Aim:

To perform data preprocessing in Python using Pandas and Scikit-learn, including handling missing data, encoding categorical variables, and applying feature scaling techniques.

Procedure:

1. Import the necessary Python libraries — Pandas, NumPy, and preprocessing modules from Scikit-learn.

2. Load the dataset (pre_process_datasample.csv) using pd.read_csv().

3. Handle missing data:

Replace missing numerical values with the mean using SimpleImputer(strategy="mean").

Replace missing categorical values with the mode using fillna().

4. Encode categorical data using OneHotEncoder to convert text values into numerical form.

5. Combine the encoded and numerical data using np.concatenate().

6. Apply feature scaling using:

StandardScaler for standardization.

MinMaxScaler for normalization.

7. Display the final preprocessed dataset.

In [3]:

```
import pandas as pd
import numpy as np
file_path="C:\\Users\\Asus\\Downloads\\pre_process_datasample (1).csv"
df=pd.read_csv(file_path)
print(df)
```

```
   Country   Age   Salary Purchased
0   France  44.0  72000.0        No
1    Spain  27.0  48000.0       Yes
2  Germany  30.0  54000.0        No
3    Spain  38.0  61000.0        No
4  Germany  40.0      NaN       Yes
5   France  35.0  58000.0       Yes
6    Spain   NaN  52000.0        No
7   France  48.0  79000.0       Yes
8  Germany  50.0  83000.0        No
9   France  37.0  67000.0       Yes
```

In [4]:

```
df.head()
```

Out[4]:

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

In [5]:

```
df['Country'].fillna(df['Country'].mode()[0], inplace=True)
features = df.iloc[:, :-1].values
print(features)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 nan]
 ['France' 35.0 58000.0]
 ['Spain' nan 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

```
C:\Users\Asus\AppData\Local\Temp\ipykernel_5992\1814570434.py:1: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained assi
gnment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work becau
se the intermediate object on which we are setting values always behaves as a cop
y.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.meth
od({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to pe
rform the operation inplace on the original object.


  df['Country'].fillna(df['Country'].mode()[0], inplace=True)
```

In [6]:
```
label=df.iloc[:,-1].values
```

In [7]:
```
from sklearn.impute import SimpleImputer
age=SimpleImputer(strategy="mean",missing_values=np.nan)
Salary=SimpleImputer(strategy="mean",missing_values=np.nan)
age.fit(features[:,[1]])
```

Out[7]:
```
▾ SimpleImputer
SimpleImputer()
```

In [8]:
```
Salary.fit(features[:,[2]])
```

Out[8]:
```
▾ SimpleImputer
SimpleImputer()
```

In [9]:
```
SimpleImputer()
```

Out[9]:
```
▾ SimpleImputer
SimpleImputer()
```

In [10]:
```
features[:,[1]]=age.transform(features[:,[1]])
features[:,[2]]=Salary.transform(features[:,[2]])
features
```

Out[10]:
```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, 63777.77777777778],
       ['France', 35.0, 58000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

In [11]:
```
from sklearn.preprocessing import OneHotEncoder
oh=OneHotEncoder(sparse_output=False)
```

```
Country=oh.fit_transform(features[:,[0]])
Country
```

Out[11]:

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

In [12]:

```
final_set=np.concatenate((Country,features[:,[1,2]]),axis=1)
final_set
```

Out[12]:

```
array([[1.0, 0.0, 0.0, 44.0, 72000.0],
       [0.0, 0.0, 1.0, 27.0, 48000.0],
       [0.0, 1.0, 0.0, 30.0, 54000.0],
       [0.0, 0.0, 1.0, 38.0, 61000.0],
       [0.0, 1.0, 0.0, 40.0, 63777.77777777778],
       [1.0, 0.0, 0.0, 35.0, 58000.0],
       [0.0, 0.0, 1.0, 38.77777777777778, 52000.0],
       [1.0, 0.0, 0.0, 48.0, 79000.0],
       [0.0, 1.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```

In [13]:

```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
sc.fit(final_set)
feat_standard_scaler=sc.transform(final_set)
feat_standard_scaler
```

Out[13]:

```
array([[ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         7.58874362e-01,  7.49473254e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.71150388e+00, -1.43817841e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
        -1.27555478e+00, -8.91265492e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
        -1.13023841e-01, -2.53200424e-01],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.77608893e-01,  6.63219199e-16],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -5.48972942e-01, -5.26656882e-01],
       [-8.16496581e-01, -6.54653671e-01,  1.52752523e+00,
         0.00000000e+00, -1.07356980e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
         1.34013983e+00,  1.38753832e+00],
       [-8.16496581e-01,  1.52752523e+00, -6.54653671e-01,
         1.63077256e+00,  1.75214693e+00],
       [ 1.22474487e+00, -6.54653671e-01, -6.54653671e-01,
        -2.58340208e-01,  2.93712492e-01]])
```

In [14]:

```
from sklearn.preprocessing import MinMaxScaler
mms=MinMaxScaler(feature_range=(0,1))
mms.fit(final_set)
feat_minmax_scaler=mms.transform(final_set)
feat_minmax_scaler
```

```
Out[14]: array([[1.        , 0.        , 0.        , 0.73913043, 0.68571429],
                [0.        , 0.        , 1.        , 0.        , 0.        ],
                [0.        , 1.        , 0.        , 0.13043478, 0.17142857],
                [0.        , 0.        , 1.        , 0.47826087, 0.37142857],
                [0.        , 1.        , 0.        , 0.56521739, 0.45079365],
                [1.        , 0.        , 0.        , 0.34782609, 0.28571429],
                [0.        , 0.        , 1.        , 0.51207729, 0.11428571],
                [1.        , 0.        , 0.        , 0.91304348, 0.88571429],
                [0.        , 1.        , 0.        , 1.        , 1.        ],
                [1.        , 0.        , 0.        , 0.43478261, 0.54285714]])
```

Result:

The data preprocessing operations — missing value imputation, encoding, and feature scaling — were completed successfully.