

- Study Gini Index
- Compute the toy example using Gini Index
- Change criterion in the imported library, using Gini Index
- Compare Gini Index vs Entropy
- Use another dataset (data.csv)
- Play with parameters:

max_depth min_samples_split min_samples_leaf

- Explain your understanding after trying these different parameters

Step 1: Study Gini Index

What is the Gini Index?

The **Gini Index** (or **Gini Impurity**) is a measure used in decision tree algorithms to evaluate the **impurity** of a dataset. It determines how mixed the classes are within a node.

Formula:

Mathematically, The Gini Index is represented by

$$Gini\ impurity = 1 - \sum (p(i)^2)$$

Another commonly used formula is:

$$Gini\ impurity = 1 - \sum (p(i) * (1 - p(i)))$$

Gini Gain (สนใจ) = Gini Impurity (ทั้งหมด) - GiniImpurity (สนใจ)

Step 2: Compute the toy example using Gini Index

```
In [2]: # Write your code here
import pandas as pd
df_toy = pd.read_csv('toy_data.csv')
df_toy
```

Out[2]:

	age	income	student	credit rating	buys computer
0	<=30	high	no	fair	no
1	<=30	high	no	excellent	no
2	31-40	high	no	fair	yes
3	>40	medium	no	fair	yes
4	>40	low	yes	fair	yes
5	>40	low	yes	excellent	no
6	31-40	low	yes	excellent	yes
7	<=30	medium	no	fair	no
8	<=30	low	yes	fair	yes
9	>40	medium	yes	fair	yes
10	<=30	medium	yes	excellent	yes
11	31-40	medium	no	excellent	yes
12	31-40	high	yes	fair	yes
13	>40	medium	no	excellent	no

In [4]: `df_toy.isnull().sum()`

```
Out[4]: age          0
income        0
student       0
credit rating  0
buys computer 0
dtype: int64
```

In [15]: `import pandas as pd`

```
def gini_impurity(series):
    """ คำนวณ Gini Impurity = Gini Index ของ series """
    probs = series.value_counts(normalize=True) # คำนวณอัตราส่วนของแต่ละ class
    return 1 - sum(probs ** 2) # ใช้สูตร Gini Impurity

def gini_gain(df, feature, target):
    """ คำนวณ Gini Gain ของ feature เทียบกับ target """
    gini_parent = gini_impurity(df[target]) # Gini ของชุดข้อมูลหลัก
    weighted_gini = sum(
        (len(subset) / len(df)) * gini_impurity(subset[target])
        for _, subset in df.groupby(feature) # แบ่งข้อมูลตาม feature
    )
    return gini_parent - weighted_gini # คำนวณ Gini Gain

# คำนวณ Gini Impurity ของ target
gini_value = gini_impurity(df_toy['buys computer'])
print(f"Gini Impurity : {gini_value:.4f}")

# คำนวณ Gini Gain ของทุก Feature
```

```
for feature in ['age', 'income', 'student', 'credit rating']:
    print(f"Gini Gain for {feature}: {gini_gain(df_toy, feature, 'buys computer')}
```

Gini Impurity : 0.4592

Gini Gain for age: 0.1163

Gini Gain for income: 0.0187

Gini Gain for student: 0.0918

Gini Gain for credit rating: 0.0306

Step3: Change criterion in the imported library, using Gini Index

```
In [18]: from sklearn.preprocessing import LabelEncoder

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply Label Encoding for all categorical columns
df_toy['age'] = label_encoder.fit_transform(df_toy['age'])
df_toy['income'] = label_encoder.fit_transform(df_toy['income'])
df_toy['student'] = label_encoder.fit_transform(df_toy['student'])
df_toy['credit rating'] = label_encoder.fit_transform(df_toy['credit rating'])
df_toy['buys computer'] = label_encoder.fit_transform(df_toy['buys computer'])

df_toy
```

```
Out[18]:
```

	age	income	student	credit rating	buys computer
0	1	0	0	1	0
1	1	0	0	0	0
2	0	0	0	1	1
3	2	2	0	1	1
4	2	1	1	1	1
5	2	1	1	0	0
6	0	1	1	0	1
7	1	2	0	1	0
8	1	1	1	1	1
9	2	2	1	1	1
10	1	2	1	0	1
11	0	2	0	0	1
12	0	0	1	1	1
13	2	2	0	0	0

```
In [19]: x = df_toy.drop('buys computer', axis=1) #features
y = df_toy['buys computer'] #label
```

```
In [20]: import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

