Problem 1. *Decision Tree by hand*
Consider the following toy dataset with four features (x_1, x_2, x_3, x_4) *and their corresponding class labels (A, B, C):*

Data Point	x_1	x_2	x_3	x_4	Class
1	1	0	1	0	A
2	1	1	0	1	A
3	0	1	1	1	B
4	1	0	0	1	B
5	0	1	1	0	C
6	0	1	0	1	C

Table 1: Dataset with Class Labels

Consider building a Decision Tree, Which feature will be the best for the first split? show all steps

Steps:

1. *Calculate Entropy for the Root Node:*

$$\text{Entropy}\left(S_{root}\right) = -\sum_{i=1}^{c} p_i \log_2\left(p_i\right)$$

where S_{root} *is the set of samples in the root node,* c *is the number of classes, and* p_i *is the proportion of samples of class* i *in the root node.*

2. *Calculate Entropy for Each Possible Split*

3. *Calculate Information Gain:*

$$\text{Entropy}\left(S_{root}\right) - \sum_{i=1}^{k} \frac{|S_i|}{|S_{rowe}|} \times \text{Entropy}\left(S_i\right)$$

- *Entropy* (S_{root}) *is the entropy of the root node.*
- k *is the number of resulting nodes after the split.*
- $|S_i|$ *is the number of samples in the* i *th node after the split.*
- $|S_{root}|$ *is the number of samples in the root node.*

4. *Choose the Best Split*

Problem 2. *Based on the previous example, consider using Python to develop the constructed Decision Tree with entropy as the impurity measure. If we have a new data point with feature values $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, and $x_4 = 0$, which class label would the Decision Tree assign to this data point, and how did it reach this decision?*

To determine the class label for the new data point, we follow the decision path of the Decision Tree. Starting from the root node, the tree uses the feature values $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, and $x_4 = 0$ to make a series of splits until it reaches a leaf node. At the leaf node, the majority class of the training samples in that node determines the predicted class label for the new data point.

To visualize the Decision Tree, we can use the tree.plot_tree function in Python, which provides a diagram of the constructed tree with each node's splitting criteria and class labels at the leaf nodes. The tree diagram helps in understanding how the Decision Tree makes decisions based on the input features.

Let's proceed with using Python to develop the Decision Tree and visualize it using tree.plot_tree.

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
```

Listing 1: Header

Problem 3. *The objective function of the K-Means clustering algorithm aims to find optimal cluster centers (centroids) that minimize the sum of squared distances between data points and their corresponding centroids. Mathematically, the objective function can be defined as:*

$$J = \arg \min_{\{w_{ij}\},\{\mathbf{c}_j\}} \sum_{i=1}^{n} \sum_{j=1}^{K} w_{ij} \cdot \|\mathbf{p}_i - \mathbf{c}_j\|^2$$

Use the following Data points and Centroids along with the objective function to yield the minimum possible value. (in other words, what is the best value of J)

Data Points:

$$\mathbf{p}_1 = 2$$
$$\mathbf{p}_2 = 4$$
$$\mathbf{p}_3 = 7$$
$$\mathbf{p}_4 = 10$$

Initial Centroids:

$$\mathbf{c}_1 = 3$$
$$\mathbf{c}_2 = 8$$

Problem 4. *Develop the following algorithm using Python, Choose a dataset of interest. for example, you can use the popular Iris dataset, Visualize the results by plotting the data points with different colors representing the clusters, and mark the centroids of the clusters. Set the number of clusters (K) for K-Means = 3.*

Data: Dataset $\mathbf{X}$ with n data points
Result: Cluster centroids $\mathbf{C}$ and data point assignments
Initialize: Choose the number of clusters K and randomly select K data points as the initial centroids $\mathbf{C}$
Repeat:
while *not converged* **do**
  for $i = 1$ *to* n **do**
    Assign data point $\mathbf{p}_i$ to the nearest centroid $\mathbf{c}_j$:

$$j = \arg\min_j \|\mathbf{p}_i - \mathbf{c}_j\|^2$$

  end
  for $j = 1$ *to* K **do**
    Update centroid $\mathbf{c}_j$ by computing the mean of data points assigned to it:

$$\mathbf{c}_j = \frac{1}{n_j} \sum_{i=1}^{n} w_{ij} \cdot \mathbf{p}_i$$

    where n_j is the number of data points assigned to centroid $\mathbf{c}_j$.
  end
end
Output: The final centroids $\mathbf{C}$ represent the cluster centers, and the data points are assigned to the nearest centroids, forming K clusters

Algorithm 1: K-Means Algorithm