### Importation des bibliothèques

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

### Chargement du Dataset

In [2]:
```
# Charger les données
data = pd.read_csv("Employeetest.csv")
data
```

Out[2]:

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4648** | Bachelors | 2013 | Bangalore | 3 | 26 | Female | No | 4 | 0 |
| **4649** | Masters | 2013 | Pune | 2 | 37 | Male | No | 2 | 1 |
| **4650** | Masters | 2018 | New Delhi | 3 | 27 | Male | No | 5 | 1 |
| **4651** | Bachelors | 2012 | Bangalore | 3 | 30 | Male | Yes | 2 | 0 |
| **4652** | Bachelors | 2015 | Bangalore | 3 | 33 | Male | Yes | 4 | 0 |

4653 rows × 9 columns

### Préparation des données

In [3]:
```
#Vérification des valeurs manquantes
data.isnull().sum()
```

Out[3]:
```
Education                   0
JoiningYear                 0
City                        0
PaymentTier                 0
Age                         0
Gender                      0
EverBenched                 0
ExperienceInCurrentDomain   0
LeaveOrNot                  0
dtype: int64
```

In [4]:
```
#Vérification des valeurs aberrantes
col = ['Age', 'ExperienceInCurrentDomain']
df_clean = data.copy()
for col in col:
    Q1 = df_clean[col].quantile(0.25)
    Q3 = df_clean[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    df_clean = df_clean[(df_clean[col] >= lower_bound) & (df_clean[col] <= upper_bound)]
print("Taille avant :", len(data))
print("Taille après :", len(df_clean))
```

```
Taille avant : 4653
Taille après : 4653
```

### Encodage

In [5]:
```
# Encodage des variables catégorielles simples
encoder = LabelEncoder()
data['EverBenched'] = encoder.fit_transform(data['EverBenched'])
data['Gender'] = encoder.fit_transform(data['Gender'])
data['Education'] = encoder.fit_transform(data['Education'])
```

In [6]:
```
# Encodage One-Hot pour la ville
ohe = OneHotEncoder(sparse_output=False)
type_encoded = ohe.fit_transform(data[['City']])
type_encoded_df = pd.DataFrame(type_encoded, columns=ohe.get_feature_names_out(['City']))

data = pd.concat([data.reset_index(drop=True), type_encoded_df.reset_index(drop=True)], axis=1)
data = data.drop(columns="City")
data.head(30)
```

Out[6]:

| | Education | JoiningYear | PaymentTier | Age | Gender | EverBenched | ExperienceInCurrentDomain | LeaveOrNot | City_Bangalore | City_New Delhi |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 2017 | 3 | 34 | 1 | 0 | 0 | 0 | 1.0 | 0.0 |
| **1** | 0 | 2013 | 1 | 28 | 0 | 0 | 3 | 1 | 0.0 | 0.0 |
| **2** | 0 | 2014 | 3 | 38 | 0 | 0 | 2 | 0 | 0.0 | 1.0 |
| **3** | 1 | 2016 | 3 | 27 | 1 | 0 | 5 | 1 | 1.0 | 0.0 |
| **4** | 1 | 2017 | 3 | 24 | 1 | 1 | 2 | 1 | 0.0 | 0.0 |
| **5** | 0 | 2016 | 3 | 22 | 1 | 0 | 0 | 0 | 1.0 | 0.0 |
| **6** | 0 | 2015 | 3 | 38 | 1 | 0 | 0 | 0 | 0.0 | 1.0 |
| **7** | 0 | 2016 | 3 | 34 | 0 | 0 | 2 | 1 | 1.0 | 0.0 |
| **8** | 0 | 2016 | 3 | 23 | 1 | 0 | 1 | 0 | 0.0 | 0.0 |
| **9** | 1 | 2017 | 2 | 37 | 1 | 0 | 2 | 0 | 0.0 | 1.0 |
| **10** | 1 | 2012 | 3 | 27 | 1 | 0 | 5 | 1 | 1.0 | 0.0 |
| **11** | 0 | 2016 | 3 | 34 | 1 | 0 | 3 | 0 | 0.0 | 0.0 |
| **12** | 0 | 2018 | 3 | 32 | 1 | 1 | 5 | 1 | 0.0 | 0.0 |
| **13** | 0 | 2016 | 3 | 39 | 1 | 0 | 2 | 0 | 1.0 | 0.0 |
| **14** | 0 | 2012 | 3 | 37 | 1 | 0 | 4 | 0 | 1.0 | 0.0 |
| **15** | 0 | 2017 | 1 | 29 | 1 | 0 | 3 | 0 | 1.0 | 0.0 |
| **16** | 0 | 2014 | 3 | 34 | 0 | 0 | 2 | 0 | 1.0 | 0.0 |
| **17** | 0 | 2014 | 3 | 34 | 1 | 0 | 4 | 0 | 0.0 | 0.0 |
| **18** | 0 | 2015 | 2 | 30 | 0 | 0 | 0 | 1 | 0.0 | 0.0 |
| **19** | 0 | 2016 | 2 | 22 | 0 | 0 | 0 | 1 | 0.0 | 1.0 |
| **20** | 0 | 2012 | 3 | 37 | 1 | 0 | 0 | 0 | 1.0 | 0.0 |