```
from sklearn.tree import DecisionTreeClassifier, plot_tree

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_

clf = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42)
clf.fit(x_train, y_train)
```

Out[20]:

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=42)
```

```
In [24]: print(x_train.shape)
         print(x_test.shape)
```

(11, 4)

(3, 4)

```
In [27]: from sklearn.tree import DecisionTreeClassifier
         # Initialize the Decision Tree classifier
         clf = DecisionTreeClassifier(criterion='entropy', random_state=42) # Using 'ent
         # Train the model
         clf.fit(x_train, y_train)
         # Predict on the test set
         y_pred = clf.predict(x_test)
```

```
In [29]: from sklearn.metrics import accuracy_score, classification_report, confusion_mat

         # Calculate accuracy
         accuracy = accuracy_score(y_test, y_pred)
         print(f"Accuracy: {accuracy:.2f}")

         # Classification report
         print("Classification Report:")
         print(classification_report(y_test, y_pred))

         # Confusion Matrix
         print("Confusion Matrix:")
         print(confusion_matrix(y_test, y_pred))
```

Accuracy: 1.00

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
1	1.00	1.00	1.00	2
accuracy			1.00	3
macro avg	1.00	1.00	1.00	3
weighted avg	1.00	1.00	1.00	3

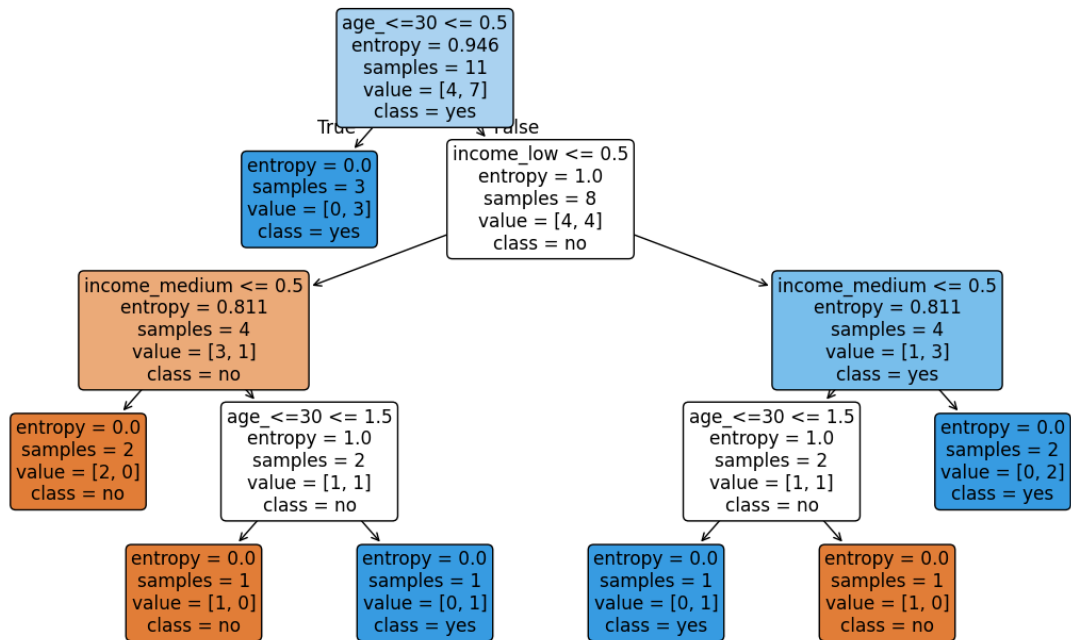
Confusion Matrix:

```
[[1 0]
 [0 2]]
```

```
In [32]: import matplotlib.pyplot as plt
         from sklearn.tree import plot_tree

         # Plot the decision tree
         plt.figure(figsize=(14, 8))
```

```
plot_tree(clf, filled=True, feature_names=X.columns, class_names=['no', 'yes'],
plt.show())
```



Step 4. Compare the Gini and Entropy criterion decision tree

```
In [45]: clf_tuned = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42)
clf_tuned.fit(x_train, y_train)
```

```
Out[45]: ▼ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=42)
```

```
In [46]: clf_entropy = DecisionTreeClassifier(criterion='entropy', random_state=42)
clf_entropy.fit(x_train, y_train)
```

