```
import zipfile
import os

zip_file_path = '/content/archive6.zip'
output_dir = './'

# Create the output directory if it doesn't exist
os.makedirs(output_dir, exist_ok=True)

# Unzip the file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(output_dir)

print(f"Successfully unzipped '{zip_file_path}' to '{output_dir}'")
```

```
Successfully unzipped '/content/archive6.zip' to './'
```

```
import zipfile

with zipfile.ZipFile('archive6.zip', 'r') as zip_ref:
    zip_ref.extractall()
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder
```

```
df = pd.read_csv('World Economic Classifications v2.csv')
df.head()
```

	country_name	un_class_2014	imf_class_2023	g7	eu_member	fuel_exp_country	wealth_rank	gdp_ppp_2022	gdp_pc_2022
0	Luxembourg	Developed	Advanced	No	Yes	No	1.0	$142,214.00	$127,046.00
1	Singapore	Developing	Advanced	No	No	No	2.0	$127,565.00	$78,115.00
2	Ireland	Developed	Advanced	No	Yes	No	3.0	$126,905.00	$105,362.00
3	Norway	Developed	Advanced	No	No	No	4.0	$114,899.00	$106,594.00
4	Qatar	Developing	Emerging	No	No	Yes	5.0	$114,648.00	$88,046.00

Next steps: Generate code with df | New interactive sheet

```
df['gdp_ppp_2022'] = df['gdp_ppp_2022'].replace('[\$,]', '', regex=True).astype(float)
df['gdp_pc_2022'] = df['gdp_pc_2022'].replace('[\$,]', '', regex=True).astype(float)
```

```
<>:1: SyntaxWarning: invalid escape sequence '\$'
<>:2: SyntaxWarning: invalid escape sequence '\$'
<>:1: SyntaxWarning: invalid escape sequence '\$'
<>:2: SyntaxWarning: invalid escape sequence '\$'
/tmp/ipython-input-871049917.py:1: SyntaxWarning: invalid escape sequence '\$'
  df['gdp_ppp_2022'] = df['gdp_ppp_2022'].replace('[\$,]', '', regex=True).astype(float)
/tmp/ipython-input-871049917.py:2: SyntaxWarning: invalid escape sequence '\$'
  df['gdp_pc_2022'] = df['gdp_pc_2022'].replace('[\$,]', '', regex=True).astype(float)
```

```
df['g7'] = df['g7'].map({'Yes':1, 'No':0})
df['eu_member'] = df['eu_member'].map({'Yes':1, 'No':0})
df['fuel_exp_country'] = df['fuel_exp_country'].map({'Yes':1, 'No':0})
```

```
X = df[['wealth_rank','gdp_ppp_2022','gdp_pc_2022','g7','eu_member','fuel_exp_country']]
y = df['imf_class_2023']
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Create a temporary DataFrame for cleaning X_train and y_train together
train_data = pd.concat([X_train, y_train], axis=1)

# Drop rows where any of the values are NaN
train_data_cleaned = train_data.dropna()

# Separate X_train_cleaned and y_train_cleaned
X_train_cleaned = train_data_cleaned[X_train.columns]
y_train_cleaned_str = train_data_cleaned[y_train.name]

# Initialize and fit label_encoder on *all unique labels* from both train and test sets
# This ensures it knows all possible categories before transforming splits
label_encoder = LabelEncoder()
all_labels_for_fitting = pd.concat([y_train, y_test]).dropna().unique()
label_encoder.fit(all_labels_for_fitting)

y_train_cleaned_encoded = label_encoder.transform(y_train_cleaned_str)

model = SVC()
model.fit(X_train_cleaned, y_train_cleaned_encoded)
```

▾ SVC

SVC()

```
# Create a temporary DataFrame for cleaning X_test and y_test together
test_data = pd.concat([X_test, y_test], axis=1)

# Drop rows where any of the values are NaN
test_data_cleaned = test_data.dropna()

# Separate X_test_cleaned and y_test_cleaned
X_test_cleaned = test_data_cleaned[X_test.columns]
y_test_cleaned_str = test_data_cleaned[y_test.name]

# Transform y_test_cleaned using the *fitted* label_encoder
y_test_cleaned_encoded = label_encoder.transform(y_test_cleaned_str)

# Predict. Model now expects numerical labels and will output them.
```

```
y_pred_encoded = model.predict(X_test_cleaned)

# Now you can proceed to calculate metrics using y_test_cleaned_encoded and y_pred_encoded
```

```
accuracy = accuracy_score(y_test_cleaned_encoded, y_pred_encoded)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8484848484848485
```

```
accuracy = accuracy_score(y_test_cleaned_encoded, y_pred_encoded)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8484848484848485
```

```
f1 = f1_score(y_test_cleaned_encoded, y_pred_encoded, average='weighted')
print("F1 Score:", f1)
```

