In [1]:
```
import pandas as pd

df = pd.read_csv('data_dt.csv')
df.head()
```

Out[1]:

	ID	Age	Income	Gender	Marital Status	Buys
0	1	0-21	High	Male	Single	No
1	2	0-21	High	Male	Married	No
2	3	21-35	High	Male	Single	Yes
3	4	35-100	Medium	Male	Single	Yes
4	5	35-100	Low	Female	Single	Yes

In [2]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['Age'] = le.fit_transform(df['Age'])
df['Income'] = le.fit_transform(df['Income'])
df['Gender'] = le.fit_transform(df['Gender'])
df['Marital Status'] = le.fit_transform(df['Marital Status'])
df['Buys'] = le.fit_transform(df['Buys'])
df
```

Out[2]:

	ID	Age	Income	Gender	Marital Status	Buys
0	1	0	0	1	1	0
1	2	0	0	1	0	0
2	3	1	0	1	1	1
3	4	2	2	1	1	1
4	5	2	1	0	1	1
5	6	2	1	0	0	0
6	7	1	1	0	0	1
7	8	0	2	1	1	0
8	9	0	1	0	0	1
9	10	2	2	0	1	1
10	11	0	2	0	0	1
11	12	1	2	1	0	1
12	13	1	0	0	1	1
13	14	2	2	1	0	0

In [3]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df.iloc[:,1:-1], df.iloc[:,-1],
print(len(X_train))
print(len(X_test))
print(X_train)
print(X_test)

#iloc: This is an indexer used in Pandas to select data by integer-based location.
```

```
#[:, 1:-1]: This part of the expression specifies the rows and columns to select.
#: before the comma (,) in the first position indicates that you want to select all r
#1:-1 after the comma (,) in the second position indicates that you want to select co
#but not including, the last column.
#So, df.iloc[:, 1:-1] selects all rows of a DataFrame df while excluding the first an

#[:, -1]: This part of the expression specifies the rows and columns to select.
#: before the comma (,) in the first position indicates that you want to select all r
#-1 after the comma (,) in the second position indicates that you want to select only
#So, df.iloc[:, -1] selects all rows of a DataFrame df while only including the last
```

```
11
3
    Age  Income  Gender  Marital Status
2     1       0       1               1
10    0       2       0               0
13    2       2       1               0
11    1       2       1               0
4     2       1       0               1
8     0       1       0               0
9     2       2       0               1
0     0       0       1               1
12    1       0       0               1
6     1       1       0               0
3     2       2       1               1
   Age  Income  Gender  Marital Status
5    2       1       0               0
1    0       0       1               0
7    0       2       1               1
```

In [4]:
```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

Out[4]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [5]:
```
model.score(X_test, y_test)
```

Out[5]:
```
0.6666666666666666
```

In [6]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, model.predict(X_test))

#True positive:predicted positive and it's true.
#False positive: predicted positive and it's false
#False negative: predicted negative and it's false
#True negative: predicted negative and it's true
```

Out[6]:
```
array([[2, 1],
       [0, 0]], dtype=int64)
```

In [ ]: