

```
!pip install pandas numpy scikit-learn nltk matplotlib seaborn imbalanced-learn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (2.0.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.12/dist-packages (0.14.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from p
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (20
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-lear
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-lea
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from sci
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplot
Requirement already satisfied: cyclor>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplo
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplo
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotl
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplot
Requirement already satisfied: sklearn-compat<0.2,>=0.1.5 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files

unbalanceddataset.csv

unbalanceddataset.csv(text/csv) - 212273 bytes, last modified: 4/15/2026 - 100% done

Saving unbalanceddataset.csv to unbalanceddataset (1).csv

```
import pandas as pd

# Replace 'unbalanceddataset.csv' with your actual file name
df = pd.read_csv('unbalanceddataset.csv')
print("Dataset loaded successfully")
print(f"Total samples: {len(df)}")
print("\nClass distribution:")
print(df['sentiment'].value_counts())
df.head()
```

Dataset loaded successfully
Total samples: 2600

Class distribution:
sentiment
positive 2000
negative 600
Name: count, dtype: int64

	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

```

import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

# Download required NLTK data
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')

stemmer = PorterStemmer()
stop_words = set(stopwords.words('english'))

def clean_text(text):
    text = text.lower()
    text = re.sub(r'http\S+', '', text)          # Remove URLs
    text = re.sub(r'^a-zA-Z\s', '', text)        # Remove punctuation & numbers
    words = word_tokenize(text)
    words = [stemmer.stem(w) for w in words if w not in stop_words]
    return ' '.join(words)

df['clean'] = df['text'].apply(clean_text)
print("Text preprocessing completed")
df[['text', 'clean', 'sentiment']].head()

```

```

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[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!
Text preprocessing completed

```

	text	clean	sentiment
0	Java Concurrency in Practice is probably the b...	java concurr practic probabl best java book iv...	positive
1	haha aww hun i bet you are more creative tha...	haha aww hun bet creativ	positive
2	_pickle lol, thank you very much Hope you're h...	pickl lol thank much hope your great day	positive
3	Out for an evening on the town with jeremy. Sa...	even town jeremi sad carri cant come	negative
4	- just took over the #1 Most Endorsed spot on...	took endors spot twindexxcom thank endors	positive



```

from sklearn.model_selection import train_test_split

X = df['clean']
y = df['sentiment']

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

print(f"Training set size: {len(X_train)}")
print(f"Test set size: {len(X_test)}")
print("\nTraining set distribution:")
print(y_train.value_counts())

```

```

Training set size: 2080
Test set size: 520

```

```
Training set distribution:
sentiment
positive    1600
negative    480
Name: count, dtype: int64
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1,2))
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

print(f"TF-IDF training data shape: {X_train_tfidf.shape}")
```

```
TF-IDF training data shape: (2080, 5000)
```

```
from sklearn.feature_extraction.text import CountVectorizer

count_vec = CountVectorizer(max_features=5000, ngram_range=(1,2))
X_train_count = count_vec.fit_transform(X_train)
X_test_count = count_vec.transform(X_test)

print(f"CountVectorizer training data shape: {X_train_count.shape}")
```

```
CountVectorizer training data shape: (2080, 5000)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

lr = LogisticRegression(max_iter=1000, random_state=42)
lr.fit(X_train_tfidf, y_train)

y_pred_lr = lr.predict(X_test_tfidf)

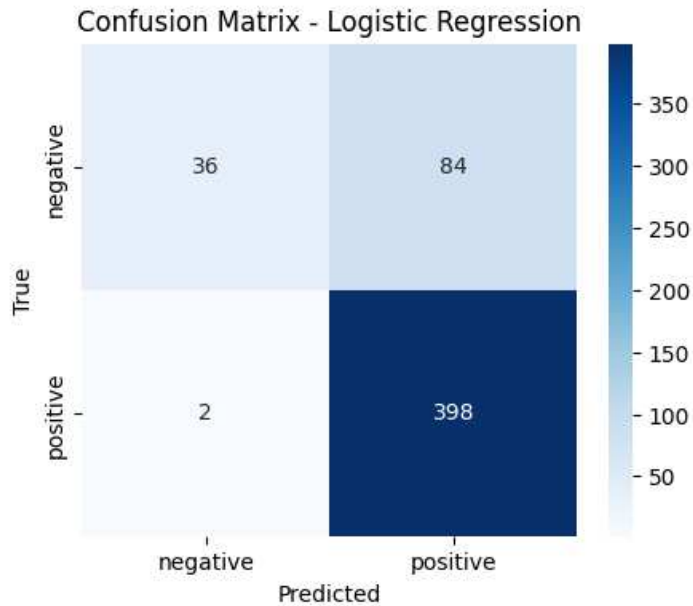
print("✅ Logistic Regression")
print(f"Accuracy: {accuracy_score(y_test, y_pred_lr):.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred_lr))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred_lr)
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=lr.classes_, yticklabels=lr.classes_)
plt.title("Confusion Matrix - Logistic Regression")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

