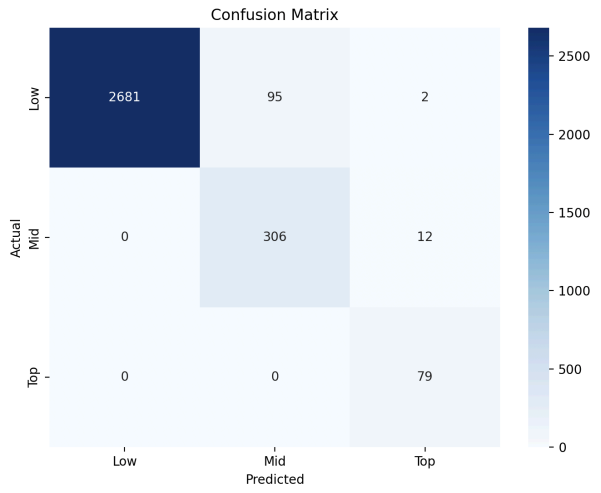
Question:

1. **Dataset Selection:** Download any publicly available dataset suitable for **multiclass classification**. Examples include the Iris dataset, Wine dataset, or any other dataset with **more than two classes**. Make sure the dataset has enough samples and features for training a logistic regression model.
2. **Data Preparation:**
   - Load the dataset and perform any necessary preprocessing
   - Split the dataset into training and testing sets (e.g., 70% training, 30% testing).
3. **Model Building:**
   - Build a **multiclass logistic regression** model using Python (e.g., scikit-learn).
   - Train the model on the training set.
4. **Evaluation:**
   - Evaluate the model on the testing set using appropriate **performance metrics** for multiclass classification, including:
     - Accuracy
     - Precision (per class and macro/micro average)
     - Recall (per class and macro/micro average)
     - F1-score (per class and macro/micro average)
     - Confusion matrix
5. **Analysis:**
   - Display all the above metric values clearly.
   - Provide a brief interpretation of the model's performance. For example, mention which classes are predicted well and which classes are misclassified more often.




```
Confusion Matrix:
[[2681   95    2]
 [   0  306   12]
 [   0    0   79]]

Classification Report:
              precision    recall  f1-score   support

         Low       1.00      0.97      0.98      2778
         Mid       0.76      0.96      0.85       318
         Top       0.85      1.00      0.92        79

    accuracy                           0.97      3175
   macro avg       0.87      0.98      0.92      3175
weighted avg       0.97      0.97      0.97      3175


Example Prediction:
Predicted class = Low
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv('/Users/ishan-college/Desktop/AI (ML Course)/Datasets/animes.csv')

df = df.dropna(subset=["ranked"])  # ranked must not be NaN
df["ranked"] = pd.to_numeric(df["ranked"], errors="coerce")
df = df.dropna(subset=["ranked"])

def convert_rank(r):
    if r <= 200:
        return "Top"
    elif r <= 1000:
        return "Mid"
    else:
        return "Low"

df["rank_category"] = df["ranked"].apply(convert_rank)
target_col = "rank_category"

features = ["members", "popularity", "episodes", "score"]
df = df.dropna(subset=features)

X = df[features]
y = df[target_col]

le = LabelEncoder()
y = le.fit_transform(y)

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

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = LogisticRegression(
    max_iter=5000,
    solver='lbfgs',
    multi_class='auto',
    class_weight='balanced'
)
model.fit(X_train_scaled, y_train)

y_pred = model.predict(X_test_scaled)

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes_, yticklabels=le.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

print("\nConfusion Matrix:")
print(cm)

print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))

example_idx = 0
example_features = X_test_scaled[example_idx].reshape(1, -1)
predicted_class = le.inverse_transform(model.predict(example_features))[0]
print("\nExample Prediction:")
print("Predicted class =", predicted_class)
```