| | Education | JoiningYear | PaymentTier | Age | Gender | EverBenched | ExperienceInCurrentDomain | LeaveOrNot | City_Bangalore | City_New Delhi |
|---|---|---|---|---|---|---|---|---|---|---|
| **21** | 1 | 2017 | 2 | 28 | 1 | 0 | 4 | 0 | 0.0 | 1.0 |
| **22** | 0 | 2017 | 2 | 36 | 1 | 0 | 3 | 0 | 0.0 | 1.0 |
| **23** | 0 | 2015 | 3 | 27 | 1 | 1 | 5 | 0 | 1.0 | 0.0 |
| **24** | 0 | 2017 | 3 | 29 | 1 | 0 | 4 | 0 | 1.0 | 0.0 |
| **25** | 0 | 2013 | 3 | 22 | 0 | 1 | 0 | 0 | 1.0 | 0.0 |
| **26** | 0 | 2016 | 3 | 37 | 1 | 0 | 2 | 0 | 1.0 | 0.0 |
| **27** | 0 | 2015 | 3 | 23 | 1 | 0 | 1 | 0 | 1.0 | 0.0 |
| **28** | 0 | 2013 | 2 | 31 | 0 | 0 | 2 | 1 | 0.0 | 0.0 |
| **29** | 1 | 2017 | 2 | 30 | 0 | 0 | 2 | 0 | 0.0 | 1.0 |

### Standarisation des données

In [10]:
```
# Standardisation
colonnes_a_standardiser = ['PaymentTier', 'Age', 'JoiningYear', 'ExperienceInCurrentDomain']
preprocessor = ColumnTransformer(
    transformers=[('scaler', StandardScaler(), colonnes_a_standardiser)],
    remainder='passthrough')
data_transformed = preprocessor.fit_transform(data)
colonnes_finales = colonnes_a_standardiser + [col for col in data.columns if col not in colonnes_a_standardiser]
data_scaled = pd.DataFrame(data_transformed, columns=colonnes_finales)
data_scaled.head()
data=data_scaled
```

### Séparation

In [11]:
```
# Séparation des données
X = data.drop(columns="LeaveOrNot")
```

```
y = data["LeaveOrNot"]
print(data['LeaveOrNot'].value_counts())
```

```
LeaveOrNot
0.0    3053
1.0    1600
Name: count, dtype: int64
```

### Equilibrage du Dataset

In [12]:
```
# Equilibrage des données avec la méthode SMOTE
X = data.drop('LeaveOrNot', axis=1)
y = data['LeaveOrNot'].astype(int)
# Print the original class distribution
print("Original class distribution:", dict(zip(*np.unique(y, return_counts=True))))
# Initialize SMOTE object
smote = SMOTE(sampling_strategy='auto',random_state=42)
# Apply SMOTE to generate synthetic samples for the minority class
X, y = smote.fit_resample(X, y)
# Print the new class distribution after SMOTE
print("Resampled class distribution:", dict(zip(*np.unique(y, return_counts=True))))
```

```
Original class distribution: {np.int64(0): np.int64(3053), np.int64(1): np.int64(1600)}
Resampled class distribution: {np.int64(0): np.int64(3053), np.int64(1): np.int64(3053)}
```

