Experiment No: 05

●**Aim: To implement Graph based clustering and CART algorithm.**

●Theory:

Graph Clustering:

Data mining involves analyzing large data sets, which helps you to identify essential rules and patterns in your data story. On the other hand, graph clustering is classifying similar objects in different clusters on one graph. In a biological instance, the objects can have similar physiological features, such as body height. Still, the objects can be of the same species. When you want to perform graph clustering, some parameters you can consider include data point density and the distance between data points. If you are a data scientist or a retailer, you have a significant role in graph clustering. This is because it can help you gather vital information on how data points relate to each other. When you use graph clustering methods in data mining, you identify relationships in your data story.

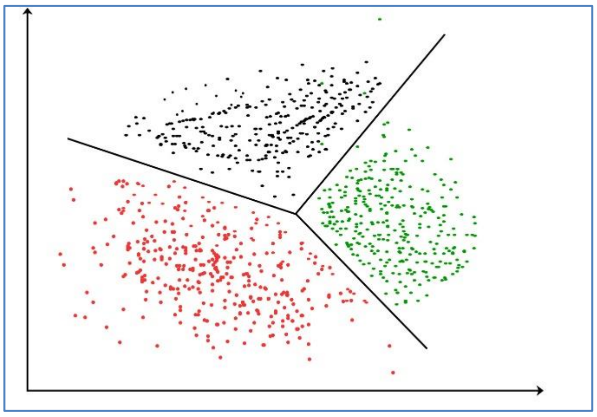
Hierarchical Graph Clustering:

It is one of the most common graph clustering methods you can use. When you utilize this clustering method, your graph appears as partitions of hierarchical structures. This method has two types of strategies, namely:

- Divisive strategy
- Agglomerative strategy

When you use this method, you partition your graph (line chart, bar chart, gauge chart, etc.) into clusters that appear with a k-shape. You can use k-means to compute the centroids and their vector quantities. It takes you easy steps in constructing the K-mean clustering method, which includes:

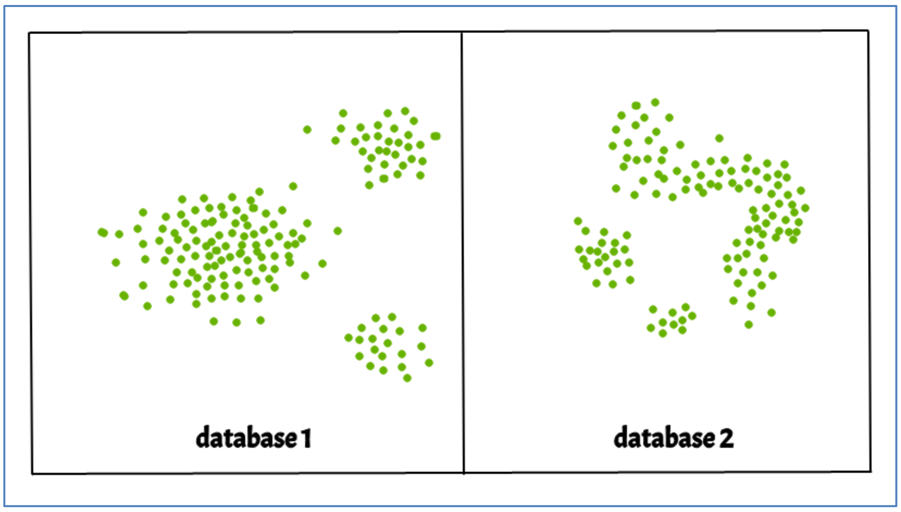
1. You can select K to represent the original centroids.
2. Next is to assign points to the nearest centroids from your K-cluster.
3. Conduct a re-evaluation of your centroids to ensure they do not change.
4. One advantage you can obtain from this method is that it can help you process large data sets.



