Etape 1:Chargement des données

importation de les bibliotheque

In [98]:
```
import pandas as pd
from sklearn.model_selection import train_test_split  # Pour la division de dataset
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder # pour transformer en valeur numerique
from sklearn.tree import DecisionTreeClassifier, plot_tree
import matplotlib.pyplot as plt
```

charger dataset

In [99]:
```
df=pd.read_csv("Employeetest.csv")
# affichage:
df.head()
```

Out[99]:

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCurrentDomain	LeaveOrNot
0	Bachelors	2017	Bangalore	3	34	Male	No	0	0
1	Bachelors	2013	Pune	1	28	Female	No	3	1
2	Bachelors	2014	New Delhi	3	38	Female	No	2	0
3	Masters	2016	Bangalore	3	27	Male	No	5	1
4	Masters	2017	Pune	3	24	Male	Yes	2	1

les information sur cette dataset:

In [100…
```
df.info()
# Cette dataset contient 8 colonnes et 4653 ligne
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4653 entries, 0 to 4652
Data columns (total 9 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Education                  4653 non-null   object
 1   JoiningYear                4653 non-null   int64
 2   City                       4653 non-null   object
 3   PaymentTier                4653 non-null   int64
 4   Age                        4653 non-null   int64
 5   Gender                     4653 non-null   object
 6   EverBenched                4653 non-null   object
 7   ExperienceInCurrentDomain  4653 non-null   int64
 8   LeaveOrNot                 4653 non-null   int64
dtypes: int64(5), object(4)
memory usage: 327.3+ KB
```

In [101…

```
df.describe()
```

Out[101…

	JoiningYear	PaymentTier	Age	ExperienceInCurrentDomain	LeaveOrNot
count	4653.000000	4653.000000	4653.000000	4653.000000	4653.000000
mean	2015.062970	2.698259	29.393295	2.905652	0.343864
std	1.863377	0.561435	4.826087	1.558240	0.475047
min	2012.000000	1.000000	22.000000	0.000000	0.000000
25%	2013.000000	3.000000	26.000000	2.000000	0.000000
50%	2015.000000	3.000000	28.000000	3.000000	0.000000
75%	2017.000000	3.000000	32.000000	4.000000	1.000000
max	2018.000000	3.000000	41.000000	7.000000	1.000000

Etape 2:Nettoyage

In [102…

```
# la verfification des valeurs manquants:
df.isnull().sum()
# dans cette data n'existe pas
```

Out[102…

```
Education                   0
JoiningYear                 0
City                        0
PaymentTier                 0
Age                         0
Gender                      0
EverBenched                 0
ExperienceInCurrentDomain   0
LeaveOrNot                  0
dtype: int64
```

Etape 3:Transformation

In [103…

```
# colonne 'Education'
print(df['Education'].value_counts())
print("==============================")
# colonne 'City'
print(df['City'].value_counts())
print("==============================")
# colonne 'Gender'
print(df['Gender'].value_counts())
print("==============================")
# colonne 'EverBenched'
print(df['EverBenched'].value_counts())
print("==============================")
```

```
Education
Bachelors    3601
Masters       873
PHD           179
Name: count, dtype: int64
==============================
City
Bangalore    2228
Pune         1268
New Delhi    1157
Name: count, dtype: int64
==============================
Gender
Male      2778
Female    1875
Name: count, dtype: int64
==============================
EverBenched
No     4175
Yes     478
Name: count, dtype: int64
==============================
```

In [104…

```python
# les colonne non numerique:
tab_col=['Education','City','Gender','EverBenched']

# on va transformer chaque colonne
# on faire une boucle pour a chaque colonne faire transformation
le=LabelEncoder()
for col in tab_col:
    df[col]=le.fit_transform(df[col])

df.head()
```

Out[104…

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCurrentDomain	LeaveOrNot
0	0	2017	0	3	34	1	0	0	0
1	0	2013	2	1	28	0	0	3	1
2	0	2014	1	3	38	0	0	2	0
3	1	2016	0	3	27	1	0	5	1
4	1	2017	2	3	24	1	1	2	1

Etape 4:Split train/test

In [105…

```
# Separation de donnee:
X=df.drop(columns='LeaveOrNot')
y=df['LeaveOrNot']

# Division:
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3257, 8)
(1396, 8)
(3257,)
(1396,)
```

Etape 5:Entrainement du modele Decision Tree

In [110…

```
# on creer un model de  arbre de decision avec le constuctor "DecisionTreeClassifier()"
model=DecisionTreeClassifier(
    criterion='entropy',
    max_depth=4,
    random_state=42
```