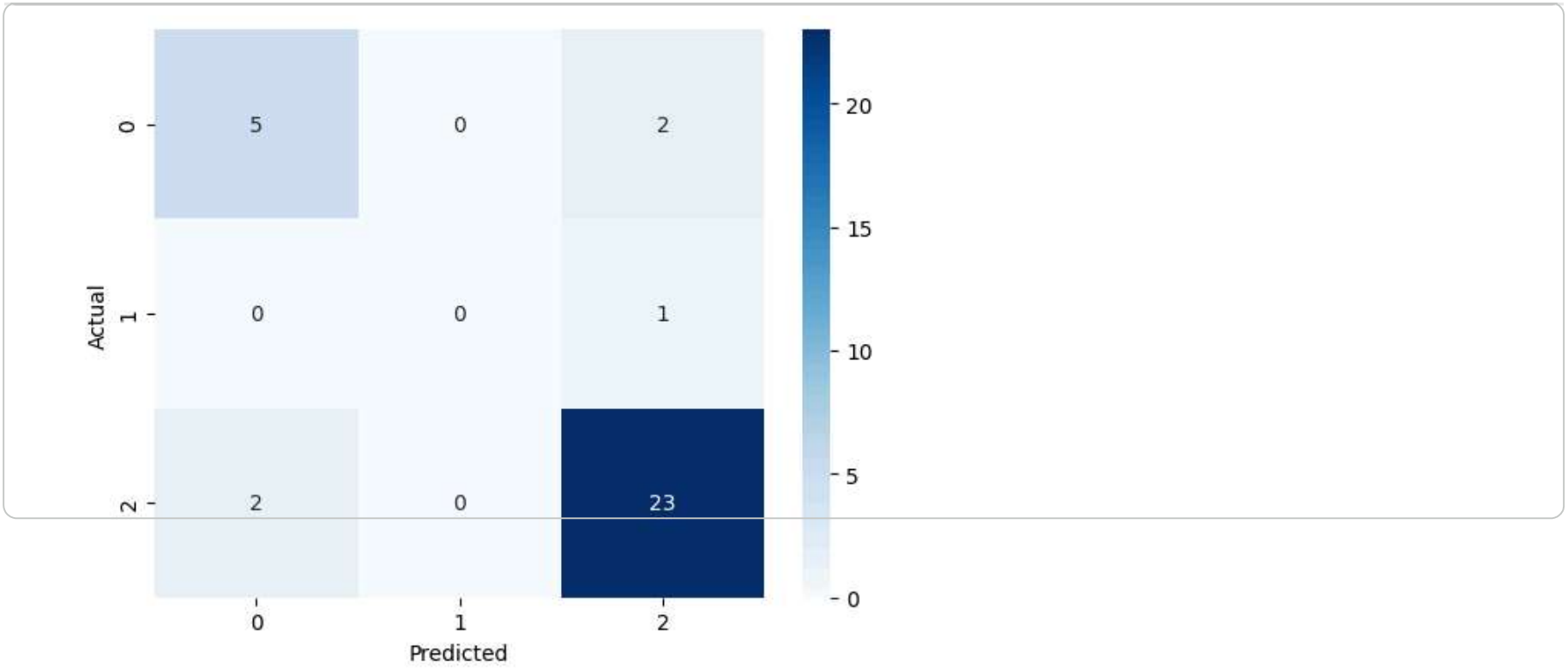
```
F1 Score: 0.8348187759952467
```

```
cm = confusion_matrix(y_test_cleaned_encoded, y_pred_encoded)
print("Confusion Matrix:\n", cm)
```

```
Confusion Matrix:
 [[ 5  0  2]
 [ 0  0  1]
 [ 2  0 23]]
```

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```


5
0
2
0
0
1
2
0
23
Actual
0
1
2
Predicted
0
1
2
20
15
10
5
0