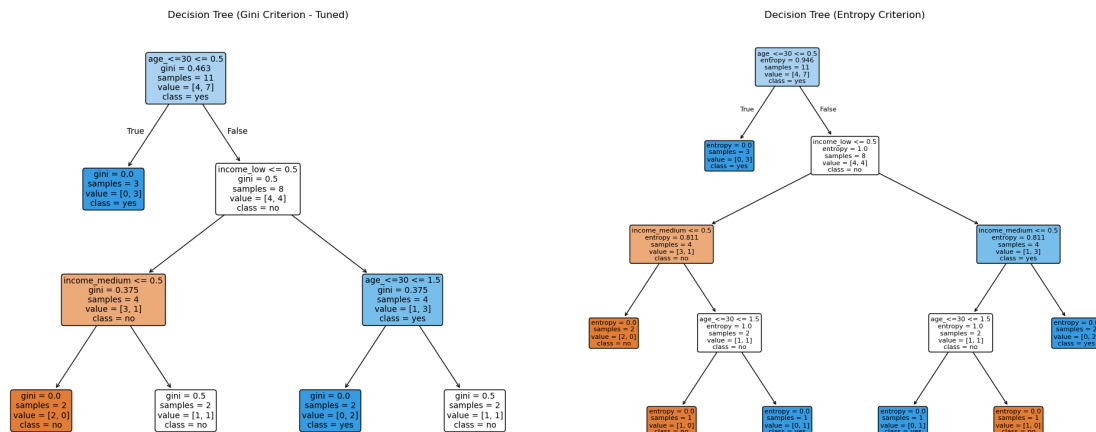
```
Out[46]: ▼ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

```
In [50]: fig, axs = plt.subplots(1, 2, figsize=(20, 8))

# Plot the tuned Gini decision tree
plot_tree(clf_tuned, filled=True, feature_names=X.columns, class_names=['no', 'y
axs[0].set_title("Decision Tree (Gini Criterion - Tuned)")

# Plot the entropy decision tree
plot_tree(clf_entropy, filled=True, feature_names=X.columns, class_names=['no',
axs[1].set_title("Decision Tree (Entropy Criterion)")

plt.tight_layout()
plt.show()
```



Step 5: Use another dataset (data.csv)

```
In [51]: # Write your code here
df = pd.read_csv('dataset.csv')
df
```

```
Out[51]:
```

	feature_0	feature_1	feature_2	feature_3	feature_4	target
0	0.374540	0.950714	0.731994	0.598658	0.156019	0
1	0.155995	0.058084	0.866176	0.601115	0.708073	1
2	0.020584	0.969910	0.832443	0.212339	0.181825	1
3	0.183405	0.304242	0.524756	0.431945	0.291229	0
4	0.611853	0.139494	0.292145	0.366362	0.456070	1
...
145	0.841829	0.139772	0.795267	0.201627	0.163656	1
146	0.164266	0.814575	0.665197	0.523065	0.358830	1
147	0.877201	0.392445	0.816599	0.439135	0.376944	1
148	0.462680	0.301378	0.747609	0.502720	0.232213	0
149	0.899575	0.383891	0.543553	0.906472	0.624238	1

150 rows × 6 columns

```
In [3]: df.isnull().sum()
```

```
Out[3]: feature_0    0
feature_1    0
feature_2    0
feature_3    0
feature_4    0
target      0
dtype: int64
```

```
In [57]: df
```

Out[57]:

	feature_0	feature_1	feature_2	feature_3	feature_4	target
0	0.374540	0.950714	0.731994	0.598658	0.156019	0
1	0.155995	0.058084	0.866176	0.601115	0.708073	1
2	0.020584	0.969910	0.832443	0.212339	0.181825	1
3	0.183405	0.304242	0.524756	0.431945	0.291229	0
4	0.611853	0.139494	0.292145	0.366362	0.456070	1
...
145	0.841829	0.139772	0.795267	0.201627	0.163656	1
146	0.164266	0.814575	0.665197	0.523065	0.358830	1
147	0.877201	0.392445	0.816599	0.439135	0.376944	1
148	0.462680	0.301378	0.747609	0.502720	0.232213	0
149	0.899575	0.383891	0.543553	0.906472	0.624238	1

150 rows × 6 columns