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

#Prediction:
y_pred=model.predict(X_test)
```

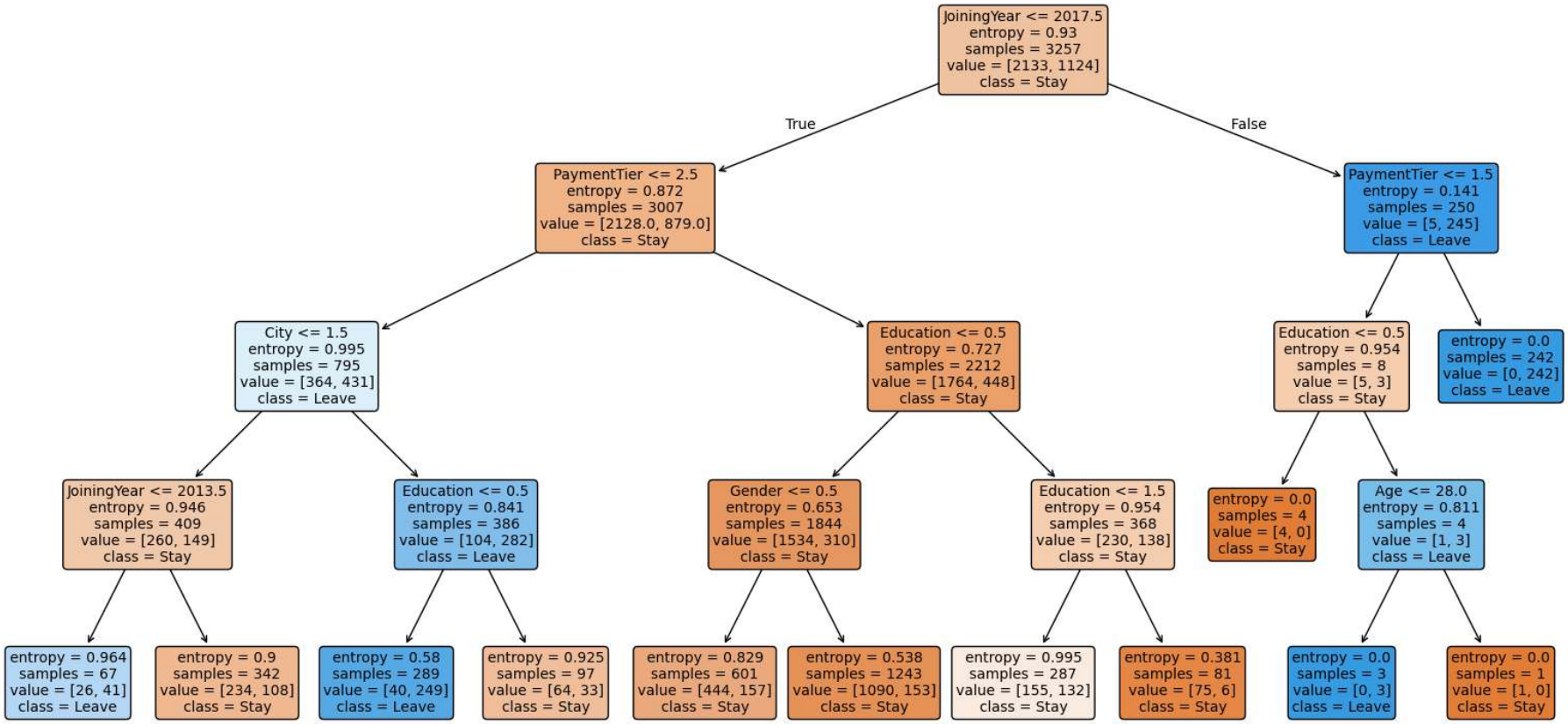
Etape 6: Evaluation du modele

In [111…
```
performance=accuracy_score(y_pred,y_test)
print("Evaluation du modele :")
print(f"Accuracy :{performance*100:.4f}%")
```

```
Evaluation du modele :
Accuracy :81.7335%
```

Etape 7:Visualisation de l'arbre

In [112…
```
plt.figure(figsize=(20, 10))
plot_tree(model, feature_names=X.columns, class_names=['Stay', 'Leave'], filled=True, rounded=True, fontsize=10)
plt.show()
```



In [113…

```python
import joblib
"""
importe la bibliothèque Joblib, spécialisée dans la sérialisation (sauvegarde/chargement)
d'objets Python volumineux, notamment les modèles scikit-learn.

il est Optimisee pour les objets numpy et scikit-learn, plus rapide et compacte que pickle pour ce cas d'usage.
"""
# Sauvegarder le modèle entraîné sous le nom "decision_tree_model.pkl"
#joblib.dump() sérialise l'objet et l'écrit sur disqu,
#il retourne une liste contenant le chemin du fichier écrit
joblib.dump(model, 'decision_tree_model.pkl')
```

```
print("Modele sauvegarde sous le nom 'decision_tree_model.pkl'")
```

```
Modele sauvegarde sous le nom 'decision_tree_model.pkl'
```