```
# Basic libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# NLP libraries
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Word Cloud (for visuals)
from wordcloud import WordCloud

# Download NLTK stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
# Step 2: Create a richer multilingual dataset
data = {
    'text': [
        "I love this app",                    # English - Positive
        "Worst experience ever",              # English - Negative
        "यह बहुत अच्छा है",                   # Hindi - Positive
        "बहुत बुरा अनुभव",                    # Hindi - Negative
        "ఈ అప్లికేషన్ నన్ను ఆకట్టుకుంది",      # Telugu - Positive
        "నన్ను కోపగొట్టింది",                 # Telugu - Negative
        "இந்த செயலி அருமை",                    # Tamil - Positive (This app is awesome)
        "மிகவும் மோசமான அனுபவம்",              # Tamil - Negative (Very bad experience)
        "ಈ ಅಪ್ಲಿಕೇಶನ್ ಚೆನ್ನಾಗಿದೆ",            # Kannada - Positive (This app is good)
        "ಇದು ಹೇಗೋ ಇಷ್ಟವಿಲ್ಲ",                 # Kannada - Negative (I didn't like it)
        "এই অ্যাপটা দারুন",                    # Bengali - Positive (This app is great)
        "খুবই খারাপ অভিজ্ঞতা",                 # Bengali - Negative (Very bad experience)
    ],
    'label': [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]  # 1 = Positive, 0 = Negative
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Show the updated dataset
df
```

	text	label
0	I love this app	1
1	Worst experience ever	0
2	यह बहुत अच्छा है	1
3	बहुत बुरा अनुभव	0
4	ఈ అప్లికేషన్ నన్ను ఆకట్టుకుంది	1
5	నన్ను కోపగొట్టింది	0
6	இந்த செயலி அருமை	1
7	மிகவும் மோசமான அனுபவம்	0
8	ಈ ಅಪ್ಲಿಕೇಶನ್ ಚೆನ್ನಾಗಿದೆ	1
9	ಇದು ಹೇಗೋ ಇಷ್ಟವಿಲ್ಲ	0
10	এই অ্যাপটা দারুন	1
11	খুবই	

What can I help you build?

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Step 3: Text Preprocessing

import string
from nltk.corpus import stopwords

# Load English stopwords
stop_words = set(stopwords.words('english'))

# Define a basic text cleaning function
def clean_text(text):
    words = text.lower().split()  # lowercase + split into words
    words = [word.strip(string.punctuation) for word in words]  # remove punctuat
    words = [word for word in words if word not in stop_words]  # remove stopworc
    return " ".join(words)

