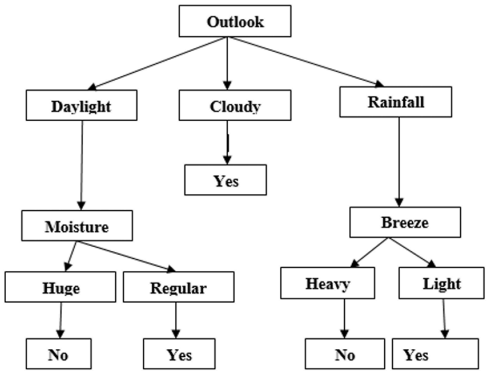
Open in Colab

Introduction to Classification using Decision Trees

A **decision tree** is a popular supervised learning algorithm used for both classification and regression tasks. In classification, the goal is to predict the class label of an observation by splitting the data based on feature values in a tree-like structure.

- **Root Node**: The first node in the decision tree.
- **Internal Nodes**: Nodes that represent decisions based on features.
- **Leaf Nodes**: Terminal nodes that contain the predicted class.

The decision tree makes predictions by following a path from the root to a leaf node.



Day	Outlook	Temperature	Moisture	Breeze	Play Game
D1	Daylight	Hot	Huge	Light	No
D2	Daylight	Hot	Huge	Heavy	No
D3	Cloudy	Hot	Huge	Light	Yes
D4	Rainfall	Warm	Huge	Light	Yes
D5	Rainfall	Cold	Regular	Light	Yes
D6	Rainfall	Cold	Regular	Heavy	No
D7	Cloudy	Cold	Regular	Heavy	Yes
D8	Daylight	Warm	Huge	Light	No
D9	Daylight	Cold	Regular	Light	Yes
D10	Rainfall	Warm	Regular	Light	Yes
D11	Daylight	Warm	Regular	Heavy	Yes
D12	Cloudy	Warm	Huge	Heavy	Yes
D13	Cloudy	Hot	Regular	Light	Yes
D14	Rainfall	Warm	Huge	Heavy	No

The Gini Index for Splitting

The **Gini index** is a measure of impurity or impurity reduction used in decision trees to determine the best feature to split at a node. Here's an example to clarify its working.

Example: Loan Default Dataset

Data:

Age Group	Loan Default (Target)
Young	Yes
Young	No

Age Group	Loan Default (Target)
Middle	No
Middle	No
Old	No
Old	Yes

Step 1: Calculate the Gini Index for the Root Node

The root node contains all the data. The formula for Gini Index is:

$G = 1 - \sum_{i=1}^{c} p_i^2$

Where:

- (c) = number of classes (Yes and No in this case),
- (p_i) = proportion of instances in class (i).
- Total Instances = 6
- Default = Yes: 2, Default = No: 4

$G_{\text{root}} = 1 - \left(\left(\frac{2}{6}\right)^2 + \left(\frac{4}{6}\right)^2 \right) = 1 - (0.111 + 0.444) = 0.445$

Step 2: Evaluate Splits Based on "Age Group"

Let's split the dataset by the `Age Group` feature into three groups: `Young`, `Middle`, and `Old`.

Split Results:

- **Group 1: Young**
  - Instances: 2 (Yes: 1, No: 1)
  - Gini: $G_{\text{Young}} = 1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 \right) = 1 - (0.25 + 0.25) = 0.5$
- **Group 2: Middle**
  - Instances: 2 (Yes: 0, No: 2)
  - Gini: $G_{\text{Middle}} = 1 - \left(\left(\frac{0}{2}\right)^2 + \left(\frac{2}{2}\right)^2 \right) = 1 - (0 + 1) = 0$
- **Group 3: Old**
  - Instances: 2 (Yes: 1, No: 1)

- Gini: $G_{\text{Old}} = 1 - \left(\left(\frac{1}{2} \right)^2 + \left(\frac{1}{2} \right)^2 \right) = 1 - (0.25 + 0.25) = 0.5$

Step 3: Calculate the Weighted Gini for the Split

To determine the overall Gini Index after the split, we calculate the weighted average:

$$G_{\text{split}} = \frac{n_{\text{Young}}}{n} G_{\text{Young}} + \frac{n_{\text{Middle}}}{n} G_{\text{Middle}} + \frac{n_{\text{Old}}}{n} G_{\text{Old}}$$

Where (n_{\text{Group}}) is the number of instances in each group.

$$G_{\text{split}} = \frac{2}{6}(0.5) + \frac{2}{6}(0) + \frac{2}{6}(0.5) = 0.167 + 0 + 0.167 = 0.334$$

Step 4: Calculate Gini Gain (Impurity Reduction)

The **Gini Gain** is the reduction in impurity achieved by the split:

$$\text{Gini Gain} = G_{\text{root}} - G_{\text{split}}$$

$$\text{Gini Gain} = 0.445 - 0.334 = 0.111$$

Interpretation

- **The feature "Age Group" reduces the impurity from 0.445 to 0.334**, achieving a Gini Gain of 0.111.
- The Decision Tree algorithm would use this reduction (or compare it with other features) to decide whether to split on `Age Group`.

This process is repeated for all features, and the split that maximizes impurity reduction is chosen. This ensures that the tree grows in a way that creates the purest possible nodes at each step.

