Gini Index

- Gini index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when randomly chosen.
- But what is actually meant by 'impurity'? If all the elements belong to a single class, then it can be called pure. The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes.
- A Gini Index of 0.5 denotes equally distributed elements into some classes.

- **Formula for Gini Index**

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

- where p_i is the probability of an object being classified to a particular class.
- While building the decision tree, we would prefer choosing the attribute/feature with the least Gini index as the root node.

Implementing Decision Trees

Step 1: Importing the Modules

- We import the DecisionTreeClassifier class from the sklearn package. This is an in-built class where the entire decision tree algorithm is coded.
- In this program, we shall use the iris dataset that can be imported from sklearn.datasets.
- The pydotplus package is used for visualizing the decision tree.

```
import pydotplus
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
```

Implementing DT

Step 2: Exploring the data

- Next, we make our data ready by loading it from the datasets package using the load_iris() method.
- We assign the data to the iris variable. This iris variable has two keys, one is a data key where all the inputs are present, namely, sepal length, sepal width, petal length, and petal width.
- In the target key, we have the flower type which has the values, Iris Setosa, Iris Versicolour, and Iris Virginica. We load these in the features and target variables respectively.

```
iris = datasets.load_iris()
features = iris.data
target = iris.target
print(features)
print(target)
```

Implementing DT

Step 3: Create a decision tree classifier object

- Here, we load the DecisionTreeClassifier in a variable named model, which was imported earlier from the sklearn package.

```
Decisiontree=DecisionTreeClassifier(random_state=0)
```

Step 4: Fitting the Model

- This is the core part of the training process where the decision tree is constructed by making splits in the given data. We train the algorithm with features and target values that are sent as arguments to the fit() method. This method is to fit the data by training the model on features and target.

```
model = decisiontree.fit(features, target)
```

Implementing DT

Step 5: Making the Predictions

- In this step, we take a sample observation and make a prediction. We create a new list comprising the flower sepal and petal dimensions. Further, we use the predict() method on the trained model to check for the class it belongs to. We can also check the probability (class probability) of the prediction by using the predict_proba method.

```
observation = [[ 5, 4, 3, 2]] # Predict observation's class
model.predict(observation)
model.predict_proba(observation)
```

Implementing DT

Step 7: Dot Data for the predictions

- In this step, we export our trained model in DOT format (a graph description language). To achieve that, we use the tree class that can be imported from the sklearn package. On top of that, we use the export_graphviz method with the decision tree, features and the target variables as the parameters.

```
from sklearn import tree
 dot_data = tree.export_graphviz(decisiontree,
out_file=None, feature_names=iris.feature_names,
class_names=iris.target_names )
```

Implementing DT

Step 8: Drawing the Graph

- In the last step, we visualize the decision tree using an Image class that is to be imported from the IPython.display package.

```
from IPython.display import Image
 graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```

Q.1 Define Gini Index?

Q.2 Discuss the implementing of Decision trees?

Q.3 Try the following steps-

- Importing the dataset
- Exploring the dataset
- Loading the libraries for DT
- Predictions &
- Visualizations