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

dataset = pd.read_csv("Data.csv")
df = pd.DataFrame(dataset)

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

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

print(X)
print(y)
```

```
[['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]]
['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
```

```
df.isnull().sum()
```

```
Country      0
Age          1
Salary       1
Purchased    0
dtype: int64
```

```
df1 = df.copy()
```

```
# summarize the shape of the raw data
print("Before:",df1.shape)

# drop rows with missing values
df1.dropna(inplace=True)

# summarize the shape of the data with missing rows removed
print("After:",df1.shape)
```

```
Before: (10, 4)
After: (8, 4)
```

```
df2 = df.copy()

import warnings
warnings.filterwarnings('ignore')

# Fill only numeric columns with mean
df2_numeric = df2.select_dtypes(include=[np.number])  # Select numeric
columns
df2[df2_numeric.columns] = df2_numeric.fillna(df2_numeric.mean())  #
Fill only numeric columns

# Count NaN values after filling
print(df2.isnull().sum())

# Display DataFrame
print(df2)
```

```
Country      0
Age          0
Salary       0
Purchased    0
dtype: int64
   Country        Age        Salary Purchased
0   France  44.000000  72000.000000        No
1    Spain  27.000000  48000.000000       Yes
2  Germany  30.000000  54000.000000        No
3    Spain  38.000000  61000.000000        No
4  Germany  40.000000  63777.777778       Yes
5   France  35.000000  58000.000000       Yes
6    Spain  38.777778  52000.000000        No
7   France  48.000000  79000.000000       Yes
8  Germany  50.000000  83000.000000        No
9   France  37.000000  67000.000000       Yes
```

```
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)
```

```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in
/home/vikramaditya/micromamba/lib/python3.9/site-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in
/home/vikramaditya/micromamba/lib/python3.9/site-packages (from
scikit-learn) (2.0.0)
Requirement already satisfied: scipy>=1.6.0 in
/home/vikramaditya/micromamba/lib/python3.9/site-packages (from
scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in
/home/vikramaditya/micromamba/lib/python3.9/site-packages (from
scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/home/vikramaditya/micromamba/lib/python3.9/site-packages (from
scikit-learn) (3.5.0)
Note: you may need to restart the kernel to use updated packages.
```

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

```
print(X)
```

```
[['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]]
```

```
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))
```

```
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
```

```
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 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]]
```

```
pd.get_dummies(df2)
```

```
         Age        Salary  Country_France  Country_Germany
Country_Spain  \
0  44.000000  72000.000000            True            False
False
1  27.000000  48000.000000           False            False
True
2  30.000000  54000.000000           False             True
False
3  38.000000  61000.000000           False            False
True
4  40.000000  63777.777778           False             True
False
5  35.000000  58000.000000            True            False
False
6  38.777778  52000.000000           False            False
True
7  48.000000  79000.000000            True            False
False
8  50.000000  83000.000000           False             True
False
```

```
9  37.000000  67000.000000            True             False
False
```

```
   Purchased_No  Purchased_Yes
0          True          False
1         False           True
2          True          False
3          True          False
4         False           True
5         False           True
6          True          False
7         False           True
8          True          False
9         False           True
```

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

print(y)
```

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

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 1)

print(X_train)
```

```
[[0.0 0.0 1.0 38.77777777777778 52000.0]
 [0.0 1.0 0.0 40.0 63777.77777777778]
 [1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 38.0 61000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [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 35.0 58000.0]]
```

```
print(X_test)
```

```
[[0.0 1.0 0.0 30.0 54000.0]
 [1.0 0.0 0.0 37.0 67000.0]]
```

```
print(y_train)
```

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

```
print(y_test)
```

```
[0 1]
```

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

print(X_train[:, 3:])
```

```
[[0.5120772946859904 0.11428571428571432]
 [0.5652173913043479 0.45079365079365075]
 [0.7391304347826089 0.6857142857142855]
 [0.4782608695652175 0.37142857142857144]
 [0.0 0.0]
 [0.9130434782608696 0.8857142857142857]
 [1.0 1.0]
 [0.34782608695652173 0.2857142857142856]]
```

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

print(X_train[:, 3:])
```

```
[[-0.19159184384578537 -1.0781259408412425]
 [-0.014117293757057581 -0.07013167641635436]
 [0.5667085065333245 0.6335624327104541]
 [-0.3045301939022482 -0.3078661727429788]
 [-1.9018011447007983 -1.4204636155515822]
 [1.1475343068237058 1.2326533634535486]
 [1.4379472069688963 1.5749910381638883]
 [-0.740149544120035 -0.5646194287757338]]
```