Density-Based Graph Clustering Method:

Density-based methods work wonders when you want to identify clusters in larger data sets. This is because you can analyze data points based on their density. Two close data points are termed, neighbors. The concept behind this method is density connectivity and density reachability. You will find many data scientists and business persons using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Some of the steps you can use in this method include:

- You can begin the clustering process when you find enough data points in your graph.
- Your current data point acts as the starting point.
- Your starting point to the new cluster uses a similar distance which defines its neighborhood.
- You will continue this process in every new cluster until you label all data points.
- An advantage you can find from this graph clustering method is that you can recognize noisy data.



CART Algorithm:

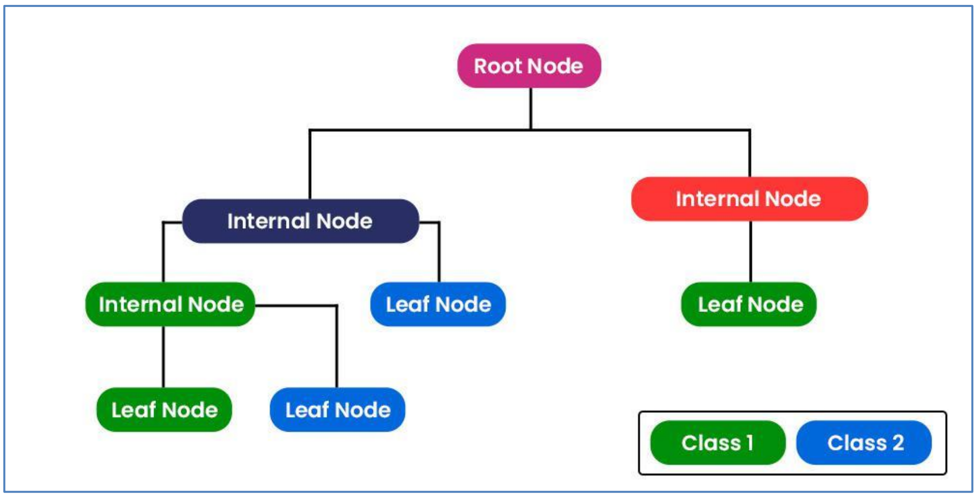
CART(Classification And Regression Tree) is a variation of the decision tree algorithm. It can handle both classification and regression tasks. Scikit-Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees (also called "growing" trees). CART was first produced by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone in 1984.

CART is a predictive algorithm used in Machine learning and it explains how the target variable's values can be predicted based on other matters. It is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end.

In the decision tree, nodes are split into sub-nodes on the basis of a threshold value of an attribute. The root node is taken as the training set and is split into two by considering the best attribute and threshold value. Further, the subsets are also split using the same logic. This continues till the last pure sub-set is found in the tree or the maximum number of leaves possible in that growing tree.

The CART algorithm works via the following process:

- The best split point of each input is obtained.
- Based on the best split points of each input in Step 1, the new "best" split point is identified.
- Split the chosen input according to the "best" split point.
- Continue splitting until a stopping rule is satisfied or no further desirable splitting is available.



Classification tree

A classification tree is an algorithm where the target variable is categorical. The algorithm is then used to identify the “Class” within which the target variable is most likely to fall. Classification trees are used when the dataset needs to be split into classes that belong to the response variable(like yes or no)

Regression tree

A Regression tree is an algorithm where the target variable is continuous and the tree is used to predict its value. Regression trees are used when the response variable is continuous. For example, if the response variable is the temperature of the day.

Pseudo-code of the CART algorithm

```
d = 0, endtree = 0
Note(0) = 1, Node(1) = 0, Node(2) = 0
while endtree < 1
  if Node(2d-1) + Node(2d) + .... + Node(2d+1-2) = 2 - 2d+1
    endtree = 1
  else
    do i = 2d-1, 2d, .... , 2d+1-2
      if Node(i) > -1
        Split tree
      else
        Node(2i+1) = -1
        Node(2i+2) = -1
      end if
    end do
  end if
d = d + 1
end while
```

CART model representation

CART models are formed by picking input variables and evaluating split points on those variables until an appropriate tree is produced.