In [13]:
```
print(X.head())
print(y.head())
```

```
   PaymentTier       Age  JoiningYear  ExperienceInCurrentDomain  Education  \
0     0.537503  0.954645     1.039638                  -1.864901        0.0
1    -3.025177 -0.288732    -1.107233                   0.060554        0.0
2     0.537503  1.783563    -0.570515                  -0.581264        0.0
3     0.537503 -0.495961     0.502921                   1.344191        1.0
4     0.537503 -1.117650     1.039638                  -0.581264        1.0

   Gender  EverBenched  City_Bangalore  City_New Delhi  City_Pune
0     1.0          0.0             1.0             0.0        0.0
1     0.0          0.0             0.0             0.0        1.0
2     0.0          0.0             0.0             1.0        0.0
3     1.0          0.0             1.0             0.0        0.0
4     1.0          1.0             0.0             0.0        1.0
0    0
1    1
2    0
3    1
4    1
Name: LeaveOrNot, dtype: int64
```

### Division des données

In [16]:
```
# Division des données (données pour l'entraînnemet et données pour le test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

In [18]:
```
X_train.shape
```

Out[18]: (4579, 10)

### Création du model (Support Vector Machine)

#### *Avec kernel linear*

In [65]:
```
#Création du modèle SVM (linear)
svm_model_linear = SVC(kernel='linear', random_state=42)
# Entraînement
svm_model_linear.fit(X_train, y_train)
# Prédiction
y_pred = svm_model_linear.predict(X_test)
```

```
# Evaluation du model
print("Accuracy :", accuracy_score(y_test, y_pred))
print("\nClassification Report :\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy : 0.6548788474132285

Classification Report :
              precision    recall  f1-score   support

           0       0.65      0.71      0.68       775
           1       0.67      0.60      0.63       752

    accuracy                           0.65      1527
   macro avg       0.66      0.65      0.65      1527
weighted avg       0.66      0.65      0.65      1527


Confusion Matrix:
 [[551 224]
 [303 449]]
```

***Kernel linear***

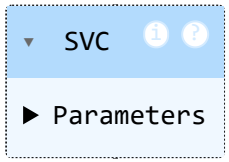
Il trace une ligne droite pour séparer les classes.

Ici les données ne sont pas parfaitement séparables par une ligne droite, le modèle ne peut pas bien capturer toutes les nuances, c'est pour ça certains points sont mal classés il a donné un accuracy un peu faible.

***Avec kernel RBF***

In [19]:
```
# SVM (noyau RBF)
svm_model = SVC(kernel='rbf', C=10, gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)
```

Out[19]:



### Entraînement du model

In [20]:
```
# Prédiction
y_pred = svm_model.predict(X_test)
```

### Evaluation du model

In [21]:
```
# Evaluation du model
print("Accuracy :", accuracy_score(y_test, y_pred))
print("\nClassification Report :\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Accuracy : 0.8369351669941061

Classification Report :
               precision    recall  f1-score   support

           0       0.79      0.93      0.85       775
           1       0.91      0.74      0.82       752

    accuracy                           0.84      1527
   macro avg       0.85      0.84      0.84      1527
weighted avg       0.85      0.84      0.84      1527


Confusion Matrix:
 [[719  56]
 [193 559]]
```

***Kernel rbf***

RBF peut créer des frontières courbes, pour mieux séparer les points.

Les points qui étaient mélangés avec linear peuvent maintenant être correctement classés, car il calcule la distance entre tous les points dans un espace transformé, donc on a moins d'erreurs automatiquement l'accuracy est plus haute.

In [22]:
```
import joblib
"""
```

```
importe la bibliothèque Joblib, spécialisée dans la sérialisation (sauvegarde/chargement)
d'objets Python volumineux, notamment les modeles 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 "support_vector_machine_model.pkl"
#joblib.dump() sérialise L'objet et l'écrit sur disque
#il retourne une liste contenant le chemin du fichier écrit
joblib.dump(svm_model, 'support_vector_machine_model.pkl')

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

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