```
In [58]: from sklearn.model_selection import train_test_split

# แยก Features (X) และ Labels (y)
x = df.drop(columns=['target'])
y = df['target']

# แบ่งข้อมูลเป็น train (80%) และ test (20%)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_

# Create and train a Decision Tree model with entropy criterion
clf = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
clf.fit(x_train, y_train)
```

Out[58]:

DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)

```
In [59]: print(x_train.shape)
print(x_test.shape)
```

(120, 5)

(30, 5)

```
In [60]: from sklearn.tree import DecisionTreeClassifier

# Initialize the Decision Tree classifier
clf = DecisionTreeClassifier(criterion='entropy', random_state=42) # Using 'ent

# Train the model
clf.fit(x_train, y_train)
```

```
# Predict on the test set
y_pred = clf.predict(x_test)
```

In [61]: `from sklearn.metrics import accuracy_score, classification_report, confusion_mat`

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Accuracy: 0.67

Classification Report:

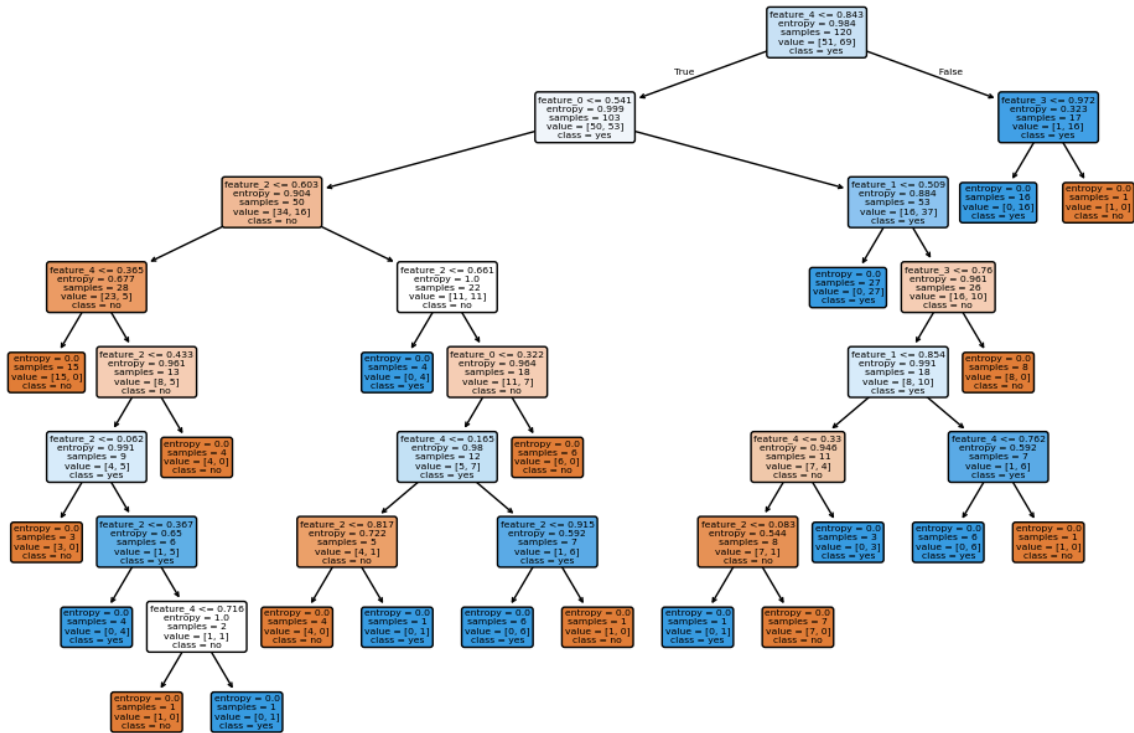
	precision	recall	f1-score	support
0	0.62	0.42	0.50	12
1	0.68	0.83	0.75	18
accuracy			0.67	30
macro avg	0.65	0.62	0.62	30
weighted avg	0.66	0.67	0.65	30

Confusion Matrix:

```
[[ 5  7]
 [ 3 15]]
```

In [62]: `import matplotlib.pyplot as plt`
`from sklearn.tree import plot_tree`

```
# Plot the decision tree
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=x.columns, class_names=['no', 'yes'],
plt.show())
```