Steps to create a Decision Tree using the CART algorithm:

Greedy algorithm: In this The input space is divided using the Greedy method which is known as a recursive binary spitting. This is a numerical method within which all of the values are aligned and several other split points are tried and assessed using a cost function.

Stopping Criterion: As it works its way down the tree with the training data, the recursive binary splitting method described above must know when to stop splitting. The most frequent halting method is to utilize a minimum amount of training data allocated to every leaf node. If the count is smaller than the specified threshold, the split is rejected and also the node is considered the last leaf node.

Tree pruning: Decision tree's complexity is defined as the number of splits in the tree. Trees with fewer branches are recommended as they are simple to grasp and less prone to cluster the data. Working through each leaf node in the tree and evaluating the effect of deleting it using a hold-out test set is the quickest and simplest pruning approach.

Data preparation for the CART: No special data preparation is required for the CART algorithm.

Advantages of CART

- Results are simplistic.
- Classification and regression trees are Nonparametric and Nonlinear.
- Classification and regression trees implicitly perform feature selection.
- Outliers have no meaningful effect on CART.
- It requires minimal supervision and produces easy-to-understand models.

Limitations of CART

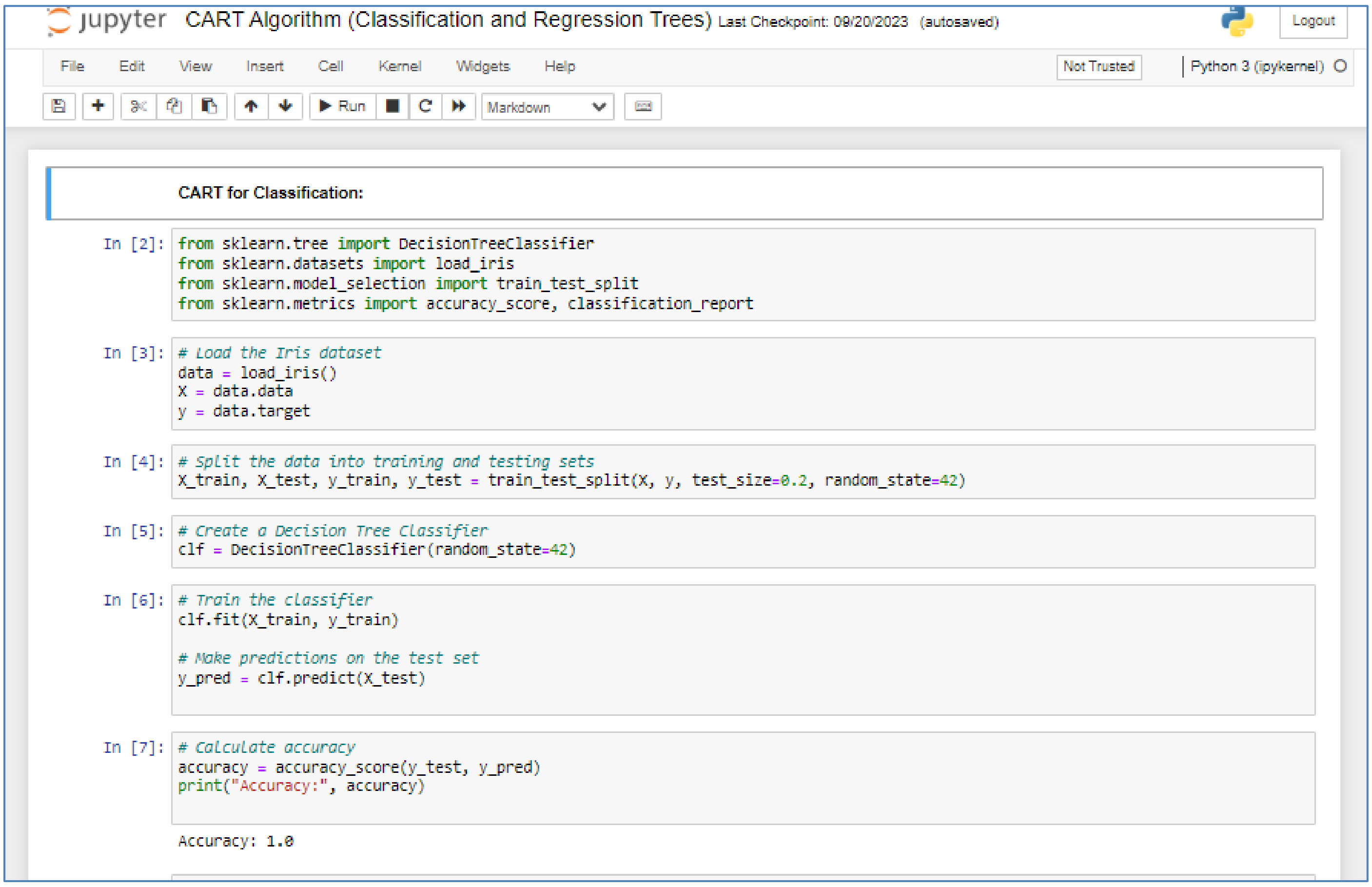
- Overfitting.
- High Variance.
- low bias.
- the tree structure may be unstable.

