- 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

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

Another commonly used formula is:

$$Gini\ impurity = 1 - \Sigma\left(p(i) * \left(1 - p(i)\right)\right)$$

`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
```

```python
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]:

```python
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]:

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

In [20]:

```python
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]:

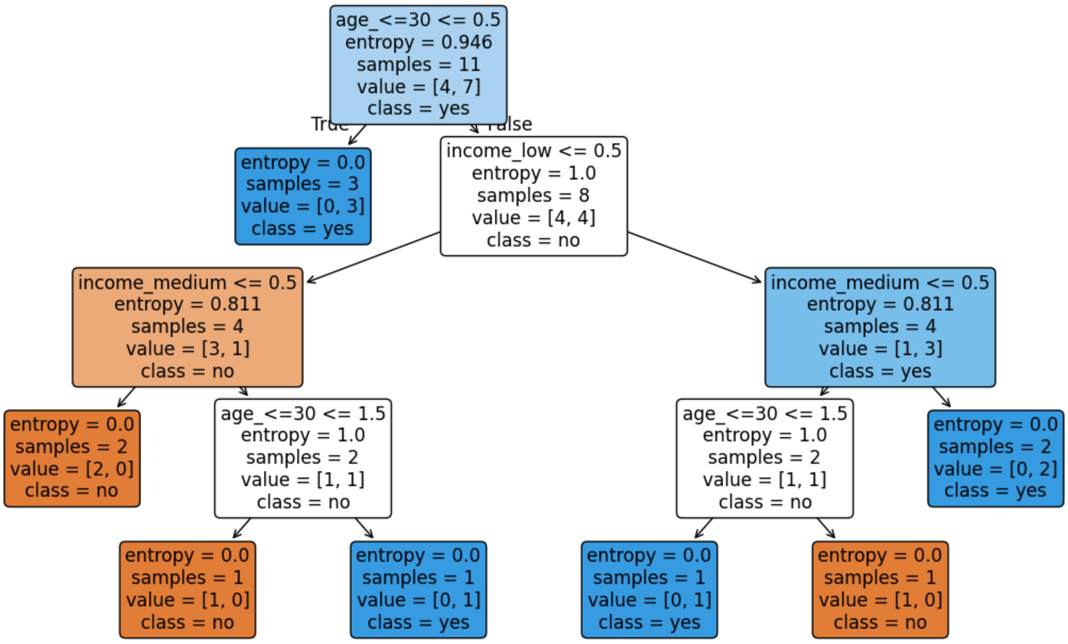
```
from sklearn.metrics import accuracy_score, classification_report, confusion_mat

# 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))
```

```python
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]:
```python
clf_tuned = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=4
clf_tuned.fit(x_train, y_train)
```

Out[45]:

▾ DecisionTreeClassifier

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

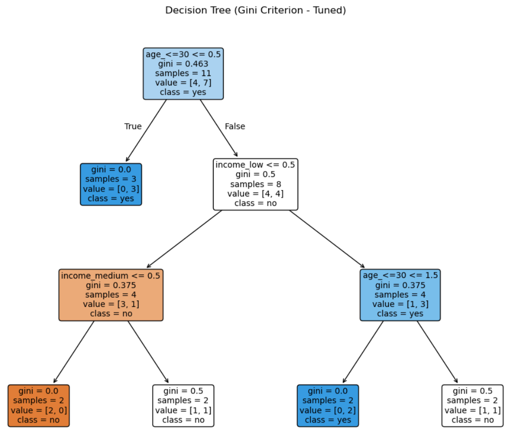
In [46]:
```python
clf_entropy = DecisionTreeClassifier(criterion='entropy', random_state=42)
clf_entropy.fit(x_train, y_train)
```

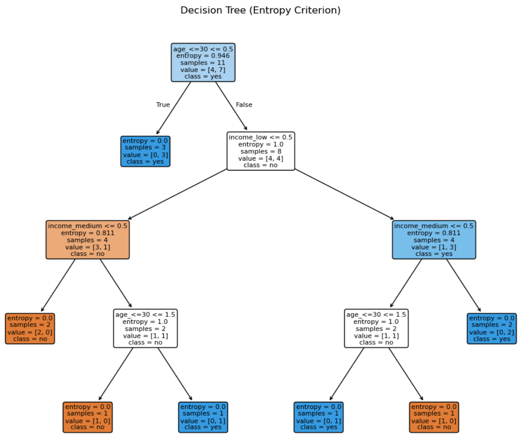
Out[46]:

▾ DecisionTreeClassifier

