MACHINE LEARNING LAB

EXERCISE 4

Aim :

1. Use the classification.csv file and compute the gini index for age and salary column.
2. Create decision tree algorithm from scratch without using sklearn library. you may assume that all the columns in the data will be categorical in nature. Give a new data for prediction and print the predicted output along with the probabilities.

Algorithm 1 :

1. **Load and understand the data**
2. **Define Gini Index Function**:
   - Take a column of values as input.
   - Calculate the total number of samples in the column.
   - Count the occurrences of each unique value in the column.
   - Calculate the probability of each class by dividing the count of each class by the total number of samples.
   - Compute the Gini index using the formula: (\text{Gini index} = 1 - \sum (\text{probability of each class}^2)).
   - Return the computed Gini index.
3. **Compute Gini Index for 'Age' and 'EstimatedSalary' Columns**:
   - Apply the Gini index function to both the 'Age' and 'EstimatedSalary' columns.
4. **Understanding Gini Index**:
   - Insights into the impurity or inequality of each feature in the dataset: The Gini index ranges from 0 to 1, with different interpretations at these extremes:

- **Closer to 0**: A Gini index closer to 0 indicates low impurity or inequality in the dataset. It means that the samples are predominantly of the same class or category. For example, if all samples in a node belong to the same class, the Gini index would be 0, indicating perfect purity.
- **Closer to 1**: A Gini index closer to 1 indicates high impurity or inequality in the dataset. It means that the samples are evenly distributed among different classes or categories. For example, if the samples are evenly distributed across multiple classes in a node, the Gini index would approach 1, indicating high impurity.

In summary, a lower Gini index signifies better separation of classes or categories in a decision tree node, while a higher Gini index suggests a more mixed distribution of classes, leading to higher impurity.

Algorithm 2 :

1. **Define the Node Structure**: Define a class to represent nodes in the decision tree. Each node contains information such as the feature to split on, the threshold value for splitting, references to left and right child nodes, and the predicted class if the node is a leaf.
2. **Grow the Tree Recursively**:
   - Start with the root node.
   - For each node:
     - Calculate the impurity measure (e.g., Gini index or entropy) for each possible split.
     - Select the split that maximally reduces impurity.
     - Split the dataset into left and right subsets based on the selected feature and threshold.
     - Recursively grow the left and right subtrees.
3. **Stopping Criteria**:
   - Define stopping criteria to prevent overfitting, such as:
     - Maximum tree depth.
     - Minimum number of samples required to split a node.
     - Minimum number of samples required to be at a leaf node.
     - Maximum number of leaf nodes.
4. **Splitting Criteria**:
   - Choose an impurity measure (e.g., Gini index or entropy) to evaluate splits.
   - For each feature, evaluate all possible thresholds to find the one that minimizes impurity.
5. **Prediction**:
   - Once the tree is grown, make predictions for new samples by traversing the tree from the root node to a leaf node.
   - At each node, compare the feature value of the sample to the threshold value stored in the node.
   - Move to the left or right child node based on the comparison.
   - Repeat until reaching a leaf node, which contains the predicted class.

1. Code and Output :

In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
```

```
C:\Users\TEJU\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A NumPy
version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version
1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:

```python
data=pd.read_csv(r"C:\Users\TEJU\Downloads\classification.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              400 non-null    int64
 1   EstimatedSalary  400 non-null    int64
 2   Purchased        400 non-null    int64
dtypes: int64(3)
memory usage: 9.5 KB
```

In [12]:
```
#Let us now compute gini index
def gini_index(column_values):
    total_samples=len(column_values)
    value_counts = column_values.value_counts()
    probabilities = value_counts / total_samples
    gini_index = 1 - sum(probabilities ** 2)
    return gini_index
```

In [13]:
```
gini_age = gini_index(data['Age'])
print("Gini index for Age:", gini_age)
```

```
Gini index for Age: 0.9681124999999999
```

In [14]:
```
gini_salary = gini_index(data['EstimatedSalary'])
print("Gini index for EstimatedSalary:", gini_salary)
```

```
Gini index for EstimatedSalary: 0.9875
```

