Start coding or generate with AI.

Mahima gupta

DEVELOPER

📧 guptamahima108@gmail.com

Introduction to Twitter Sentiment Analyzer

From Fundamentals to Real-World AI/NLP — Your LLM Journey Starts Here.

🎯 Objective:

Analyze sentiment in a dataset of tweets to classify their sentiment as positive, neutral, or negative. You'll preprocess the tweets, extract features, and train a machine learning model to predict sentiment. Finally, you'll evaluate the model and test it on new tweets.

📚 Sentiment Analysis — Table of Contents

- 📁 **Load Dataset**
  Load labeled tweets from CSV using pandas.
- ✂️ **Tokenize Text**
  Break tweets into tokens (words) for processing.
- 🧹 **Remove Noise**
  Stopword removal, stemming, and cleaning.
- 🧠 **Feature Extraction**
  TF-IDF, word embeddings (Spacy/GloVe).
- 🤖 **Model Training**
  Train ML models like Logistic Regression.
- 📈 **Evaluation**
  Confusion matrix, accuracy, ROC curve.
- 🧪 **Test on New Tweets**
  Predict sentiment on live or new data.

step 1 import all the necessary libraries and download all the import tools into the workspace

```
import pandas as pd
import nltk
import re
from nltk import PorterStemmer
nltk.download('punkt_tab')

nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

Start coding or generate with AI.

📁 Load Dataset

```
data=pd.read_csv("Tweets.csv")
```

Double-click (or enter) to edit

```
data
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_
	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	
	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	
	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	
	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	
	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	
.	...	...	...	...	...	...	
35	569587686496825344	positive	0.3487	NaN	0.0000	American	
36	569587371693355008	negative	1.0000	Customer Service Issue	1.0000	American	
37	569587242672398336	neutral	1.0000	NaN	NaN	American	
38	569587188687634433	negative	1.0000	Customer Service Issue	0.6659	American	
39	569587140490866689	neutral	0.6771	NaN	0.0000	American	

0 rows × 15 columns

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

Tokenize text

```
from nltk.tokenize import word_tokenize
cols_data = [
    'tweet_id','airline_sentiment','airline_sentiment_confidence','negativereason',
    'negativereason_confidence','airline','airline_sentiment_gold','name',
    'negativereason_gold','retweet_count','text','tweet_coord','tweet_created',
    'tweet_location','user_timezone'
]

for cols in cols_data:
    data[cols + '_tokens'] = data[cols].apply(lambda x: word_tokenize(str(x)))
print(data[['user_timezone', 'user_timezone_tokens']].head())
token_columns = [cols + '_tokens' for cols in cols_data]
```

```
df = data[token_columns]
```

```
                 user_timezone                  user_timezone_tokens
0  Eastern Time (US & Canada)  [Eastern, Time, (, US, &, Canada, )]
1  Pacific Time (US & Canada)  [Pacific, Time, (, US, &, Canada, )]
2  Central Time (US & Canada)  [Central, Time, (, US, &, Canada, )]
3  Pacific Time (US & Canada)  [Pacific, Time, (, US, &, Canada, )]
4  Pacific Time (US & Canada)  [Pacific, Time, (, US, &, Canada, )]
```

Explanation of above code : actually we need to apply tokenization on whole dataset so we will follow these steps that are mentioned below:

1.create a seprate list and store each column in it.

2.run the for loop on the list [cols +_tokens]-> this is called concatenation it will add the tokenize data to their respective columns.

3.data[cols].apply(lambda x: word_tokenize(str(x)))->this lambda function is used in pandas to conver all the numeric or alphabetical data into the string and then word_tokenization is applied over it.

