step-by-step explanation of the Decision Tree Classifier code for the "Buy a Computer" prediction:

### 🔶 1. **Import Libraries**

python

CopyEdit

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn import tree

import matplotlib.pyplot as plt

* pandas: Used to create and manage tabular data.
* DecisionTreeClassifier: The machine learning model used here.
* LabelEncoder: Converts categorical (string) values into numeric values.
* tree: For visualizing the decision tree.
* matplotlib.pyplot: For plotting the tree graphically.

### 🔶 2. **Create Dataset**

python

CopyEdit

data = {

'Age': ['Youth', 'Youth', 'Middle', 'Senior', ...],

...

}

df = pd.DataFrame(data)

* A dictionary represents the dataset.
* It’s then converted into a DataFrame, which is a tabular format similar to a spreadsheet.
* Each row is a person, and columns are their features like Age, Income, etc.

### 🔶 3. **Encode Categorical Variables**

python

CopyEdit

le = LabelEncoder()

for column in df.columns:

df[column] = le.fit\_transform(df[column])

* Machine learning models can't handle text, so we convert string values (like "Youth", "High") into numbers.
* LabelEncoder maps each unique category to a number.
  + Example: "Youth" → 2, "Senior" → 1, "Middle" → 0

### 🔶 4. **Split Features and Target**

python

CopyEdit

X = df.drop('Buys\_Computer', axis=1) # Features

y = df['Buys\_Computer'] # Target

* X contains all input columns (independent variables).
* y contains the target column (Buys\_Computer) — this is what we’re trying to predict.

### 🔶 5. **Train the Model**

python

CopyEdit

model = DecisionTreeClassifier(criterion='entropy')

model.fit(X, y)

* Creates a decision tree model using entropy (a measure of information gain).
* fit(X, y) trains the model using the input data and the target output.

### 🔶 6. **Visualize the Tree**

python

CopyEdit

plt.figure(figsize=(10, 6))

tree.plot\_tree(model, feature\_names=X.columns, class\_names=['No', 'Yes'], filled=True)

plt.show()

* This block draws the decision tree:
  + feature\_names: shows which feature is used at each split.
  + class\_names: labels the final decisions.
  + filled=True: colors the tree nodes based on class.
* The result is a visual diagram showing how decisions are made.

### ✅ Summary:

* We built a decision tree from categorical data.
* Encoded it into numeric values.
* Trained the model to classify if someone would buy a computer.
* Visualized the logic in a tree format.