```
import pandas as pd

# Creating the dataset
data = {
    "Country": ["India", "Pakistan", "Bhutan", "Bangladesh", "Nepal",
"Srilanka", "Burma", "China", "Afganistan"],
    "Age": [49, 32, 35, None, 45, 40, None, 53, 55],
    "Salary": [62000, 38000, 44000, 51000, None, 48000, 42000, 69000,
73000],
    "Purchased": ["No", "Yes", "No", "No", "Yes", "Yes", "No", "Yes",
"No"]
}

# Creating the DataFrame
df = pd.DataFrame(data)

# Displaying the DataFrame
print(df)
```

```
      Country   Age   Salary Purchased
0       India  49.0  62000.0        No
1    Pakistan  32.0  38000.0       Yes
2      Bhutan  35.0  44000.0        No
3  Bangladesh   NaN  51000.0        No
4       Nepal  45.0      NaN       Yes
5    Srilanka  40.0  48000.0       Yes
6       Burma   NaN  42000.0        No
7       China  53.0  69000.0       Yes
8  Afganistan  55.0  73000.0        No
```

```
from sklearn.impute import SimpleImputer
import numpy as np

X = df.iloc[:,:-1].values #Takes all rows of all columns except the
last column
Y = df.iloc[:,-1].values # Takes all rows of the last column
X
Y
```

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

```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(X[:, 1:3])
X[:, 1:3] = np.round(imputer.transform(X[:, 1:3]),2)
X
# from sklearn.impute import SimpleImputer
# imputer = SimpleImputer(missing_values = 'NaN', strategy = 'median',
axis=0)
# imputer = imputer.fit(X[:, 1:3])
# X[:, 1:3] = imputer.transform(X[:, 1:3])
```

```
# X
# from sklearn.impute import SimpleImputer
# imputer = SimpleImputer(missing_values = 'NaN', strategy =
'most_frequent', axis=0)
# imputer = imputer.fit(X[:, 1:3])
# X[:, 1:3] = imputer.transform(X[:, 1:3])
# X
```

```
array([['India', 49.0, 62000.0],
       ['Pakistan', 32.0, 38000.0],
       ['Bhutan', 35.0, 44000.0],
       ['Bangladesh', 44.14, 51000.0],
       ['Nepal', 45.0, 53375.0],
       ['Srilanka', 40.0, 48000.0],
       ['Burma', 44.14, 42000.0],
       ['China', 53.0, 69000.0],
       ['Afganistan', 55.0, 73000.0]], dtype=object)
```

```
df = pd.DataFrame(X, columns=['Country', 'Age', 'Salary'])
df
```

```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 9,\n  \"fields\": [\n
{\n      \"column\": \"Country\",\n      \"properties\": {\n
\"dtype\": \"string\",\n        \"num_unique_values\": 9,\n
\"samples\": [\n          \"China\",\n          \"Pakistan\",\n
\"Srilanka\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Age\",\n      \"properties\": {\n        \"dtype\": \"date\",\n
\"min\": 32.0,\n        \"max\": 55.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          32.0,\n          40.0,\n
49.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Salary\",\n      \"properties\": {\n        \"dtype\": \"date\",\n
\"min\": 38000.0,\n        \"max\": 73000.0,\n
\"num_unique_values\": 9,\n        \"samples\": [\n          69000.0,\
n          38000.0,\n          48000.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

```
from sklearn.preprocessing import LabelEncoder
```

```
df['Country'] = LabelEncoder().fit_transform(df['Country'])
df['Country']
```

```
0    5
1    7
2    2
3    1
4    6
5    8
6    3
```

```
7    4
8    0
Name: Country, dtype: int64
```

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

```
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=0)
```

```
from sklearn.linear_model import LogisticRegression
```

```
clf = LogisticRegression(random_state=0, solver='lbfgs',
max_iter=1000)
clf.fit(X_train, Y_train)
```

```
LogisticRegression(max_iter=1000, random_state=0)
```

```
# Predicting the test set results
Y_pred = clf.predict(X_test)
```

```
# Display the predicted values
print(Y_pred)
```

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

```
# Import accuracy_score from sklearn
from sklearn.metrics import accuracy_score
```

```
# Calculate and print the accuracy of the model
accuracy = accuracy_score(Y_test, Y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 0.00%
```