In [15]:
```
#let us try to build a decision tree using sklearn for this dataset
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [16]:
```
X = data[['Age','EstimatedSalary']]
y = data['Purchased']
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_stat
```

In [18]:
```
#by default, takes gini criterion
model=DecisionTreeClassifier(random_state=42)
```

In [19]:
```
model.fit(X_train,y_train)
```

Out[19]:
```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

In [21]:
```
y_pred=model.predict(X_test)
```

In [22]:
```
accuracy=accuracy_score(y_test,y_pred)
print("Accuracy of decision tree is:",accuracy)
```

```
Accuracy of decision tree is: 0.85
```

In [23]:
```
print("Classification Report:",classification_report(y_test,y_pred))
print("Confusion Matrix:",confusion_matrix(y_test,y_pred))
```

```
Classification Report:               precision    recall  f1-score   support

           0       0.88      0.88      0.88        73
           1       0.81      0.81      0.81        47

    accuracy                           0.85       120
   macro avg       0.84      0.84      0.84       120
weighted avg       0.85      0.85      0.85       120

Confusion Matrix: [[64  9]
 [ 9 38]]
```

2. Code and Output :

In [32]:
```
from collections import Counter
```

In [34]:
```
from sklearn.datasets import load_iris
iris=load_iris()
X=np.array(iris.data)
Y=np.array(iris.target)
```

In [35]:
```
class Node:
    def __init__(self, feature_index=None, threshold=None, left=None, right=None, va
        self.feature_index = feature_index  # Index of the feature to split on
        self.threshold = threshold  # Threshold value for the feature
        self.left = left  # Left child (subtree)
        self.right = right  # Right child (subtree)
        self.value = value  # Predicted class if the node is a leaf

# Define the DecisionTree class
class DecisionTree:
    def __init__(self, max_depth=None):
        self.max_depth = max_depth  # Maximum depth of the tree
        self.root = None  # Root node of the tree
    def fit(self, X, y):
        self.root = self._grow_tree(X, y, depth=0)

    def _grow_tree(self, X, y, depth):
        # Stopping criteria
        if depth == self.max_depth or len(np.unique(y)) == 1:
            return Node(value=np.argmax(np.bincount(y)))

        n_samples, n_features = X.shape
        best_gini = np.inf
        best_feature_index = None
        best_threshold = None

        # Calculate Gini index for each feature and threshold
        for feature_index in range(n_features):
            thresholds = np.unique(X[:, feature_index])
            for threshold in thresholds:
                left_indices = np.where(X[:, feature_index] <= threshold)[0]
                right_indices = np.where(X[:, feature_index] > threshold)[0]

                gini = self._gini_index(y[left_indices], y[right_indices])
                if gini < best_gini:
                    best_gini = gini
                    best_feature_index = feature_index
                    best_threshold = threshold

        # Split the dataset based on the best feature and threshold
        left_indices = np.where(X[:, best_feature_index] <= best_threshold)[0]
        right_indices = np.where(X[:, best_feature_index] > best_threshold)[0]

        # Recursively grow the left and right subtrees
        left = self._grow_tree(X[left_indices], y[left_indices], depth + 1)
        right = self._grow_tree(X[right_indices], y[right_indices], depth + 1)

        return Node(feature_index=best_feature_index, threshold=best_threshold, left

    def _gini_index(self, *groups):
        # Calculate Gini index for a split
        total_samples = sum(len(group) for group in groups)
        gini = 0.0
        for group in groups:
            size = float(len(group))
            if size == 0:
                continue
            score = 0.0
            for class_val in np.unique(group):
                p = (np.sum(group == class_val) / size)
                score += p * p
            gini += (1.0 - score) * (size / total_samples)
        return gini

    def predict(self, X):
        # Make predictions for each sample in X
        return np.array([self._predict(sample, self.root) for sample in X])

    def _predict(self, sample, node):
        # Recursively traverse the tree to make predictions
        if node.value is not None:
            return node.value
        if sample[node.feature_index] <= node.threshold:
            return self._predict(sample, node.left)
        else:
            return self._predict(sample, node.right)
```

In [37]:
```
# Initialize and train the decision tree model
tree = DecisionTree(max_depth=3)
tree.fit(X, Y)
```