Applications of the CART algorithm

- For quick Data insights.
- In Blood Donors Classification.

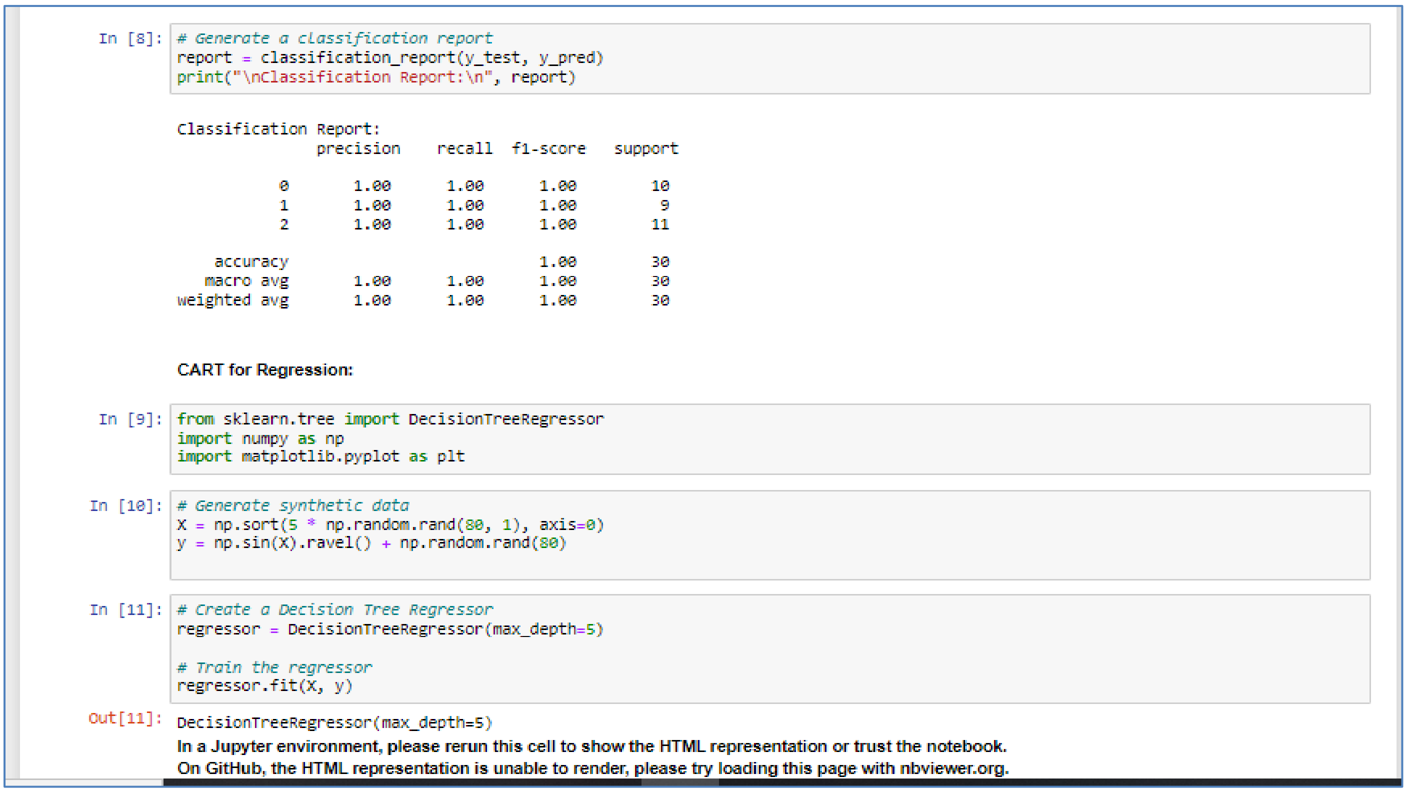
- For environmental and ecological data.