4.token_columns = [cols + '_tokens' for cols in cols_data] df = data[token_columns] --> this step will create a seprate dataset of the tokenized columns.

now the step written below df.head(20) this will show the first 20 rows of dataset named as df.

```
df.head(20)
```

	tweet_id_tokens	airline_sentiment_tokens	airline_sentiment_confidence_tokens	negativereason_tokens	negativereason_confidenc
0	[570306133677760513]	[neutral]	[1.0]	[nan]	
1	[570301130888122368]	[positive]	[0.3486]	[nan]	
2	[570301083672813571]	[neutral]	[0.6837]	[nan]	
3	[570301031407624196]	[negative]	[1.0]	[Bad, Flight]	
4	[570300817074462722]	[negative]	[1.0]	[Ca, n't, Tell]	
5	[570300767074181121]	[negative]	[1.0]	[Ca, n't, Tell]	
6	[570300616901320704]	[positive]	[0.6745]	[nan]	
7	[570300248553349120]	[neutral]	[0.634]	[nan]	
8	[570299953286942721]	[positive]	[0.6559]	[nan]	
9	[570295459631263746]	[positive]	[1.0]	[nan]	
10	[570294189143031808]	[neutral]	[0.6769]	[nan]	
11	[570289724453216256]	[positive]	[1.0]	[nan]	
12	[570289584061480960]	[positive]	[1.0]	[nan]	
13	[570287408438120448]	[positive]	[0.6451]	[nan]	
14	[570285904809598977]	[positive]	[1.0]	[nan]	

15	[570282469121007616]	[negative]	[0.6842]	[Late, Flight]
16	[570277724385734656]	[positive]	[1.0]	[nan]
17	[570276917301137409]	[negative]	[1.0]	[Bad, Flight]
18	[570270684619923457]	[positive]	[1.0]	[nan]
19	[570267956648792064]	[positive]	[1.0]	[nan]

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

Remove Noise

```
token_columns = [col + '_tokens' for col in cols_data]
for col in token_columns:
    clean_col = col.replace('_tokens', '_clean_tokens')
    data[clean_col] = data[col].apply(
        lambda tokens: [re.sub(r'[^a-zA-Z0-9]', '', word) for word in tokens if re.sub(r'[^a-zA-Z0-9]', '', word)]
    )
print(data[[col for col in data.columns if 'clean_tokens' in col]].head())
clean_token_columns = [col + '_clean_tokens' for col in cols_data]
df1 = data[clean_token_columns]
print(df1.head())
```

```
1                          [nan]                         [0]
2                          [nan]                         [0]
3                          [nan]                         [0]
4                          [nan]                         [0]

                                text_clean_tokens tweet_coord_clean_tokens  \
0              [VirginAmerica, What, dhepburn, said]                   [nan]
1 [VirginAmerica, plus, you, ve, added, commerci...                   [nan]
2 [VirginAmerica, I, did, nt, today, Must, mean,...                   [nan]
3 [VirginAmerica, it, s, really, aggressive, to,...                   [nan]
4 [VirginAmerica, and, it, s, a, really, big, ba...                   [nan]

  tweet_created_clean_tokens tweet_location_clean_tokens  \
0   [20150224, 113552, 0800]                       [nan]
1   [20150224, 111559, 0800]                       [nan]
2   [20150224, 111548, 0800]                [Lets, Play]
3   [20150224, 111536, 0800]                       [nan]
4   [20150224, 111445, 0800]                       [nan]

    user_timezone_clean_tokens
0  [Eastern, Time, US, Canada]
1  [Pacific, Time, US, Canada]
2  [Central, Time, US, Canada]
3  [Pacific, Time, US, Canada]
4  [Pacific, Time, US, Canada]
```

```
df1.head()
```

	tweet_id_clean_tokens	airline_sentiment_clean_tokens	airline_sentiment_confidence_clean_tokens	negativereason_clean_tokens	negati
0	[570306133677760513]	[neutral]	[10]	[nan]	
1	[570301130888122368]	[positive]	[03486]	[nan]	
2	[570301083672813571]	[neutral]	[06837]	[nan]	
3	[570301031407624196]	[negative]	[10]	[Bad, Flight]	
4	[570300817074462722]	[negative]	[10]	[Ca, nt, Tell]	