✅ Logistic Regression

Accuracy: 0.8346

	precision	recall	f1-score	support
negative	0.95	0.30	0.46	120
positive	0.83	0.99	0.90	400
accuracy			0.83	520
macro avg	0.89	0.65	0.68	520
weighted avg	0.85	0.83	0.80	520



```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=200, random_state=42)
rf.fit(X_train_tfidf, y_train)

y_pred_rf = rf.predict(X_test_tfidf)

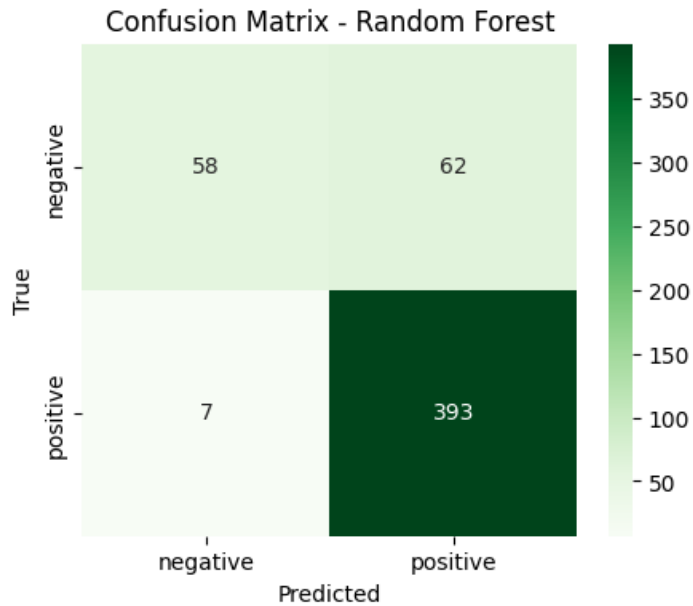
print("✅ Random Forest")
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf))

# Confusion matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(5,4))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Greens', xticklabels=rf.classes_, yticklabels=rf.classes_)
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

✅ Random Forest

Accuracy: 0.8673

Classification Report:		precision	recall	f1-score	support
negative		0.89	0.48	0.63	120
positive		0.86	0.98	0.92	400
accuracy				0.87	520
macro avg		0.88	0.73	0.77	520
weighted avg		0.87	0.87	0.85	520



```
from sklearn.svm import SVC
```

```
svm = SVC(kernel='linear', random_state=42, probability=True)
```

```
svm.fit(X_train_tfidf, y_train)
```

```
y_pred_svm = svm.predict(X_test_tfidf)
```

```
print("✅ Support Vector Machine (SVM)")
```

```
print(f"Accuracy: {accuracy_score(y_test, y_pred_svm):.4f}")
```

```
print("\nClassification Report:\n", classification_report(y_test, y_pred_svm))
```

```
# Confusion matrix
```

```
cm_svm = confusion_matrix(y_test, y_pred_svm)
```

```
plt.figure(figsize=(5,4))
```

```
sns.heatmap(cm_svm, annot=True, fmt='d', cmap='Oranges', xticklabels=svm.classes_, yticklabels=svm.classes_)
```

```
plt.title("Confusion Matrix - SVM")
```

```
plt.xlabel("Predicted")
```

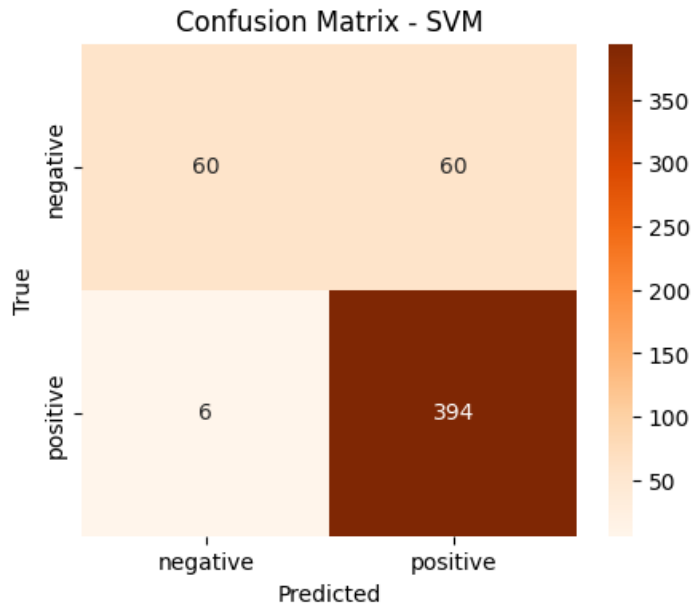
```
plt.ylabel("True")
```