In [40]:
```
# Make predictions and print the predicted output along with probabilities
predictions = tree.predict(X)
for i, pred in enumerate(predictions):
    probabilities = np.bincount(tree.predict(X[i].reshape(1, -1)).flatten().astype(i
    print(f"Sample {i + 1}: Predicted Class: {pred}, Probabilities: {probabilities}"
```

```
Sample 1: Predicted Class: 0, Probabilities: [1.]
Sample 2: Predicted Class: 0, Probabilities: [1.]
Sample 3: Predicted Class: 0, Probabilities: [1.]
Sample 4: Predicted Class: 0, Probabilities: [1.]
Sample 5: Predicted Class: 0, Probabilities: [1.]
Sample 6: Predicted Class: 0, Probabilities: [1.]
Sample 7: Predicted Class: 0, Probabilities: [1.]
Sample 8: Predicted Class: 0, Probabilities: [1.]
Sample 9: Predicted Class: 0, Probabilities: [1.]
Sample 10: Predicted Class: 0, Probabilities: [1.]
Sample 11: Predicted Class: 0, Probabilities: [1.]
Sample 12: Predicted Class: 0, Probabilities: [1.]
Sample 13: Predicted Class: 0, Probabilities: [1.]
Sample 14: Predicted Class: 0, Probabilities: [1.]
Sample 15: Predicted Class: 0, Probabilities: [1.]
Sample 16: Predicted Class: 0, Probabilities: [1.]
Sample 17: Predicted Class: 0, Probabilities: [1.]
Sample 18: Predicted Class: 0, Probabilities: [1.]
Sample 19: Predicted Class: 0, Probabilities: [1.]
Sample 20: Predicted Class: 0, Probabilities: [1.]
Sample 21: Predicted Class: 0, Probabilities: [1.]
Sample 22: Predicted Class: 0, Probabilities: [1.]
Sample 23: Predicted Class: 0, Probabilities: [1.]
Sample 24: Predicted Class: 0, Probabilities: [1.]
Sample 25: Predicted Class: 0, Probabilities: [1.]
Sample 26: Predicted Class: 0, Probabilities: [1.]
Sample 27: Predicted Class: 0, Probabilities: [1.]
Sample 28: Predicted Class: 0, Probabilities: [1.]
Sample 29: Predicted Class: 0, Probabilities: [1.]
Sample 30: Predicted Class: 0, Probabilities: [1.]
Sample 31: Predicted Class: 0, Probabilities: [1.]
Sample 32: Predicted Class: 0, Probabilities: [1.]
Sample 33: Predicted Class: 0, Probabilities: [1.]
Sample 34: Predicted Class: 0, Probabilities: [1.]
Sample 35: Predicted Class: 0, Probabilities: [1.]
Sample 36: Predicted Class: 0, Probabilities: [1.]
Sample 37: Predicted Class: 0, Probabilities: [1.]
Sample 38: Predicted Class: 0, Probabilities: [1.]
Sample 39: Predicted Class: 0, Probabilities: [1.]
Sample 40: Predicted Class: 0, Probabilities: [1.]
Sample 41: Predicted Class: 0, Probabilities: [1.]
Sample 42: Predicted Class: 0, Probabilities: [1.]
Sample 43: Predicted Class: 0, Probabilities: [1.]
Sample 44: Predicted Class: 0, Probabilities: [1.]
Sample 45: Predicted Class: 0, Probabilities: [1.]
Sample 46: Predicted Class: 0, Probabilities: [1.]
Sample 47: Predicted Class: 0, Probabilities: [1.]
Sample 48: Predicted Class: 0, Probabilities: [1.]
Sample 49: Predicted Class: 0, Probabilities: [1.]
Sample 50: Predicted Class: 0, Probabilities: [1.]
Sample 51: Predicted Class: 1, Probabilities: [0. 1.]
Sample 52: Predicted Class: 1, Probabilities: [0. 1.]
Sample 53: Predicted Class: 1, Probabilities: [0. 1.]
Sample 54: Predicted Class: 1, Probabilities: [0. 1.]
Sample 55: Predicted Class: 1, Probabilities: [0. 1.]
Sample 56: Predicted Class: 1, Probabilities: [0. 1.]
Sample 57: Predicted Class: 1, Probabilities: [0. 1.]
Sample 58: Predicted Class: 1, Probabilities: [0. 1.]
Sample 59: Predicted Class: 1, Probabilities: [0. 1.]
Sample 60: Predicted Class: 1, Probabilities: [0. 1.]
Sample 61: Predicted Class: 1, Probabilities: [0. 1.]
Sample 62: Predicted Class: 1, Probabilities: [0. 1.]
Sample 63: Predicted Class: 1, Probabilities: [0. 1.]
Sample 64: Predicted Class: 1, Probabilities: [0. 1.]
Sample 65: Predicted Class: 1, Probabilities: [0. 1.]
Sample 66: Predicted Class: 1, Probabilities: [0. 1.]
Sample 67: Predicted Class: 1, Probabilities: [0. 1.]
Sample 68: Predicted Class: 1, Probabilities: [0. 1.]
Sample 69: Predicted Class: 1, Probabilities: [0. 1.]
Sample 70: Predicted Class: 1, Probabilities: [0. 1.]
Sample 71: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 72: Predicted Class: 1, Probabilities: [0. 1.]
Sample 73: Predicted Class: 1, Probabilities: [0. 1.]
Sample 74: Predicted Class: 1, Probabilities: [0. 1.]
Sample 75: Predicted Class: 1, Probabilities: [0. 1.]
Sample 76: Predicted Class: 1, Probabilities: [0. 1.]
Sample 77: Predicted Class: 1, Probabilities: [0. 1.]
