# DECISION TREE

In [2]:
```
import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
```

In [10]:
```
df = pd.read_excel("CourseCompletionPrediction.xlsx")
df.head()
```

Out[10]:

| | Study_Duration | Tests_Attended | Assignment_Submissions | Participation | Course_Completed |
|---|---|---|---|---|---|
| **0** | 12 | 8 | 5 | High | Yes |
| **1** | 8 | 5 | 4 | Medium | Yes |
| **2** | 10 | 7 | 3 | Low | No |
| **3** | 15 | 10 | 6 | High | Yes |
| **4** | 6 | 3 | 2 | Low | No |

In [8]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn import tree
import matplotlib.pyplot as plt
import pandas as pd
```

## Encoding

In [15]:
```
one_hot_encoded = pd.get_dummies(df['Participation'], prefix='Participation')
df = pd.concat([df, one_hot_encoded], axis=1)
df.drop('Participation', axis=1, inplace=True)
```

In [58]:
```
df.head()
```

Out[58]:

| | Study_Duration | Tests_Attended | Assignment_Submissions | Course_Completed | Participation_High | Participation_Low | Participat |
|---|---|---|---|---|---|---|---|
| **0** | 12 | 8 | 5 | 1 | True | False | |
| **1** | 8 | 5 | 4 | 1 | False | False | |
| **2** | 10 | 7 | 3 | 0 | False | True | |
| **3** | 15 | 10 | 6 | 1 | True | False | |
| **4** | 6 | 3 | 2 | 0 | False | True | |

In [19]:
```
label_encoder = LabelEncoder()
df['Course_Completed'] = label_encoder.fit_transform(df['Course_Completed'])
```

In [56]:
```
df.head()
```

Out[56]:

| | Study_Duration | Tests_Attended | Assignment_Submissions | Course_Completed | Participation_High | Participation_Low | Participat |
|---|---|---|---|---|---|---|---|
| **0** | 12 | 8 | 5 | 1 | True | False | |
| **1** | 8 | 5 | 4 | 1 | False | False | |
| **2** | 10 | 7 | 3 | 0 | False | True | |
| **3** | 15 | 10 | 6 | 1 | True | False | |
| **4** | 6 | 3 | 2 | 0 | False | True | |

## Splitting the Data

In [24]:
```
X = df[['Study_Duration', 'Tests_Attended', 'Assignment_Submissions',
        'Participation_High', 'Participation_Low', 'Participation_Medium']]
y = df['Course_Completed']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [36]:
```
model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

Out[36]:
```
DecisionTreeClassifier()
```

In [38]:
```
y_predict = model.predict(X_test)
y_predict
```

Out[38]:
```
array([1, 0, 1, 1, 1, 1, 0, 1, 1, 1])
```

In [40]:
```
accuracy = accuracy_score(y_test, y_predict)
precision = precision_score(y_test, y_predict)
recall = recall_score(y_test, y_predict)
f1 = f1_score(y_test, y_predict)