Next steps: Generate code with df1 | View recommended plots | New interactive sheet

now we have our datasset df1 which contains clean and tokenized data

firstly we will convert the data into lower case and then we will remove the stopwords

```
for col in [c + '_clean_tokens' for c in cols_data]:
    data[col] = data[col].apply(lambda tokens: [token.lower() for token in tokens])
lower_case_tokens=[col + '_clean_tokens' for col in cols_data]
df2=data[lower_case_tokens]
print(df2.head())
```

```
  tweet_id_clean_tokens airline_sentiment_clean_tokens  \
0  [570306133677760513]                      [neutral]
1  [570301130888122368]                     [positive]
2  [570301083672813571]                      [neutral]
3  [570301031407624196]                     [negative]
4  [570300817074462722]                     [negative]

  airline_sentiment_confidence_clean_tokens negativereason_clean_tokens  \
0                                      [10]                       [nan]
1                                   [03486]                       [nan]
2                                   [06837]                       [nan]
3                                      [10]               [bad, flight]
4                                      [10]              [ca, nt, tell]
```

```
  negativereason_confidence_clean_tokens airline_clean_tokens  \
0                                  [nan]    [virgin, america]
1                                   [00]    [virgin, america]
2                                  [nan]    [virgin, america]
3                                [07033]    [virgin, america]
4                                   [10]    [virgin, america]

  airline_sentiment_gold_clean_tokens name_clean_tokens  \
0                               [nan]         [cairdin]
1                               [nan]        [jnardino]
2                               [nan]      [yvonnalynn]
3                               [nan]        [jnardino]
4                               [nan]        [jnardino]

  negativereason_gold_clean_tokens retweet_count_clean_tokens  \
0                            [nan]                        [0]
1                            [nan]                        [0]
2                            [nan]                        [0]
3                            [nan]                        [0]
4                            [nan]                        [0]

                                   text_clean_tokens tweet_coord_clean_tokens  \
0             [virginamerica, what, dhepburn, said]                    [nan]
1  [virginamerica, plus, you, ve, added, commerci...                    [nan]
2  [virginamerica, i, did, nt, today, must, mean,...                    [nan]
3  [virginamerica, it, s, really, aggressive, to,...                    [nan]
4  [virginamerica, and, it, s, a, really, big, ba...                    [nan]

  tweet_created_clean_tokens tweet_location_clean_tokens  \
0   [20150224, 113552, 0800]                       [nan]
1   [20150224, 111559, 0800]                       [nan]
2   [20150224, 111548, 0800]                [lets, play]
3   [20150224, 111536, 0800]                       [nan]
4   [20150224, 111445, 0800]                       [nan]

    user_timezone_clean_tokens
0  [eastern, time, us, canada]
1  [pacific, time, us, canada]
2  [central, time, us, canada]
3  [pacific, time, us, canada]
4  [pacific, time, us, canada]
```

stopword removal

```
from nltk.corpus import stopwords
stop_words=set(stopwords.words('english'))
for col in [c + '_clean_tokens' for c in cols_data]:
    data[col] = data[col].apply(lambda tokens: [word for word in tokens if word not in stop_words])
    print(data[[col for col in data.columns if 'clean_tokens' in col]].head())
```

```
4                                [nan]         [jhardino]

   negativereason_gold_clean_tokens retweet_count_clean_tokens  \
0                             [nan]                        [0]
1                             [nan]                        [0]
2                             [nan]                        [0]
3                             [nan]                        [0]
4                             [nan]                        [0]

                                    text_clean_tokens tweet_coord_clean_tokens  \
0                   [virginamerica, dhepburn, said]                    [nan]
1  [virginamerica, plus, added, commercials, expe...                    [nan]
2  [virginamerica, nt, today, must, mean, need, t...                    [nan]
3  [virginamerica, really, aggressive, blast, obn...                    [nan]
4           [virginamerica, really, big, bad, thing]                    [nan]

   tweet_created_clean_tokens tweet_location_clean_tokens  \
0   [20150224, 113552, 0800]                       [nan]
1   [20150224, 111559, 0800]                       [nan]
2   [20150224, 111548, 0800]                [lets, play]
3   [20150224, 111536, 0800]                       [nan]
4   [20150224, 111445, 0800]                       [nan]

    user_timezone_clean_tokens
0  [eastern, time, us, canada]
1  [pacific, time, us, canada]
2  [central, time, us, canada]
3  [pacific, time, us, canada]
4  [pacific, time, us, canada]
```