```
In [64]: # Check the depth of the tree
print(f"Tree depth (height): {clf.get_depth()}")
print(f"Number of leaves: {clf.get_n_leaves()}")
print(f"Total number of nodes: {clf.tree_.node_count}")
```

Tree depth (height): 8

Number of leaves: 21

Total number of nodes: 41

Step 6: Play with parameters:

max_depth min_samples_split min_samples_leaf

```
In [75]: from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

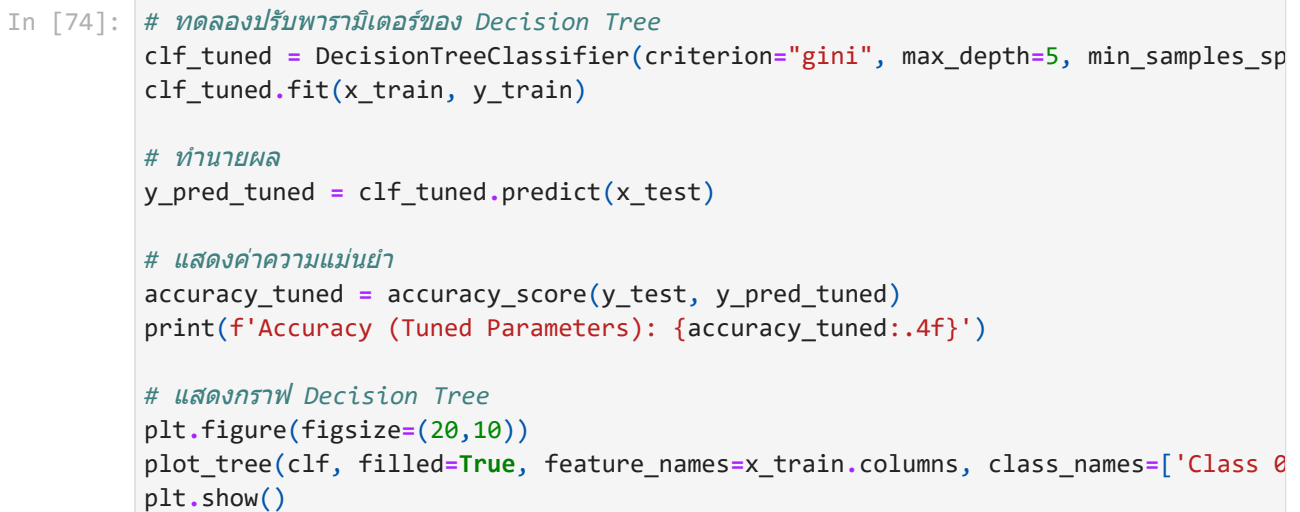
# สร้างโมเดล Decision Tree
clf = DecisionTreeClassifier(criterion="gini", max_depth=2, random_state=42)
clf.fit(x_train, y_train)

# ทำนายผล
y_pred = clf.predict(x_test)

# แสดงค่าความแม่นยำ
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy (Gini Index): {accuracy:.4f}')

# แสดงกราฟ Decision Tree
plt.figure(figsize=(20,10))
plot_tree(clf, filled=True, feature_names=x_train.columns, class_names=['Class 0', 'Class 1'])
plt.show()
```

Accuracy (Gini Index): 0.8000



```

graph TD
    Root["feature_0 <= 0.538  
gini = 0.489  
samples = 120  
value = [51, 69]  
class = Class 1"]
    
    Root -- True --> Node1["feature_4 <= 0.662  
gini = 0.486  
samples = 60  
value = [35, 25]  
class = Class 0"]
    Root -- False --> Node2["feature_1 <= 0.509  
gini = 0.391  
samples = 60  
value = [16, 44]  
class = Class 1"]
    
    Node1 -- True --> Node3["feature_2 <= 0.603  
gini = 0.393  
samples = 41  
value = [30, 11]  
class = Class 0"]
    Node1 -- False --> Node4["feature_3 <= 0.956  
gini = 0.388  
samples = 19  
value = [5, 14]  
class = Class 1"]
    
    Node2 -- True --> Node5["gini = 0.0  
samples = 33  
value = [0, 33]  
class = Class 1"]
    Node2 -- False --> Node6["feature_1 <= 0.854  
gini = 0.483  
samples = 27  
value = [16, 11]  
class = Class 0"]
    