```
plt.show()
```

✔ Support Vector Machine (SVM)

Accuracy: 0.8731

Classification Report:				
	precision	recall	f1-score	support
negative	0.91	0.50	0.65	120
positive	0.87	0.98	0.92	400
accuracy			0.87	520
macro avg	0.89	0.74	0.78	520
weighted avg	0.88	0.87	0.86	520



```
from imblearn.over_sampling import SMOTE
from collections import Counter

smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_tfidf, y_train)

print("Class distribution after SMOTE:", Counter(y_train_smote))

rf_smote = RandomForestClassifier(n_estimators=200, random_state=42)
rf_smote.fit(X_train_smote, y_train_smote)

y_pred_smote = rf_smote.predict(X_test_tfidf)

print("\n✔ Random Forest with SMOTE")
print(f"Accuracy: {accuracy_score(y_test, y_pred_smote):.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred_smote))

# Confusion matrix
cm_smote = confusion_matrix(y_test, y_pred_smote)
plt.figure(figsize=(5,4))
sns.heatmap(cm_smote, annot=True, fmt='d', cmap='Purples', xticklabels=rf_smote.classes_, yticklabels=rf_
plt.title("Confusion Matrix - RF with SMOTE")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

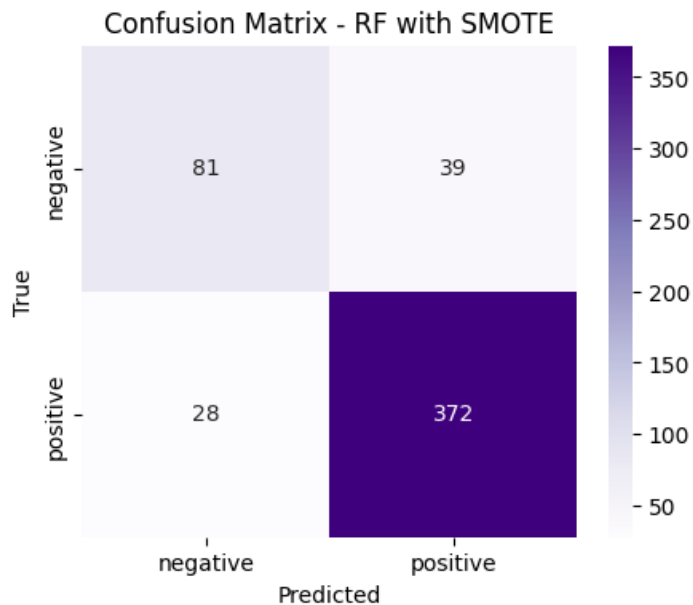
Class distribution after SMOTE: Counter({'positive': 1600, 'negative': 1600})

✅ Random Forest with SMOTE

Accuracy: 0.8712

Classification Report:

	precision	recall	f1-score	support
negative	0.74	0.68	0.71	120
positive	0.91	0.93	0.92	400
accuracy			0.87	520
macro avg	0.82	0.80	0.81	520
weighted avg	0.87	0.87	0.87	520



```
from sklearn.naive_bayes import MultinomialNB
```

```
nb = MultinomialNB()  
nb.fit(X_train_tfidf, y_train)
```

```
y_pred_nb = nb.predict(X_test_tfidf)
```

```
print("✅ Naive Bayes")  
print(f"Accuracy: {accuracy_score(y_test, y_pred_nb):.4f}")  
print("\nClassification Report:\n", classification_report(y_test, y_pred_nb))
```

✅ Naive Bayes

Accuracy: 0.7981

Classification Report:

	precision	recall	f1-score	support
negative	1.00	0.12	0.22	120
positive	0.79	1.00	0.88	400
accuracy			0.80	520
macro avg	0.90	0.56	0.55	520
weighted avg	0.84	0.80	0.73	520