Feature extraction

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Convert tokens to string (needed by TfidfVectorizer)
data['text_joined'] = data['text_clean_tokens'].apply(lambda tokens: ' '.join(tokens))

# Initialize and apply TF-IDF
tfidf = TfidfVectorizer()
X_tfidf = tfidf.fit_transform(data['text_joined'])

# Convert to DataFrame
tfidf_df = pd.DataFrame(X_tfidf.toarray(), columns=tfidf.get_feature_names_out())
print(tfidf_df.head())
```

```
    00  0011  0016  006  0162389030167  0162424965446  0162431184663  \
0  0.0   0.0   0.0  0.0            0.0            0.0            0.0
1  0.0   0.0   0.0  0.0            0.0            0.0            0.0
2  0.0   0.0   0.0  0.0            0.0            0.0            0.0
3  0.0   0.0   0.0  0.0            0.0            0.0            0.0
4  0.0   0.0   0.0  0.0            0.0            0.0            0.0

   0167560070877  0214  021mbps  ...  zkatcher  zombie  zone  zones  zoom  \
0            0.0   0.0      0.0  ...       0.0     0.0   0.0    0.0   0.0
1            0.0   0.0      0.0  ...       0.0     0.0   0.0    0.0   0.0
2            0.0   0.0      0.0  ...       0.0     0.0   0.0    0.0   0.0
3            0.0   0.0      0.0  ...       0.0     0.0   0.0    0.0   0.0
4            0.0   0.0      0.0  ...       0.0     0.0   0.0    0.0   0.0

   zrh  zrhairport  zukes  zurich  zurichnew
0  0.0         0.0    0.0     0.0        0.0
1  0.0         0.0    0.0     0.0        0.0
2  0.0         0.0    0.0     0.0        0.0
3  0.0         0.0    0.0     0.0        0.0
4  0.0         0.0    0.0     0.0        0.0