```
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

In [50]:
```python
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]:

```python
# 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]:

```python
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]:

```python
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=4
2)
```

In [59]:

```
print(x_train.shape)
print(x_test.shape)
```

```
(120, 5)
(30, 5)
```

In [60]:

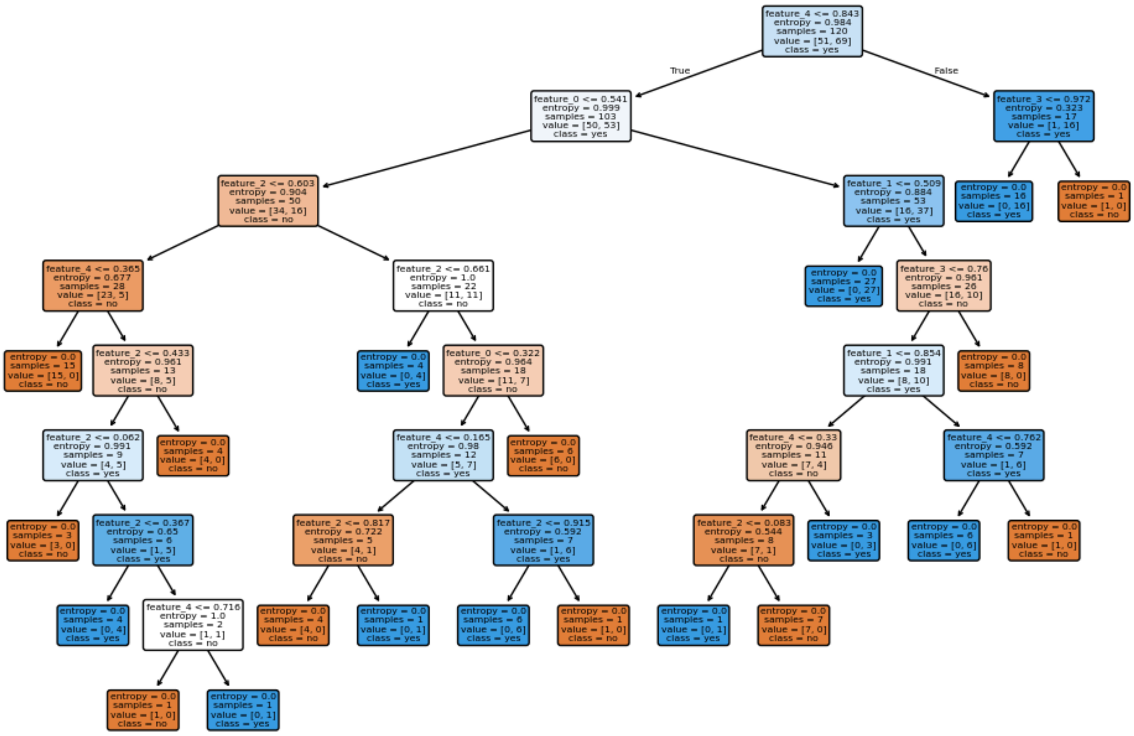
```
from sklearn.tree import DecisionTreeClassifier

# 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}")
```

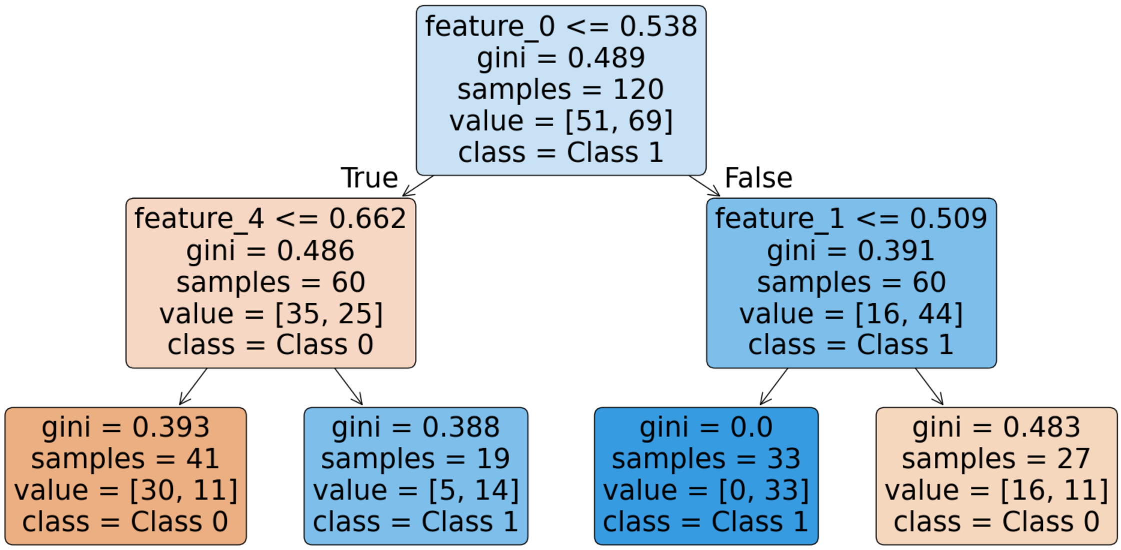
```
Tree depth (height): 8
Number of leaves: 21
Total number of nodes: 41
```

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
plt.show()
```

