for checking if the depth of the tree has crossed max depth or not, to check if the information gain is too small for the tree to grow further.

Part 6) Classifying testing data with decision trees

1. The testing set is classified into different classes/species based on the prediction from the decision tree.
2. Custom function used : predict_on_data, prediction for classifying complete testing set into different species , classifying individual data entry of testing data to a species.
3. Inbuilt function used : scatter() for visualizing the classification of testing data.

Part 7) Calculating overall and class wise accuracy

1. Overall accuracy and class wise accuracy is computed by comparing the predicted values to the actual values.
2. Custom function used : overall_accuracy for computing overall accuracy , classwise_accuracy for computing class wise accuracy.
3. Inbuilt function used : scatter() for visualizing the overall accuracy of prediction of decision tree on testing data.

Q2)

Part 1) pre-processing and splitting data

1. Data rows with NaN values are dropped.
2. Input data is divided into training , validation and testing and further split into x_train, y_train, etc.
3. Functions used : split_data_2 for splitting input data into training , validation and testing and further split into x_train, y_train, etc.

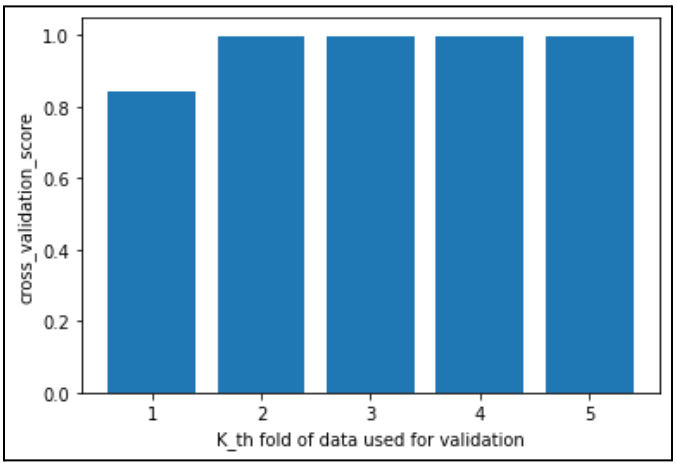
Part 2) Finding best hyperparameters

As decision trees are prone to overfitting we have to set hyperparameters, such that the regression tree does not overfit to training data and gives least error on validation data. To do this we change and adjust values of hyperparameters (maximum depth and minimum sample split) . We train our regression tree with various combinations of maximum depth and minimum sample split and compare their mean square error on validation data. The combination with least error is our optimal hyperparameter.

- Custom made functions used : optimal_hyperparameter for finding optimal depth and optimal split.
- Inbuilt function used : DecisionTreeRegressor , mean_squared_error for training regression tree on training data and for calculating mean square error of regression tree on validation data respectively.

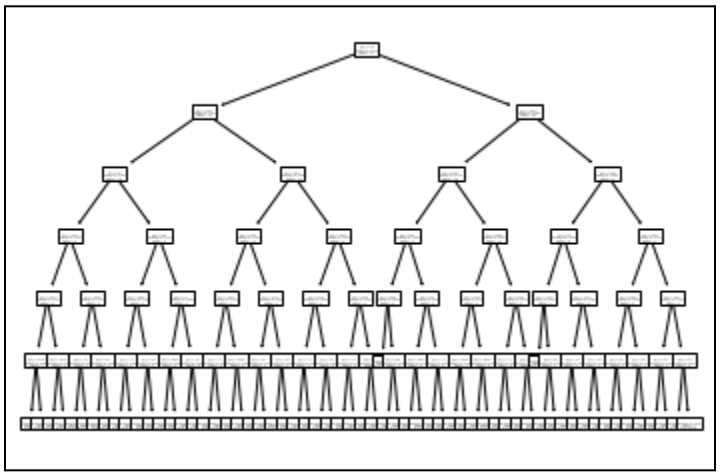
Part 3) Cross-validation, mse on testing data and printing the regression decision tree

1. We perform k-fold cross validation by dividing the input data in k parts , then we train the regression tree (having the optimal hyperparameter calculated in the previous part) on k-1 parts and do validation on the kth part.
2. K-fold cross validation is used because it ensures that every data point is used for training and testing at least once.



3. The regression tree was now trained on training data (with the previous hyperparameters) and its mean square error on testing data is calculated.

4. Final regression tree is printed.



5. Inbuilt function used : cross_val_score for calculating cross validation score, DecisionTreeRegressor for training final regression tree .