```
import numpy as np
import pandas as pd
import sklearn
```

```
df = pd.read_csv('/content/drive/My Drive/Data Science/Data.csv')
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

```
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
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Independent and Dependent Variable

```
x = df.iloc[:, :-1].values
x
```

```
array([['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]], dtype=object)
```

```
y = df.iloc[:, 3].values
y
```

```
array(['No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes'],
      dtype=object)
```

Handling missing values - Pandas

```
df['Age'] = df['Age'].fillna(value = df['Age'].mean())
df['Salary'] = df['Salary'].fillna(value = df['Salary'].median())
df
```

	Country	Age	Salary	Purchased
0	France	44.000000	72000.0	No
1	Spain	27.000000	48000.0	Yes
2	Germany	30.000000	54000.0	No
3	Spain	38.000000	61000.0	No
4	Germany	40.000000	61000.0	Yes
5	France	35.000000	58000.0	Yes
6	Spain	38.777778	52000.0	No
7	France	48.000000	79000.0	Yes
8	Germany	50.000000	83000.0	No
9	France	37.000000	67000.0	Yes

Handling Missing Values using sklearn

```
from sklearn.impute import SimpleImputer
im = SimpleImputer(missing_values = np.nan, strategy = 'mean')
im.fit(x[:, 1:3])
x[:, 1:3] = im.transform(x[:,1:3])
x
```

```
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)
```

Encoding Categorical data

Independent variable - Country - 3 Values

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
x = np.array(ct.fit_transform(x))
print(x)
```

```
[[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 48000.0]
 [1.0 0.0 0.0 35.0 58000.0]
 [0.0 0.0 1.0 27.0 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]]
```

Purchased Variable - 2 values - labelencoder

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
print(y)
```

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

Splitting dataset

```
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 = 7)
X_train
```

```
array([['Germany', 30.0, 54000.0],
       ['Spain', 27.0, 48000.0],
       ['France', 37.0, 67000.0],
       ['France', 48.0, 79000.0],
       ['Spain', 38.0, 61000.0],
       ['Spain', 38.77777777777778, 52000.0],
       ['Germany', 40.0, 63777.77777777778]], dtype=object)
```

```
X_test
```

```
array([['Germany', 50.0, 83000.0],
       ['France', 35.0, 58000.0],
       ['France', 44.0, 72000.0]], dtype=object)
```

```
y_train
```

```
array(['No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes'], dtype=object)
```

```
y_test
```

```
array(['No', 'Yes', 'No'], dtype=object)
```

Feature Scaling - Standardization

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:, 3:] = sc.fit_transform(X_train[:, 3:])
X_test[:,3:] = sc.transform(X_test[:,3:])
X_train
```

Feature Scaling - Normalization

```
from sklearn.preprocessing import Normalizer
nm = Normalizer()
X_train[:, 3:] = nm.fit_transform(X_train[:, 3:])
X_test[:, 3:] = nm.transform(X_test[:, 3:])
X_train
```