# Display metrics
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_predict))
```

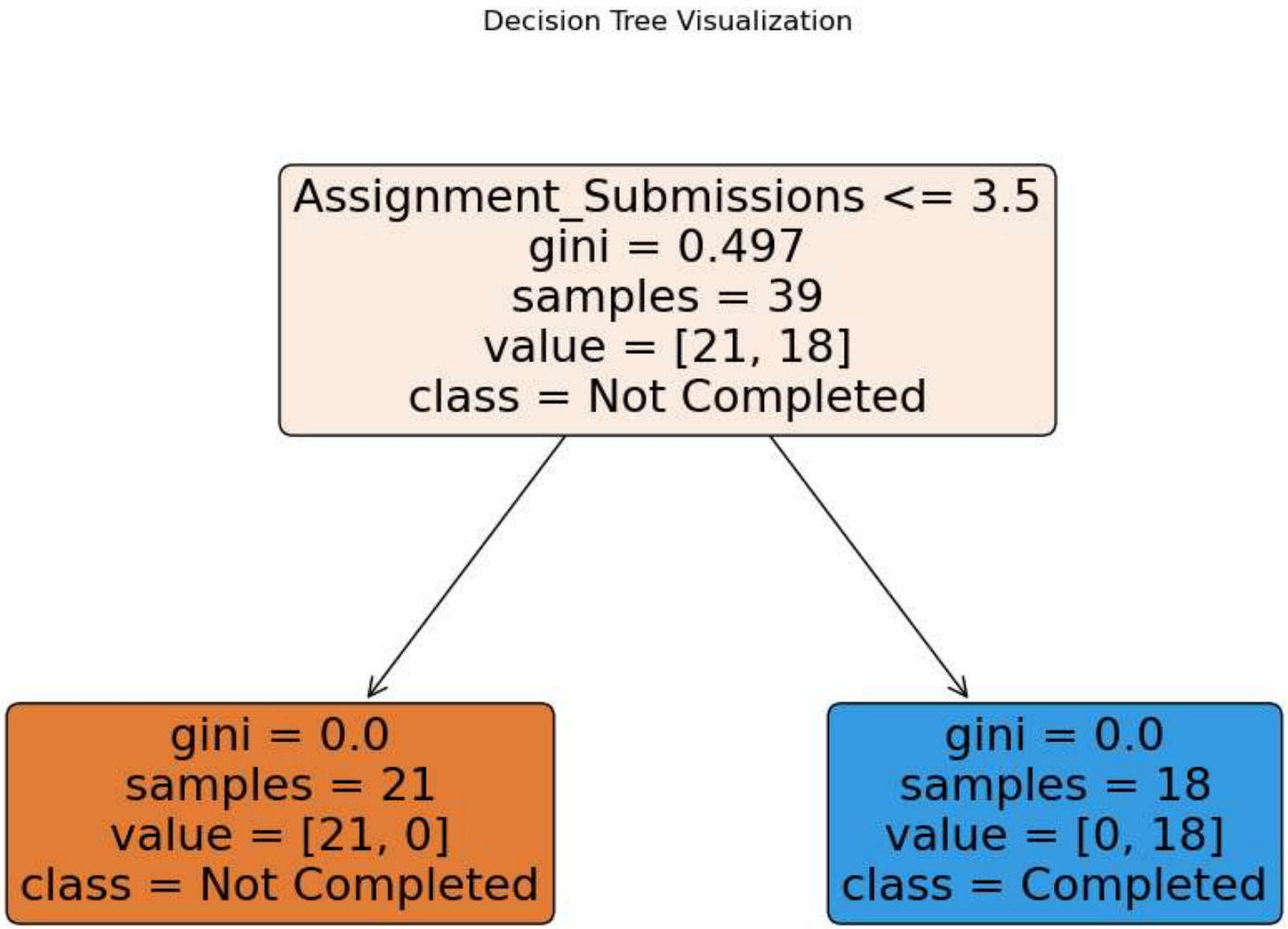
```
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1-Score: 1.00

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         8

    accuracy                           1.00        10
   macro avg       1.00      1.00      1.00        10
weighted avg       1.00      1.00      1.00        10
```

In [42]:
```
plt.figure(figsize=(12, 8))
tree.plot_tree(model, feature_names=X.columns, class_names=['Not Completed', 'Completed'], filled=True, rounded=True)
plt.title("Decision Tree Visualization")
plt.show()
```




In [46]:
```
model2 = DecisionTreeClassifier(
    random_state=42,
    max_depth=4,               # Limit the maximum depth of the tree
    min_samples_split=10,      # Minimum number of samples required to split an internal node
    min_samples_leaf=5         # Minimum number of samples required to be a leaf node
```

```
)
model2.fit(X_train, y_train)
```

Out[46]:

DecisionTreeClassifier

```
DecisionTreeClassifier(max_depth=4, min_samples_leaf=5, min_samples_split=10,
                       random_state=42)
```

In [48]:

```
y_predict2 = model2.predict(X_test)
y_predict2
```

Out[48]:

```
array([1, 0, 1, 1, 1, 1, 0, 1, 1, 1])
```

In [52]:

```
accuracy = accuracy_score(y_test, y_predict2)
precision = precision_score(y_test, y_predict2)
recall = recall_score(y_test, y_predict2)
f1 = f1_score(y_test, y_predict2)

# Display metrics
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_predict2))
```

```
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
F1-Score: 1.00

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         2
           1       1.00      1.00      1.00         8

    accuracy                           1.00        10
   macro avg       1.00      1.00      1.00        10
weighted avg       1.00      1.00      1.00        10
```

In [54]:
```
plt.figure(figsize=(12, 8))
tree.plot_tree(model2, feature_names=X.columns, class_names=['Not Completed', 'Completed'], filled=True, rounded=True
plt.title("Decision Tree Visualization")
plt.show()
```

Decision Tree Visualization