Implementation of Graph based clustering and CART algorithm:


jupyter CART Algorithm (Classification and Regression Trees) Last Checkpoint: 09/20/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

CART for Classification:

```
In [2]: from sklearn.tree import DecisionTreeClassifier
        from sklearn.datasets import load_iris
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score, classification_report

In [3]: # Load the Iris dataset
        data = load_iris()
        X = data.data
        y = data.target

In [4]: # Split the data into training and testing sets
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [5]: # Create a Decision Tree Classifier
        clf = DecisionTreeClassifier(random_state=42)

In [6]: # Train the classifier
        clf.fit(X_train, y_train)

        # Make predictions on the test set
        y_pred = clf.predict(X_test)

In [7]: # Calculate accuracy
        accuracy = accuracy_score(y_test, y_pred)
        print("Accuracy:", accuracy)

        Accuracy: 1.0
```

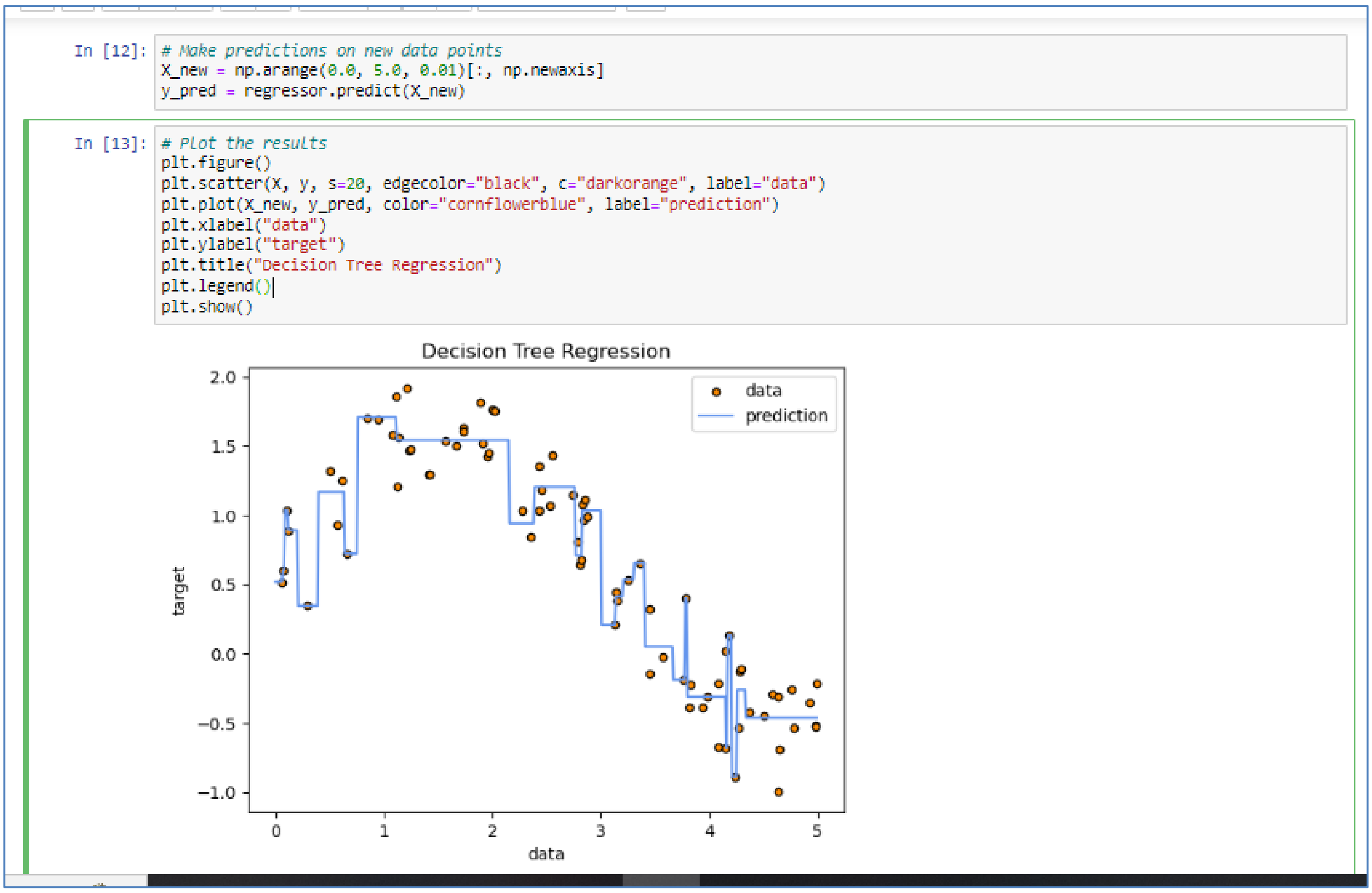


```
In [8]: # Generate a classification report
        report = classification_report(y_test, y_pred)
        print("\nClassification Report:\n", report)

        Classification Report:
                       precision    recall  f1-score   support

                   0       1.00      1.00      1.00        10
                   1       1.00      1.00      1.00         9
                   2       1.00      1.00      1.00        11