# Apply the cleaning to all rows
df['clean_text'] = df['text'].apply(clean_text)

# Show cleaned text
df
```

	text	label	clean_text
0	I love this app	1	love app
1	Worst experience ever	0	worst experience ever
2	यह बहुत अच्छा है	1	यह बहुत अच्छा है
3	बहुत बुरा अनुभव	0	बहुत बुरा अनुभव
4	ఈ అప్లికేషన్ నన్ను ఆకట్టుకుంది	1	ఈ అప్లికేషన్ నన్ను ఆకట్టుకుంది
5	నన్ను కోపగొట్టింది	0	నన్ను కోపగొట్టింది
6	இந்த செயலி அருமை	1	இந்த செயலி அருமை
7	மிகவும் மோசமான அனுபவம்	0	மிகவும் மோசமான அனுபவம்
8	ಈ ಅಪ್ಲಿಕೇಶನ್ ಚೆನ್ನಾಗಿದೆ	1	ಈ ಅಪ್ಲಿಕೇಶನ್ ಚೆನ್ನಾಗಿದೆ
9	ಇದು ಹೇಗೋ ಇಷ್ಟವಿಲ್ಲ	0	ಇದು ಹೇಗೋ ಇಷ್ಟವಿಲ್ಲ
10	এই অ্যাপটা দারুন	1	এই অ্যাপটা দারুন
11	খুবই খারাপ অভিজ্ঞতা	0	খুবই খারাপ অভিজ্ঞতা

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Step 4: Convert text to numbers using CountVectorizer

from sklearn.feature_extraction.text import CountVectorizer

# Create the vectorizer
vectorizer = CountVectorizer()

# Fit and transform the cleaned text
X = vectorizer.fit_transform(df['clean_text'])

# Labels (0 = negative, 1 = positive)
y = df['label']

# Print shape of result
print("Vector shape (rows, features):", X.shape)

# Optional: See the actual feature names (words)
print("Feature names (words):", vectorizer.get_feature_names_out())
```

```
Vector shape (rows, features): (12, 32)
Feature names (words): ['app' 'ever' 'experience' 'love' 'worst' 'अच' 'अन' 'बह' 'भव' 'यह' 'অভ'
 'এই' 'ఈత' 'পট' 'বই' 'அன' 'அர' 'இந' 'கவ' 'சம' 'பவம' 'ಯಲ' 'অপ' 'ఆకట' 'నన'
 'ಪಗ' 'ಷನ' 'অপ' 'ಇದ' 'ಇಷ' 'টব' 'ಶನ']
```

```
# Step 5: Train the Naive Bayes model

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Split the data (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create the model
model = MultinomialNB()

# Train the model
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Evaluate the model
print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Classification Report:\n", classification_report(y_test, y_pred))
```

```
✅ Accuracy: 0.3333333333333333

📊 Classification Report:
               precision    recall  f1-score   support

           0       0.33      1.00      0.50         1
           1       0.00      0.00      0.00         2

    accuracy                           0.33         3
   macro avg       0.17      0.50      0.25         3
weighted avg       0.11      0.33      0.17         3

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
# Improved clean_text function that handles multilingual text
def clean_text(text):
    if not isinstance(text, str):
        return ""
    try:
        words = text.lower().split()
        words = [word.strip(string.punctuation) for word in words]
        words = [word for word in words if word not in stop_words]
        return " ".join(words)
    except:
        return text
```

```
# Re-clean the updated full dataset
df['clean_text'] = df['text'].apply(clean_text)

# Check the last few rows
df.tail()
```

	text	label	clean_text
25	Wonderful design and UI	1	wonderful design ui
26	बहुत शानदार इंटरफेस	1	बहुत शानदार इंटरफेस
27	பயன்பாட்டு அனுபவம் அருமை	1	பயன்பாட்டு அனுபவம் அருமை
28	ಬಳಕೆದಾರ ಸ್ನೇಹಿ ಅಪ್ಲಿಕೇಶನ್	1	ಬಳಕೆದಾರ ಸ್ನೇಹಿ ಅಪ್ಲಿಕೇಶನ್
29	খুব সুন্দর অ্যাপ্লিকেশন	1	খুব সুন্দর অ্যাপ্লিকেশন

```
# Re-vectorize
X = vectorizer.fit_transform(df['clean_text'])
y = df['label']

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Naive Bayes again
model = MultinomialNB()
model.fit(X_train, y_train)

# Predict & evaluate
y_pred = model.predict(X_test)
print("✅ New Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Updated Classification Report:\n", classification_report(y_test, y_pred))
```

```
✅ New Accuracy: 0.16666666666666666

📊 Updated Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.25      0.33      0.29         3

    accuracy                           0.17         6
   macro avg       0.12      0.17      0.14         6
weighted avg       0.12      0.17      0.14         6
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression

# Vectorize using TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['clean_text'])
y = df['label']
```

```
# Split the data again
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Use Logistic Regression
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predict & Evaluate
y_pred = model.predict(X_test)

# Results
print("✅ Improved Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Updated Classification Report:\n", classification_report(y_test, y_pred))
```

```
✅ Improved Accuracy: 0.5

📊 Updated Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.50      1.00      0.67         3

    accuracy                           0.50         6
   macro avg       0.25      0.50      0.33         6
weighted avg       0.25      0.50      0.33         6

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is il
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
# Combine all clean text into one big string
all_text = " ".join(df['clean_text'])
```

```
# Generate WordCloud
wordcloud = WordCloud(width=800, height=400, background_color='white', font_path=

# Plot the WordCloud
plt.figure(figsize=(12, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Most Frequent Words in Multilingual Dataset", fontsize=16)
plt.show()
```