    Node3 -- True --> Node7["feature_4 <= 0.365  
gini = 0.159  
samples = 23  
value = [21, 2]  
class = Class 0"]
    Node3 -- False --> Node8["feature_2 <= 0.661  
gini = 0.5  
samples = 18  
value = [9, 9]  
class = Class 0"]
    
    Node4 -- True --> Node9["feature_4 <= 0.796  
gini = 0.291  
samples = 17  
value = [3, 14]  
class = Class 1"]
    Node4 -- False --> Node10["gini = 0.0  
samples = 2  
value = [2, 0]  
class = Class 0"]
    
    Node6 -- True --> Node11["feature_0 <= 0.769  
gini = 0.291  
samples = 19  
value = [15, 4]  
class = Class 0"]
    Node6 -- False --> Node12["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    
    Node7 -- True --> Node13["gini = 0.0  
samples = 15  
value = [15, 0]  
class = Class 0"]
    Node7 -- False --> Node14["feature_4 <= 0.496  
gini = 0.375  
samples = 8  
value = [6, 2]  
class = Class 0"]
    
    Node8 -- True --> Node15["gini = 0.0  
samples = 14  
value = [0, 4]  
class = Class 1"]
    Node8 -- False --> Node16["feature_2 <= 0.817  
gini = 0.459  
samples = 9  
value = [9, 5]  
class = Class 0"]
    
    Node9 -- True --> Node17["feature_4 <= 0.726  
gini = 0.49  
samples = 7  
value = [3, 4]  
class = Class 1"]
    Node9 -- False --> Node18["gini = 0.0  
samples = 10  
value = [0, 10]  
class = Class 1"]
    
    Node10 -- True --> Node19["feature_1 <= 0.584  
gini = 0.48  
samples = 10  
value = [6, 4]  
class = Class 0"]
    Node10 -- False --> Node20["gini = 0.0  
samples = 9  
value = [9, 0]  
class = Class 0"]
    
    Node11 -- True --> Node21["gini = 0.0  
samples = 9  
value = [19, 0]  
class = Class 0"]
    Node11 -- False --> Node22["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    
    Node14 -- True --> Node23["feature_2 <= 0.432  
gini = 0.444  
samples = 3  
value = [1, 2]  
class = Class 1"]
    Node14 -- False --> Node24["gini = 0.0  
samples = 5  
value = [5, 0]  
class = Class 0"]
    
    Node16 -- True --> Node25["feature_4 <= 0.306  
gini = 0.219  
samples = 8  
value = [7, 1]  
class = Class 0"]
    Node16 -- False --> Node26["feature_4 <= 0.425  
gini = 0.444  
samples = 6  
value = [2, 4]  
class = Class 1"]
    
    Node18 -- True --> Node27["gini = 0.0  
samples = 3  
value = [0, 3]  
class = Class 1"]
    Node18 -- False --> Node28["feature_2 <= 0.435  
gini = 0.375  
samples = 4  
value = [0, 3]  
class = Class 0"]
    
    Node19 -- True --> Node29["feature_1 <= 0.584  
gini = 0.48  
samples = 10  
value = [6, 4]  
class = Class 0"]
    Node19 -- False --> Node30["feature_4 <= 0.321  
gini = 0.444  
samples = 6  
value = [4, 0]  
class = Class 1"]
    
    Node20 -- True --> Node31["gini = 0.0  
samples = 9  
value = [9, 0]  
class = Class 0"]
    Node20 -- False --> Node32["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    
    Node24 -- True --> Node33["feature_2 <= 0.432  
gini = 0.444  
samples = 3  
value = [1, 2]  
class = Class 1"]
    Node24 -- False --> Node34["gini = 0.0  
samples = 5  
value = [5, 0]  
class = Class 0"]
    
    Node26 -- True --> Node35["feature_4 <= 0.306  
gini = 0.219  
samples = 8  
value = [7, 1]  
class = Class 0"]
    Node26 -- False --> Node36["feature_4 <= 0.425  
gini = 0.444  
samples = 6  
value = [2, 4]  
class = Class 1"]
    
    Node28 -- True --> Node37["gini = 0.0  
samples = 3  
value = [0, 3]  
class = Class 1"]
    Node28 -- False --> Node38["feature_2 <= 0.435  
gini = 0.375  
samples = 4  
value = [0, 3]  
class = Class 0"]
    