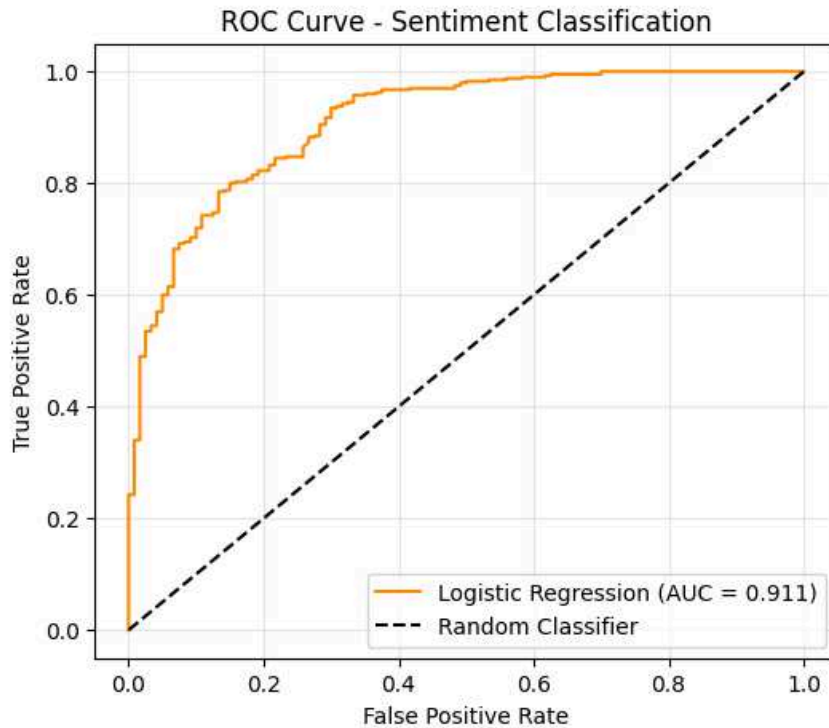
```
from sklearn.metrics import roc_curve, roc_auc_score
```

```
# Using Logistic Regression probabilities
```

```
y_probs = lr.predict_proba(X_test_tfidf)[: , 1] # probability of positive class
```

```
fpr, tpr, _ = roc_curve(y_test, y_probs, pos_label='positive')
auc = roc_auc_score(y_test, y_probs)

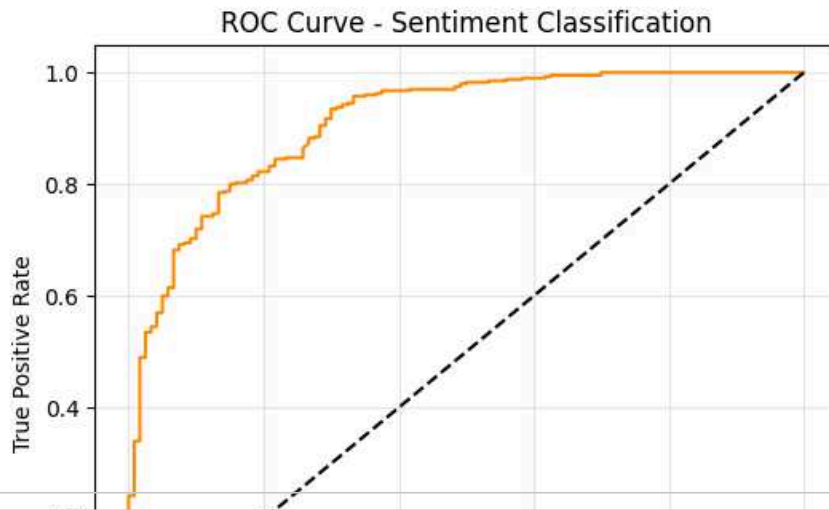
plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {auc:.3f})', color='darkorange')
plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Sentiment Classification')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



```
from sklearn.metrics import roc_curve, roc_auc_score

# Using Logistic Regression probabilities
y_probs = lr.predict_proba(X_test_tfidf)[:, 1] # probability of positive class
fpr, tpr, _ = roc_curve(y_test, y_probs, pos_label='positive')
auc = roc_auc_score(y_test, y_probs)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {auc:.3f})', color='darkorange')
plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve - Sentiment Classification')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```



```
results = pd.DataFrame({  
    'Model': ['Logistic Regression', 'Random Forest', 'SVM', 'RF + SMOTE', 'Naive Bayes'],  
    'Accuracy': [  
        accuracy_score(y_test, y_pred_lr),  
        accuracy_score(y_test, y_pred_rf),  
        , , , , ,  
    ]  
})
```