

Practical_Image_Processing_and_Natural_Language_Processing_First

Author: Anwar Aba alwakl

1. Import Libraries

```
import pandas as pd
import re

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC

from sklearn.metrics import accuracy_score, f1_score, classification_report
```

2. Load the Dataset

```
df = pd.read_csv("dataset.csv")

print(df.head())
print(df['sentiment'].value_counts())
```

```

                                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
sentiment
positive      2000
negative       600
Name: count, dtype: int64
```

3. Data Preprocessing

```
def clean_text(text):
    text = text.lower()
    text = re.sub(r'\W', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    return text

df['text'] = df['text'].apply(clean_text)

df = df[['text', 'sentiment']].dropna()
```

4. Split the Dataset

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    df['text'],
    df['sentiment'],
    test_size=0.2,
    stratify=df['sentiment'],
    random_state=42
)
```

5. Feature Representation (TF-IDF)

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(max_features=5000)

X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

6. Train Models

✓ Logistic Regression (imbalance handling)

```
y_pred = lr.predict(X_test_vec)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("F1:", f1_score(y_test, y_pred, average='weighted'))
print(classification_report(y_test, y_pred))
lr = LogisticRegression(class_weight='balanced', max_iter=200)
lr.fit(X_train_vec, y_train)
```

Accuracy: 0.8288461538461539

F1: 0.8341589025703451

	precision	recall	f1-score	support
negative	0.61	0.73	0.66	120
positive	0.91	0.86	0.89	400
accuracy			0.83	520
macro avg	0.76	0.80	0.77	520
weighted avg	0.84	0.83	0.83	520

▼ LogisticRegression ⓘ ?
LogisticRegression(class_weight='balanced', max_iter=200)

✓ Naive Bayes

```

y_pred = nb.predict(X_test_vec)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
print(classification_report(y_test, y_pred))
nb = MultinomialNB()
nb.fit(X_train_vec, y_train)

```

Accuracy: 0.775

F1 Score: 0.682341676197569

	precision	recall	f1-score	support
negative	1.00	0.03	0.05	120
positive	0.77	1.00	0.87	400
accuracy			0.78	520
macro avg	0.89	0.51	0.46	520
weighted avg	0.83	0.78	0.68	520

▼ MultinomialNB ⓘ ?

MultinomialNB()

✓ SVM

```

y_pred = svm.predict(X_test_vec)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
print(classification_report(y_test, y_pred))
svm = LinearSVC(class_weight='balanced')
svm.fit(X_train_vec, y_train)

```

Accuracy: 0.8346153846153846

F1 Score: 0.8346153846153846

	precision	recall	f1-score	support
negative	0.64	0.64	0.64	120
positive	0.89	0.89	0.89	400
accuracy			0.83	520
macro avg	0.77	0.77	0.77	520
weighted avg	0.83	0.83	0.83	520

▼ LinearSVC ⓘ ?

LinearSVC(class_weight='balanced')

7. Evaluation

```

def evaluate(model, X_test, y_test, name):
    y_pred = model.predict(X_test)

```

```

print(f"\n{name}")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))
print(classification_report(y_test, y_pred))

evaluate(lr, X_test_vec, y_test, "Logistic Regression")
evaluate(nb, X_test_vec, y_test, "Naive Bayes")
evaluate(svm, X_test_vec, y_test, "SVM")

```

Logistic Regression

Accuracy: 0.8288461538461539

F1 Score: 0.8341589025703451

	precision	recall	f1-score	support
negative	0.61	0.73	0.66	120
positive	0.91	0.86	0.89	400
accuracy			0.83	520
macro avg	0.76	0.80	0.77	520
weighted avg	0.84	0.83	0.83	520

Naive Bayes

Accuracy: 0.775

F1 Score: 0.682341676197569

	precision	recall	f1-score	support
negative	1.00	0.03	0.05	120
positive	0.77	1.00	0.87	400
accuracy			0.78	520
macro avg	0.89	0.51	0.46	520
weighted avg	0.83	0.78	0.68	520

SVM

Accuracy: 0.8346153846153846

F1 Score: 0.8346153846153846

	precision	recall	f1-score	support
negative	0.64	0.64	0.64	120
positive	0.89	0.89	0.89	400
accuracy			0.83	520
macro avg	0.77	0.77	0.77	520
weighted avg	0.83	0.83	0.83	520