    Node30 -- True --> Node39["feature_1 <= 0.584  
gini = 0.48  
samples = 10  
value = [6, 4]  
class = Class 0"]
    Node30 -- False --> Node40["feature_4 <= 0.321  
gini = 0.444  
samples = 6  
value = [4, 0]  
class = Class 1"]
    
    Node32 -- True --> Node41["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    Node32 -- False --> Node42["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    
    Node34 -- True --> Node43["feature_2 <= 0.432  
gini = 0.444  
samples = 3  
value = [1, 2]  
class = Class 1"]
    Node34 -- False --> Node44["gini = 0.0  
samples = 5  
value = [5, 0]  
class = Class 0"]
    
    Node36 -- True --> Node45["feature_4 <= 0.306  
gini = 0.219  
samples = 8  
value = [7, 1]  
class = Class 0"]
    Node36 -- False --> Node46["feature_4 <= 0.425  
gini = 0.444  
samples = 6  
value = [2, 4]  
class = Class 1"]
    
    Node38 -- True --> Node47["gini = 0.0  
samples = 3  
value = [0, 3]  
class = Class 1"]
    Node38 -- False --> Node48["feature_2 <= 0.435  
gini = 0.375  
samples = 4  
value = [0, 3]  
class = Class 0"]
    
    Node40 -- True --> Node49["feature_1 <= 0.584  
gini = 0.48  
samples = 10  
value = [6, 4]  
class = Class 0"]
    Node40 -- False --> Node50["feature_4 <= 0.321  
gini = 0.444  
samples = 6  
value = [4, 0]  
class = Class 1"]
    
    Node42 -- True --> Node51["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    Node42 -- False --> Node52["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    
    Node44 -- True --> Node53["feature_2 <= 0.432  
gini = 0.444  
samples = 3  
value = [1, 2]  
class = Class 1"]
    Node44 -- False --> Node54["gini = 0.0  
samples = 5  
value = [5, 0]  
class = Class 0"]
    
    Node46 -- True --> Node55["feature_4 <= 0.306  
gini = 0.219  
samples = 8  
value = [7, 1]  
class = Class 0"]
    Node46 -- False --> Node56["feature_4 <= 0.425  
gini = 0.444  
samples = 6  
value = [2, 4]  
class = Class 1"]
    
    Node48 -- True --> Node57["gini = 0.0  
samples = 3  
value = [0, 3]  
class = Class 1"]
    Node48 -- False --> Node58["feature_2 <= 0.435  
gini = 0.375  
samples = 4  
value = [0, 3]  
class = Class 0"]
    
    Node50 -- True --> Node59["feature_1 <= 0.584  
gini = 0.48  
samples = 10  
value = [6, 4]  
class = Class 0"]
    Node50 -- False --> Node60["feature_4 <= 0.321  
gini = 0.444  
samples = 6  
value = [4, 0]  
class = Class 1"]
    
    Node52 -- True --> Node61["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    Node52 -- False --> Node62["feature_2 <= 0.956  
gini = 0.215  
samples = 8  
value = [1, 7]  
class = Class 1"]
    
    Node54 -- True --> Node63["feature_2 <= 0.432  
gini = 0.444  
samples = 3  
value = [1, 2]  
class = Class 1"]
    Node54 -- False --> Node64["gini = 0.0  
samples = 5  
value = [5, 0]  
class = Class 0"]
    
    Node56 -- True --> Node65["feature_4 <= 0.306  
gini = 0.219  
samples = 8  
value = [7, 1]  
class = Class 0"]
    Node56 -- False --> Node66["feature_4 <= 0.425  
gini = 0.444  
samples = 6  
value = [2, 4]  
class = Class 1"]
    
    Node58 -- True --> Node67["g
```

- `max_depth` : ควบคุมความลึกของต้นไม้เพื่อหลีกเลี่ยง overfitting หรือ underfitting
- `min samples split` : ควบคุมจำนวนตัวอย่างขั้นต่ำในโนดเพื่อให้โนดนั้นสามารถแบ่งได้

- `min_samples_leaf` : ควบคุมจำนวนตัวอย่างขั้นต่ำในใบไม้เพื่อป้องกันการสร้างต้นไม้ที่มีความซับซ้อนเกินไป