In [8]:
```
### Data Preprocessing using Sk-learn

import numpy as np
import pandas as pd
```

In [9]:
```
df = pd.read_csv("C:/Users/SW20407278/Desktop/Final AI/Hands-On/Data Preprocessing/pre
```

In [10]:
```
df.head()
```

Out[10]:

	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 [11]:
```
df
```

Out[11]:

	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	NaN	50.0	83000.0	No
9	France	37.0	67000.0	Yes

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    9 non-null      object 
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

In [13]:
```
#### Sk-learn doesnot support the imputation for non-numerical columns.
#### Pandas is used for the imputation for non-numerical columns.

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

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Country    10 non-null     object 
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object 
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

In [15]:
```
### Features and Labels
features = df.iloc[:,:-1].values
label = df.iloc[:, -1].values
```

In [16]:
```
from sklearn.impute import SimpleImputer

### Creating and Instatiate the Object

age = SimpleImputer(strategy = "mean", missing_values = np.nan )
salary = SimpleImputer(strategy = "mean", missing_values = np.nan)
```

In [17]:
```
### Fitting of the object with the data

age.fit(features[:,[1]])
salary.fit(features[:,[2]])
```

Out[17]:
```
SimpleImputer()
```

In [18]:
```
### Transforming the data with fitted values

features[:, [1]] = age.transform(features[:,[1]])
features[:, [2]] = salary.transform(features[:,[2]])
```

In [19]:
```
features
```

Out[19]:
```
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],
       ['France', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

In [20]:
```
### One Hot Encoding
```

```
from sklearn.preprocessing import OneHotEncoder
oh = OneHotEncoder(sparse = False)
```

In [21]:
```
country = oh.fit_transform(features[:,[0]])
```

In [22]:
```
country
```

Out[22]:
```
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.],
       [1., 0., 0.],
       [1., 0., 0.]])
```

In [23]:
```
final_set = np.concatenate((country, features[:,[1,2]]), axis = 1)
```

In [24]:
```
final_set
```

Out[24]:
```
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],
       [1.0, 0.0, 0.0, 50.0, 83000.0],
       [1.0, 0.0, 0.0, 37.0, 67000.0]], dtype=object)
```