```
from google.colab import files
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardSc
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression

# ------------------------------------
# 1) UPLOAD + LOAD FILE
# ------------------------------------
uploaded = files.upload()

for name in uploaded.keys():
    df = pd.read_csv(name)
    print("\n🔥 Loaded File:", name)
    print("\n📄 Data Preview:\n")
    print(df)
    break

print("\n------------------------------------------------------")
print("🔍 Checking Missing Values\n")
print(df.isnull().sum())

# ------------------------------------
# 2) HANDLE MISSING VALUES
# ------------------------------------
df['Age'] = df['Age'].fillna(df['Age'].mean())
df['Salary'] = df['Salary'].fillna(df['Salary'].mean())

print("\n🧽 Missing Values Fixed!\n")
print(df)

# ------------------------------------
# 3) ENCODING (Country → OneHot, Purchased → LabelEncoder)
# ------------------------------------
X = df.iloc[:, :-1]        # features
y = df.iloc[:, -1]         # label

# Label encode Purchased
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# OneHot encode Country
ct = ColumnTransformer(
    transformers=[('encoder', OneHotEncoder(), [0])],
    remainder='passthrough'
)
```

```
X = ct.fit_transform(X)

print("\n🧩 Encoded Features:\n")
print(X)

# ------------------------------------
# 4) TRAIN-TEST SPLIT
# ------------------------------------
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0
)

print("\n📦 Train/Test Split Done")

# ------------------------------------
# 5) SCALING
# ------------------------------------
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

print("\n📊 Scaling Complete")

# ------------------------------------
# 6) TRAIN A MODEL (Logistic Regression)
# ------------------------------------
model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred)

print("\n🤖 Model Trained Successfully!")
print(f"⭐ Accuracy: {acc*100:.2f}%")

# ------------------------------------
# 7) VISUALIZATIONS
# ------------------------------------

plt.figure(figsize=(6,4))
sns.boxplot(x=df["Age"])
plt.title("Age Distribution")
plt.show()

plt.figure(figsize=(6,4))
sns.boxplot(x=df["Salary"])
plt.title("Salary Distribution")
plt.show()

plt.figure(figsize=(6,4))
sns.countplot(x=df["Purchased"])
plt.title("Purchased Count")
plt.show()
```

Choose Files pre_proces…asample.csv

pre_process_datasample.csv(text/csv) - 226 bytes, last modified: 11/19/2025 - 100% done

```
Saving pre_process_datasample.csv to pre_process_datasample (3).csv

🔥 Loaded File: pre_process_datasample (3).csv

📄 Data Preview:

   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

--------------------------------------------------
🔍 Checking Missing Values

Country      0
Age          1
Salary       1
Purchased    0
dtype: int64

🧽 Missing Values Fixed!

   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

🧩 Encoded Features:

[[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
```