Procedure and Results

- **Members (GR-G, 65)**
  - **Name:** Surajit Kundu, **Roll No:** 21MM91R09
  - **Name:** Ankur Kumar Jaiswal, **Roll No:** 21MM62R08

❖ Introduction

We have used the ID3 Algorithm for building the decision tree. The impurity of each node can be measured using both Gini and Information gain to construct the tree. We have used the Python programming language. And the following are used, libraries are NumPy, math.

A class is created named **ID3DecisionTreeClassifier.** All the required functions are defined in this class. There are few parameters like criterion, maximum depth of the tree, the threshold value of minimum gain. These can be set during the instance creation of the class.

❖ Impurity Measurement

● Gini Index

To calculate the Gini Index of an attribute from the feature set., we use the function **gini( )**. We have given the training_data, input_feature_set, and target_attribute as input. The Gini Index is determined by deducting the sum of squared of probabilities of each class from one; mathematically, Gini Index can be expressed as:

$$\text{Gini Index} = 1 - \sum_{i=1}^{n} (P_i)^2$$

Where Pi denotes the probability of an element being classified for a distinct class. First, we have calculated the probabilities of positive(1) and negative(0) examples in each input feature attribute and target attribute. Then, separately calculate the gini index of each input

attribute and target attribute by subtracting the square of the probabilities by 1. At last, We will have the gini_index of the given dataset as the output.

- **Entropy**

Entropy is a measure of uncertainty about a random variable. To calculate the entropy note, we are using the function **entropy()**. This function calculates the entropy of an attribute from the feature set. We have given the training_data, input_feature_node, and target_attribute as input.

$$\text{Entropy} = -\sum_{i=1}^{n} p_i * \text{Log}_2(p_i)$$

Here “pi” denotes the probability that it is a function of entropy. First, we calculated the probabilities of positive(1) and negative(0) examples in each input feature and target attribute. Calculate the log separately with base 2 of the probabilities of the input feature attribute and target attribute. Add these probabilities and subtract them by 1. At last, We will have the entropy of the given dataset as the output.

- **Information Gain**

Information gain is used for determining the best feature that renders maximum information about a class. To calculate the Information gain of an attribute from the feature set, we use the function **InformationGain()**. We have given the training_data, input_feature_node, and target_attribute as input. The formula for information will be expressed as :

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

We have calculated information gain by subtracting the entropy of the target node with the entropy of each input attribute in their defined proportion when entropy is taken into consideration.

❖ Decision Node Selection

The attribute with the highest Information gain will be chosen as the root node of the decision tree. The root(parent) node will divide into two child nodes. And the input feature with the subsequent highest gain other than the parent node will be taken as the child node. In this way, with the decreasing information gain of the attributes, more subtrees will be formed until all the training data is fit to the tree.

We have defined a new function named **BestFeatureAttribute().** Based on the information gain, this function will give the best feature attributes. The **argmax()** function finds the index value which has the maximum information gain. Then, find the name of the best feature which has the maximum gain. And remove it from the features list.

❖ Building Decision Tree

We have defined a function in ID3DecisionTreeClassifier, named **DecisionTree()**. This function builds the decision tree using the ID3 Algorithm. We have given a target attribute, input feature set, and training data.
This function first calculates the impurity measures, i.e., information gain or gini index of input feature attributes. Then, with the help of these impurities measures, it calculates the maximum gain. The feature with the maximum gain is selected as the best attribute and the decision tree's root node. Likewise, it will repeat until all the training data fit the tree.

❖ Read Dataset

To read the Parkinson's dataset, we have used the panda's library function. Using the panda's library **read_csv()** function, we read the CSV file into the data frame format. We have dropped the name and status column from the input data and chosen the status column as the target value.

❖ Data Preprocessing

As the Parkinson's dataset contains attribute values in numerical form, We have converted the data into binarized(0 or 1) data. In this binarization task, we first scale the data using Min-Max scaling. Then each tuple is compared with the mean. If the tuple is greater than the mean, the value is converted to 1. Otherwise, make it 0. There exist other standard ways of transforming the numerical values to intervals. In scikit-learn, there is a package named **KBinsDiscretizer(),** which performs the task of converting continuous or numerical data into a given range. Now comes the splitting part; using sklearn, We have imported the **train_test_split()** function. And using this function, we have split the dataset into an 80:20 ratio. It means we have randomly split the dataset into training data of 80 %, and test data is 20 %.

❖ Classifier Building

Using **ID3DecisionTreeClassifier,** build the model, then fit the training data into the decision tree. And predict the decision tree using the **predict()** function. Finally, with the help of the **accuracy_score()** function using criteria like entropy and gini index, we have calculated the accuracy of the test data. Using the gini index as the impurity measure, we have achieved a test accuracy of 87.17 %. When using entropy as the impurity measure, we have achieved a test accuracy of 89.74 %.

❖ Accuracy by averaging over 10 random 80/20 split

We have randomly split the dataset into an 80/20 ratio. Then, we built the decision tree by randomly splitting the 80 % data and predicted the model 10 times with 20% test data. We have noted down the accuracy each time. At last, we have taken the average of the accuracy score. The accuracy by averaging over ten random 80/20 splits is 86.98% using

entropy measure and 84.45% using Gini index. In the ten accuracy measures, the decision tree with the best test accuracy is 94.11 %.

❖ Decision Tree Depth

We have imported the **matplotlib.pyplot** library for plotting the graphs. When we have generated a graph between test accuracy and depth. In the plot, as we increase the depth of the decision tree, the accuracy is also increasing and we are getting the maximum accuracy at depth equal to 8. And When we have generated the graph between accuracy and number of nodes. In the plot, as we increase the number of nodes of the tree, the test accuracy is also increasing.

❖ Pruning

For pruning the decision tree, We have taken the same model(as in question 2) where we achieve the highest accuracy over 10 random 80/20 splits. Then we have performed pruning on the node impurity. We have also generated a graph between accuracy and impurity. We set a list of threshold values for impurity measure. The threshold value decides when a new node is created. We are gradually increasing the threshold values of the impurity in the plot , as we can see the accuracy is also increasing and accuracy is maximum when the threshold value is around 0.04.