Step-by-Step Example of Gini Calculation

Consider the following dataset:

Feature	Class
1.5	A
2.0	A
2.5	B
3.0	B
3.5	B

We want to split this dataset based on the feature value at 2.5:

- **Before Split**:
  Gini Index for the entire dataset:
  - 2 out of 5 are Class A → $p = \frac{2}{5} = 0.4$
  - 3 out of 5 are Class B → $1 - p = 0.6$

$$Gini = 1 - (0.4^2 + 0.6^2) = 1 - (0.16 + 0.36) = 1 - 0.52 = 0.48$$

- **After Split**:
  - Left node (Feature < 2.5): (p = 1/2 = 0.5), (1 - p = 0.5)

$$Gini = 1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 1 - 0.5 = 0.5$$

  - Right node (Feature ≥ 2.5): (p = 1/3 = 0.33), (1 - p = 0.67)

$$Gini = 1 - (0.33^2 + 0.67^2) = 1 - (0.11 + 0.45) = 1 - 0.56 = 0.44$$

The Gini Index for the split is a weighted sum of the Gini Indexes of the child nodes.

Python Code Example Using the Gini Index

Here's an implementation of a decision tree using the Gini Index in Python with the `scikit-learn` library:

In [1]:
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt

# Load the dataset
url = "https://raw.githubusercontent.com/msfasha/307304-Data-Mining/main/datasets/l
data = pd.read_csv(url)

# Separate features and target variable
X = data.drop('loan_default', axis=1)  # Features
y = data['loan_default']               # Target variable

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta

# Initialize the Decision Tree Classifier with Gini index
model = DecisionTreeClassifier(criterion='gini', random_state=42)

# Train the model
model.fit(X_train, y_train)
```

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

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Plot the Decision Tree
plt.figure(figsize=(20, 10))
plot_tree(model, feature_names=X.columns, class_names=['No Default', 'Default'], fi
plt.title("Decision Tree for Loan Default Prediction")
plt.savefig("decision_tree_plot.png", format="png", dpi=300, bbox_inches='tight')
plt.show()
```

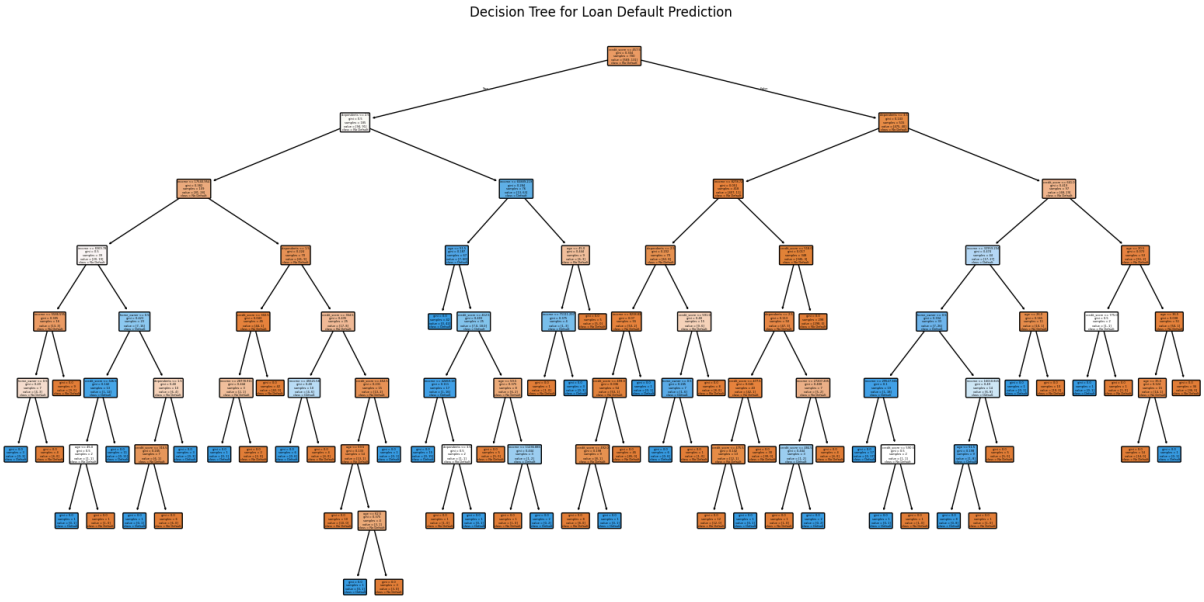
```
Accuracy: 0.86

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.92      0.91       236
           1       0.68      0.66      0.67        64

    accuracy                           0.86       300
   macro avg       0.79      0.79      0.79       300
weighted avg       0.86      0.86      0.86       300


Confusion Matrix:
 [[216  20]
 [ 22  42]]
```



Explanation:

- **Step 1**: We create a small dataset with a single feature and class labels.
- **Step 2**: The features (X) are the feature values, while the target (y) is the class labels.

- **Step 3**: We split the dataset into training and testing sets.
- **Step 4**: We initialize a decision tree classifier with the Gini Index as the criterion.
- **Step 5**: We train the model using the training set.
- **Step 6**: We use the trained model to make predictions on the test set.
- **Step 7**: We evaluate the model's accuracy.
- **Step 8**: We plot the decision tree for visualization.
