

```

In [1]: # NLP Text Classification Pipeline

# 1. Import Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.utils import resample
import re
import string

# 2. Load dataset
df = pd.read_csv('unbalanceddataset.csv')
print(df.head())

# 3. Data preprocessing

def clean_text(text):
    text = text.lower()
    text = re.sub(r'http\S+|www\S+|https\S+', '', text) # remove URLs
    text = re.sub(r'@\w+|#', '', text) # remove mentions/hashtags
    text = re.sub(r'^a-z\s', '', text) # remove punctuation/numbers
    return text

# Apply cleaning
df['clean_text'] = df['text'].apply(clean_text)

# Handle imbalance (optional: upsample minority class)
majority = df[df.sentiment == 'positive']
minority = df[df.sentiment == 'negative']

minority_upsampled = resample(minority,
                              replace=True,
                              n_samples=len(majority),
                              random_state=42)

df_balanced = pd.concat([majority, minority_upsampled])

# 4. Split dataset
X = df_balanced['clean_text']
y = df_balanced['sentiment']

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

# 5. Feature Representation

# Option 1: Bag of Words
bow_vectorizer = CountVectorizer(stop_words='english')
X_train_bow = bow_vectorizer.fit_transform(X_train)
X_test_bow = bow_vectorizer.transform(X_test)

```

```
# Option 2: TF-IDF
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# 6. Train models

# Naive Bayes (works well with text)
nb_model = MultinomialNB()
nb_model.fit(X_train_tfidf, y_train)

# Logistic Regression
lr_model = LogisticRegression(max_iter=200)
lr_model.fit(X_train_tfidf, y_train)

# 7. Evaluation

# Naive Bayes predictions
nb_preds = nb_model.predict(X_test_tfidf)
print("Naive Bayes Results:")
print("Accuracy:", accuracy_score(y_test, nb_preds))
print(classification_report(y_test, nb_preds))

# Logistic Regression predictions
lr_preds = lr_model.predict(X_test_tfidf)
print("\nLogistic Regression Results:")
print("Accuracy:", accuracy_score(y_test, lr_preds))
print(classification_report(y_test, lr_preds))

# 8. Save model (optional)
import joblib
joblib.dump(lr_model, 'lr_model.pkl')
joblib.dump(tfidf_vectorizer, 'tfidf_vectorizer.pkl')

print("Pipeline completed successfully!")
```

	text	sentiment
0	Java Concurrency in Practice is probably the b...	positive
1	haha aww hun i bet you are more creative tha...	positive
2	_pickle lol, thank you very much Hope you`re h...	positive
3	Out for an evening on the town with jeremy. Sa...	negative
4	- just took over the #1 Most Endorsed spot on...	positive

Naive Bayes Results:

Accuracy: 0.91875

	precision	recall	f1-score	support
negative	0.88	0.96	0.92	378
positive	0.96	0.89	0.92	422
accuracy			0.92	800
macro avg	0.92	0.92	0.92	800
weighted avg	0.92	0.92	0.92	800

Logistic Regression Results:

Accuracy: 0.905

	precision	recall	f1-score	support
negative	0.87	0.94	0.90	378
positive	0.94	0.88	0.91	422
accuracy			0.91	800
macro avg	0.91	0.91	0.90	800
weighted avg	0.91	0.91	0.91	800

Pipeline completed successfully!