[5 rows x 15834 columns]
```

```
!python -m spacy download en_core_web_md
```

```
Collecting en-core-web-md==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_md-3.8.0/en_core_web_md-3.8.0-py3-none-any.whl (3:
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 33.5/33.5 MB 20.0 MB/s eta 0:00:00
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_md')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

```
import spacy

# Load medium or large model (contains word vectors)
nlp = spacy.load("en_core_web_md")  # or "en_core_web_lg"
# If you haven't joined your tokens yet:
data['text_joined'] = data['text_clean_tokens'].apply(lambda tokens: ' '.join(tokens))

# Now convert each tweet into a document vector (spaCy averages token vectors)
data['spacy_vector'] = data['text_joined'].apply(lambda text: nlp(text).vector)

# Print one tweet’s vector
print(data['spacy_vector'].iloc[0])
```

```
[-0.22697    -0.16758333 -0.10122    -0.06573667 -0.05614333  0.11338001
  0.00394633 -0.16217999 -0.07816333  0.7781      0.17492999  0.05065
  0.03858333 -0.07878    -0.14878     0.14313     0.06862666 -0.32492667
 -0.13148    -0.10227666 -0.03285933 -0.04998     0.02426067 -0.11072666
 -0.10690334  0.00726967  0.02936367 -0.07882667 -0.10964    -0.13980334
 -0.16862333  0.01780467  0.00656233  0.14872666 -0.01818233 -0.14604333
 -0.012084    0.11993667 -0.28727666 -0.05272667  0.1095     -0.07440667
  0.01249867  0.010227    0.08958667 -0.06312667 -0.15585001 -0.14135
  0.19861333 -0.08347333 -0.05799333 -0.06317    -0.14030333  0.015592
 -0.00413067  0.00775867 -0.09875    -0.06439     0.04332666  0.006301
  0.11411333  0.01617267 -0.06516334 -0.11399666  0.01698833  0.18946667
 -0.00709733 -0.15994333  0.07218     0.21012335  0.06584667  0.03478667
  0.07935333  0.09755667 -0.16727     0.01664867  0.17651667  0.00873133
 -0.12641333  0.06087    -0.06315333  0.0605     -0.21314667 -0.19624667
  0.07920667 -0.03613667  0.34513333 -0.19285    -0.10499334  0.24010666
 -0.03447667 -0.09092333 -0.01301067  0.13904999  0.09727333 -0.13832334
 -0.08648    -0.07991666  0.02161033 -0.11448666 -0.03741333 -0.04506333
 -0.16206999 -0.06277666 -0.07186667 -0.06054333  0.15708333 -0.28129333
 -0.08632333  0.01785267  0.06191333  0.025157   -0.07402333 -0.10317666
  0.08156667 -0.19108333 -0.01484833 -0.05457667  0.07997667 -0.02644267
  0.08483666 -0.14266667  0.07341    -0.05039667 -0.06004667  0.04425
 -0.09638333 -0.06674    -0.05377333 -0.04417667 -0.11226667 -0.00439633
 -0.15370001 -0.032857   -0.04107    -0.02103934 -0.03520333  0.13373001
 -0.00284913  0.07476667 -0.927       0.07292666 -0.08328333  0.020809
  0.01812567  0.012654   -0.19617666 -0.09988666 -0.22588666 -0.02631567
 -0.14867666  0.09430999 -0.01251767 -0.07229333  0.13295     0.05573
  0.17896332 -0.04505333  0.17913668 -0.13775    -0.06121    -0.15772
  0.04508    -0.07405666  0.008015   -0.06375    -0.09103667 -0.02613767
 -0.25877333 -0.04444667 -0.01626967 -0.06819333 -0.10181334 -0.21732001
 -0.11647666 -0.22354001  0.0010563  -0.06712333 -0.11268333 -0.3514333
 -0.11021667 -0.19195665 -0.14650667  0.17389    -0.06653666 -0.00356
 -0.08586001  0.03279567  0.07727333  0.10881666  0.11964333 -0.00116637
 -0.04244667  0.07719667 -0.10194666 -0.05962     0.20911999 -0.24904001
 -0.01260633  0.0358     -0.05217667  0.06046667 -0.13442     0.005373
 -0.13558333  0.11502     0.07598    -0.06037667  0.05174    -0.02189033
 -0.04645    -0.06154333 -0.05645333  0.06032333  0.18690999 -0.00358967
 -0.01991633 -0.08749667  0.06232667 -0.08005    -0.04125333  0.011909
 -0.021592    0.009687   -0.11513    -0.08738333 -0.02459167  0.08707
 -0.09703001 -0.06457666  0.06506     0.03807667 -0.026583   -0.09266
 -0.15863     0.01478633  0.0022919   0.032714   -0.04469667 -0.006376
 -0.17137001  0.10432333 -0.02355133 -0.15996666 -0.011203   -0.0383
 -0.01906167 -0.04434333 -0.12962334  0.13324    -0.010595   -0.21762334
 -0.06724     0.02452433  0.06763667 -0.11074334  0.11374     0.20774001
 -0.11028334  0.11704666 -0.04857667  0.08384667 -0.15734667 -0.10696667
 -0.09717333  0.12018666 -0.05273    -0.04744    -0.08862334 -0.04085333
 -0.06865334  0.032118   -0.12962334 -0.06393334 -0.13937667  0.02398867
  0.07416    -0.11186666 -0.05047     0.06978667  0.05502667  0.06342667
  0.05410333  0.00605733  0.01561633  0.19137333  0.09449    -0.02775033
 -0.06932333  0.23187     0.08305334 -0.04668666  0.10228333  0.0816
  0.10918333  0.14331     0.08989333  0.18942334  0.11654001  0.14293332]
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics import classification_report, accuracy_score

# Convert cleaned tokens to string (if not already)
data['text_joined'] = data['text_clean_tokens'].apply(lambda x: ' '.join(x))

# TF-IDF Vectorization
```

```
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(data['text_joined'])

# Target
y = data['airline_sentiment']
```