Sample 78: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 79: Predicted Class: 1, Probabilities: [0. 1.]
Sample 80: Predicted Class: 1, Probabilities: [0. 1.]
Sample 81: Predicted Class: 1, Probabilities: [0. 1.]
Sample 82: Predicted Class: 1, Probabilities: [0. 1.]
Sample 83: Predicted Class: 1, Probabilities: [0. 1.]
Sample 84: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 85: Predicted Class: 1, Probabilities: [0. 1.]
Sample 86: Predicted Class: 1, Probabilities: [0. 1.]
Sample 87: Predicted Class: 1, Probabilities: [0. 1.]
Sample 88: Predicted Class: 1, Probabilities: [0. 1.]
Sample 89: Predicted Class: 1, Probabilities: [0. 1.]
Sample 90: Predicted Class: 1, Probabilities: [0. 1.]
Sample 91: Predicted Class: 1, Probabilities: [0. 1.]
Sample 92: Predicted Class: 1, Probabilities: [0. 1.]
Sample 93: Predicted Class: 1, Probabilities: [0. 1.]
Sample 94: Predicted Class: 1, Probabilities: [0. 1.]
Sample 95: Predicted Class: 1, Probabilities: [0. 1.]
Sample 96: Predicted Class: 1, Probabilities: [0. 1.]
Sample 97: Predicted Class: 1, Probabilities: [0. 1.]
Sample 98: Predicted Class: 1, Probabilities: [0. 1.]
Sample 99: Predicted Class: 1, Probabilities: [0. 1.]
Sample 100: Predicted Class: 1, Probabilities: [0. 1.]
Sample 101: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 102: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 103: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 104: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 105: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 106: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 107: Predicted Class: 1, Probabilities: [0. 1.]
Sample 108: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 109: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 110: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 111: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 112: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 113: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 114: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 115: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 116: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 117: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 118: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 119: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 120: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 121: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 122: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 123: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 124: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 125: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 126: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 127: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 128: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 129: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 130: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 131: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 132: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 133: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 134: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 135: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 136: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 137: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 138: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 139: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 140: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 141: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 142: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 143: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 144: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 145: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 146: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 147: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 148: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 149: Predicted Class: 2, Probabilities: [0. 0. 1.]
Sample 150: Predicted Class: 2, Probabilities: [0. 0. 1.]
```

Result :

Therefore, we were successfully able to calculate the gini index for two columns in classification.csv and create a decision tree using the sklearn library.

We were also able to create a decision tree from scratch and predict the class and probabilities for another dataset.