Lab2: Data Preprocessing Tools

Import Libraries

In [5]:

```
pip install scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) ht
tps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pk
g.dev/colab-wheels/public/simple/)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/d
ist-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/
dist-packages (from scikit-learn) (1.22.4)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/d
ist-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/
dist-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/pyth
on3.10/dist-packages (from scikit-learn) (3.1.0)
```

In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Import Dataset

In [7]:

```
dataset = pd.read_csv("Data.csv")
```

Preprocessing steps

Step 1: Divide dataframe into independent variable/ input and dependent / output features

In [8]:

```
X= dataset.iloc[:,:-1]        # remove last column
Y= dataset.iloc[:,-1]       # only include last columns
```

In [9]:

```
print(X)
```

```
   Country   Age   Salary
0   France  44.0  72000.0
1    Spain  27.0  48000.0
2  Germany  30.0  54000.0
3    Spain  38.0  61000.0
4  Germany  40.0      NaN
5   France  35.0  58000.0
6    Spain   NaN  52000.0
7   France  48.0  79000.0
8  Germany  50.0  83000.0
9   France  37.0  67000.0
```

In [10]:

```
print(Y)
```

```
0     No
1    Yes
2     No
3     No
4    Yes
5    Yes
6     No
7    Yes
8     No
9    Yes
Name: Purchased, dtype: object
```

step 2: Handle the missing values in Dataset

In [11]:

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean')        # replace me
imputer.fit(X.iloc[:,1:3])
X.iloc[:,1:3] = imputer.transform(X.iloc[:,1:3])
```

In [12]:

```
print(X)
```

```
   Country        Age        Salary
0   France  44.000000  72000.000000
1    Spain  27.000000  48000.000000
2  Germany  30.000000  54000.000000
3    Spain  38.000000  61000.000000
4  Germany  40.000000  63777.777778
5   France  35.000000  58000.000000
6    Spain  38.777778  52000.000000
7   France  48.000000  79000.000000
8  Germany  50.000000  83000.000000
9   France  37.000000  67000.000000
```

Step 3: Encoding Categorical Data

In [13]:

```
dataset['Country'].value_counts()
```

Out[13]:

```
France     4
Spain      3
Germany    3
Name: Country, dtype: int64
```

In [14]:

```
dataset['Purchased'].value_counts()
```

Out[14]:

```
No     5
Yes    5
Name: Purchased, dtype: int64
```

Two Encoding Technique

1. OneHotEncoder = Use when you have more than 2 categories
2. LabelEncoder = Use when you have exactly 2 categories

In [15]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ct= ColumnTransformer(transformers = [('encoder', OneHotEncoder(),[0])],remainder='passt
X=np.array(ct.fit_transform(X))
```

In [16]:

```
print(X)
```

```
[[1.00000000e+00 0.00000000e+00 0.00000000e+00 4.40000000e+01
  7.20000000e+04]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00 2.70000000e+01
  4.80000000e+04]
 [0.00000000e+00 1.00000000e+00 0.00000000e+00 3.00000000e+01
  5.40000000e+04]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00 3.80000000e+01
  6.10000000e+04]
 [0.00000000e+00 1.00000000e+00 0.00000000e+00 4.00000000e+01
  6.37777778e+04]
 [1.00000000e+00 0.00000000e+00 0.00000000e+00 3.50000000e+01
  5.80000000e+04]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00 3.87777778e+01
  5.20000000e+04]
 [1.00000000e+00 0.00000000e+00 0.00000000e+00 4.80000000e+01
  7.90000000e+04]
 [0.00000000e+00 1.00000000e+00 0.00000000e+00 5.00000000e+01
  8.30000000e+04]
 [1.00000000e+00 0.00000000e+00 0.00000000e+00 3.70000000e+01
  6.70000000e+04]]
```

In [17]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
Y=le.fit_transform(Y)
```

In [18]:

```
print(Y)
```

```
[0 1 0 0 1 1 0 1 0 1]
```

Step 4: Spliting Data into Training and Testing

In [19]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3,random_state=1)
```

In [20]:

```
print(X_train.shape)
print(Y_train.shape)
```

```
(7, 5)
(7,)
```

In [21]:

```
X_test
```

Out[21]:

```
array([[0.00000000e+00, 1.00000000e+00, 0.00000000e+00, 3.00000000e+01,
        5.40000000e+04],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 3.70000000e+01,
        6.70000000e+04],
       [0.00000000e+00, 0.00000000e+00, 1.00000000e+00, 3.87777778e+01,
        5.20000000e+04]])
```

step 5: Feature Scaling

min-max scaler == -1 to 1

standard scaler == -2 to 2

In [22]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
#from sklearn.preprocessing import MinMaxScaler
#mn = MinMaxScaler()
X_train[:,3:5] = sc.fit_transform(X_train[:,3:5])
X_test[:,3:5] = sc.fit_transform(X_test[:,3:5])
```

In [23]:

```
print(X_test)
```

```
[[ 0.          1.          0.         -1.38802721 -0.55138018]
 [ 1.          0.          0.          0.45941746  1.40351318]
 [ 0.          0.          1.          0.92860975 -0.852133  ]]
```