Model training

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7923497267759563
Classification Report:
               precision    recall  f1-score   support

    negative       0.81      0.94      0.87      1889
     neutral       0.67      0.45      0.54       580
    positive       0.82      0.60      0.69       459

    accuracy                           0.79      2928
   macro avg       0.77      0.66      0.70      2928
weighted avg       0.78      0.79      0.78      2928
```

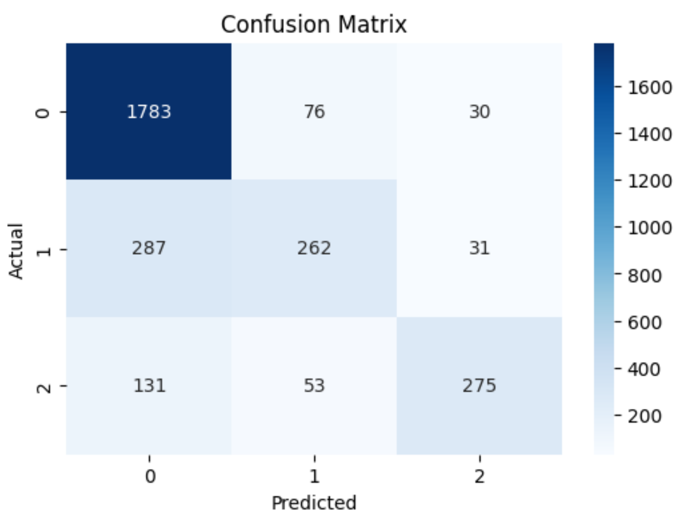
Evaluation

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report

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

# Accuracy
print("Accuracy:", accuracy_score(y_test, y_pred))

# Precision
print("Precision:", precision_score(y_test, y_pred, average='weighted'))

# Recall
print("Recall:", recall_score(y_test, y_pred, average='weighted'))

# F1 Score
print("F1 Score:", f1_score(y_test, y_pred, average='weighted'))

# Confusion Matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Classification Report (summary of all metrics)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.7923497267759563
Precision: 0.7836636649329709
Recall: 0.7923497267759563
F1 Score: 0.7778450779649687
```

```
Confusion Matrix:
 [[1783   76   30]
 [ 287  262   31]
 [ 131   53  275]]
Classification Report:
               precision    recall  f1-score   support

    negative       0.81      0.94      0.87      1889
     neutral       0.67      0.45      0.54       580
    positive       0.82      0.60      0.69       459

    accuracy                           0.79      2928
   macro avg       0.77      0.66      0.70      2928
weighted avg       0.78      0.79      0.78      2928
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot heatmap
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



Visualization

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

# Countplot for sentiment distribution
plt.figure(figsize=(8,5))
sns.countplot(data['airline_sentiment'], palette='pastel')
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()
```

```
/tmp/ipython-input-58-2393977309.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.countplot(data['airline_sentiment'], palette='pastel')
```