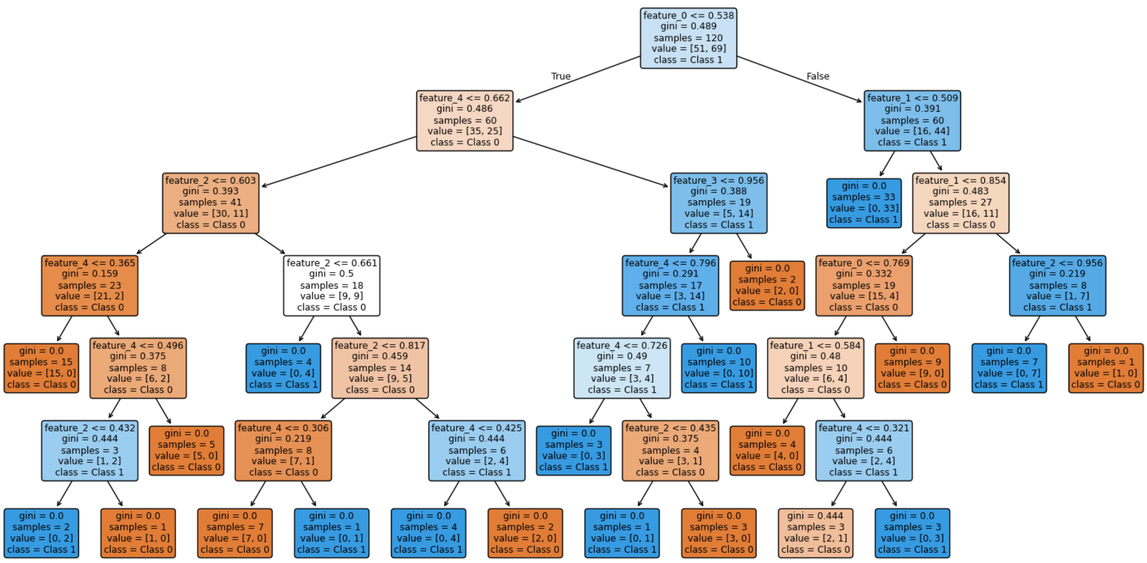
```
Accuracy (Gini Index): 0.8000
```



In [74]:

```python
# ทดลองปรับพารามิเตอร์ของ Decision Tree
clf_tuned = DecisionTreeClassifier(criterion="gini", max_depth=5, min_samples_sp
clf_tuned.fit(x_train, y_train)

# ทำนายผล
y_pred_tuned = clf_tuned.predict(x_test)

# แสดงค่าความแม่นยำ
accuracy_tuned = accuracy_score(y_test, y_pred_tuned)
print(f'Accuracy (Tuned Parameters): {accuracy_tuned:.4f}')

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

```
Accuracy (Tuned Parameters): 0.8333
```



Step 7: อธิบายความเข้าใจหลังจากทดลองใช้พารามิเตอร์ต่างๆ

- `max_depth` : ควบคุมความลึกของต้นไม้เพื่อหลีกเลี่ยง overfitting หรือ underfitting
- `min_samples_split` : ควบคุมจำนวนตัวอย่างขั้นต่ำในโหนดเพื่อให้โหนดนั้นสามารถแบ่งได้

- `min_samples_leaf` : ควบคุมจำนวนตัวอย่างขั้นต่ำในใบไม้เพื่อป้องกันการสร้างต้นไม้ที่มีความซับซ้อนเกินไป