In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Prithviraj Ghorpade\Downloads\Loans_Dataset.csv",
```

Exploratory Data Analysis

In [3]:
```
print ("Dataset Lenght: ", len(df))
print ("Dataset Shape: ", df.shape)
```

```
Dataset Lenght:  1000
Dataset Shape:  (1000, 6)
```

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   initial_payment  1000 non-null   int64 
 1   last_payment     1000 non-null   int64 
 2   credit_score     1000 non-null   int64 
 3   house_number     1000 non-null   int64 
 4   sum              1000 non-null   int64 
 5   result           1000 non-null   object
dtypes: int64(5), object(1)
memory usage: 47.0+ KB
```

In [5]:
```
df
```

Out[5]:

	initial_payment	last_payment	credit_score	house_number	sum	result
0	201	10018	250	3046	13515	yes
1	205	10016	395	3044	13660	yes
2	257	10129	109	3251	13746	yes
3	246	10064	324	3137	13771	yes
4	117	10115	496	3094	13822	yes
...	...	...	...	...	...	...
995	413	14914	523	4683	20533	No
996	359	14423	927	4838	20547	No
997	316	14872	613	4760	20561	No
998	305	14926	897	4572	20700	No
999	168	14798	834	4937	20737	No

1000 rows × 6 columns

Data Transformation

In [6]:
```
df.drop(columns=['sum'], inplace=True)
```

In [7]:
```
df["result"] = np.where(df["result"] == "yes", 1, 0)
```

In [8]:
```
X = df.drop(columns=['result'])
Y = df['result']
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, rar
```

Decision Tree Classification

In [10]:
```
clf_entropy = DecisionTreeClassifier(criterion = "entropy",
                                     random_state = 100,
                                     max_depth=3,
                                     min_samples_leaf=5)
clf_entropy.fit(X_train, y_train)
```

Out[10]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=
5,
                       random_state=100)
```

In [11]:
```
y_pred = clf_entropy.predict(X_test)
y_pred
```

Out[11]:
```
array([1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
       0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1])
```

In [12]:
```
print('Training set score: {:.2f}'.format(clf_entropy.score(X_train, y_train)))
print('Test set score: {:.2f}'.format(clf_entropy.score(X_test, y_test)))
```

```
Training set score: 0.96
Test set score: 0.94
```

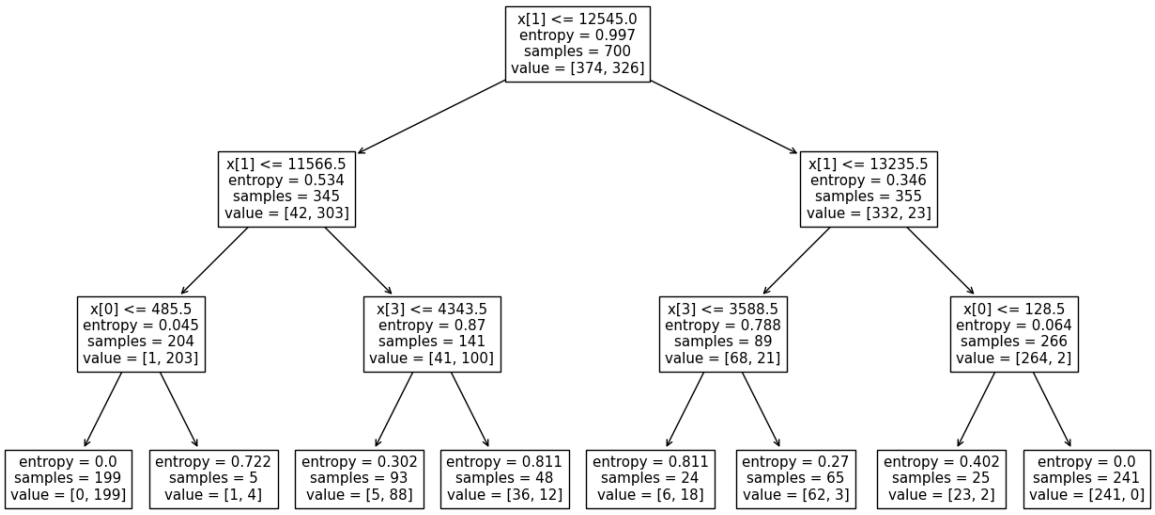
Here, the training-set accuracy score is 0.96 while the test-set accuracy to be 0.94. These two values are quite comparable. So, there is no sign of overfitting.

Decision Tree Visualization

In [13]:
```
plt.figure(figsize=(16,8))

tree.plot_tree(clf_entropy.fit(X_train, y_train))
```

Out[13]:
```
[Text(0.5, 0.875, 'x[1] <= 12545.0\nentropy = 0.997\nsamples = 700\nvalue =
[374, 326]'),
 Text(0.25, 0.625, 'x[1] <= 11566.5\nentropy = 0.534\nsamples = 345\nvalue =
[42, 303]'),
 Text(0.125, 0.375, 'x[0] <= 485.5\nentropy = 0.045\nsamples = 204\nvalue =
[1, 203]'),
 Text(0.0625, 0.125, 'entropy = 0.0\nsamples = 199\nvalue = [0, 199]'),
 Text(0.1875, 0.125, 'entropy = 0.722\nsamples = 5\nvalue = [1, 4]'),
 Text(0.375, 0.375, 'x[3] <= 4343.5\nentropy = 0.87\nsamples = 141\nvalue =
[41, 100]'),
 Text(0.3125, 0.125, 'entropy = 0.302\nsamples = 93\nvalue = [5, 88]'),
 Text(0.4375, 0.125, 'entropy = 0.811\nsamples = 48\nvalue = [36, 12]'),
 Text(0.75, 0.625, 'x[1] <= 13235.5\nentropy = 0.346\nsamples = 355\nvalue =
[332, 23]'),
 Text(0.625, 0.375, 'x[3] <= 3588.5\nentropy = 0.788\nsamples = 89\nvalue =
[68, 21]'),
 Text(0.5625, 0.125, 'entropy = 0.811\nsamples = 24\nvalue = [6, 18]'),
 Text(0.6875, 0.125, 'entropy = 0.27\nsamples = 65\nvalue = [62, 3]'),
 Text(0.875, 0.375, 'x[0] <= 128.5\nentropy = 0.064\nsamples = 266\nvalue =
[264, 2]'),
 Text(0.8125, 0.125, 'entropy = 0.402\nsamples = 25\nvalue = [23, 2]'),
 Text(0.9375, 0.125, 'entropy = 0.0\nsamples = 241\nvalue = [241, 0]')]
```



Make Predictions

In [14]:
```
# Create a DataFrame with the same column names as the training data
client_data = pd.DataFrame([[168, 14798, 834, 4937]],
                           columns=['initial_payment', 'last_payment', 'credit_s

# Make prediction
prediction = clf_entropy.predict(client_data)

# Print the result in a more readable format
print(f"Prediction for client: {'Yes' if prediction[0] == 1 else 'No'} (Raw va
```

```
Prediction for client: No (Raw value: 0)
```