            accuracy                           1.00        30
           macro avg       1.00      1.00      1.00        30
        weighted avg       1.00      1.00      1.00        30
```

CART for Regression:

```
In [9]: from sklearn.tree import DecisionTreeRegressor
        import numpy as np
        import matplotlib.pyplot as plt

In [10]: # Generate synthetic data
         X = np.sort(5 * np.random.rand(80, 1), axis=0)
         y = np.sin(X).ravel() + np.random.rand(80)

In [11]: # Create a Decision Tree Regressor
         regressor = DecisionTreeRegressor(max_depth=5)

         # Train the regressor
         regressor.fit(X, y)

Out[11]: DecisionTreeRegressor(max_depth=5)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.



```
In [12]: # Make predictions on new data points
         X_new = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
         y_pred = regressor.predict(X_new)
```

```
In [13]: # Plot the results
         plt.figure()
         plt.scatter(X, y, s=20, edgecolor="black", c="darkorange", label="data")
         plt.plot(X_new, y_pred, color="cornflowerblue", label="prediction")
         plt.xlabel("data")
         plt.ylabel("target")
         plt.title("Decision Tree Regression")
         plt.legend()
         plt.show()
```

● **Conclusion**:

graph-based clustering groups data based on similarity using graph structures, while the CART algorithm creates decision trees for classification and regression tasks.