

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

```
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: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas) (2.9.0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2024.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.5.0)
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) (2024.11.2)
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 matplotlib) (1.3.0)
Requirement already satisfied: cyclor>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.10.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.55.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil) (1.17.0)
```

```
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.csv

```
import pandas as pd
df = pd.read_csv('unbalanceddataset.csv') # غير الاسم إذا كان مختلفاً
print("تم تحميل الملف بنجاح")
print(f"عدد العينات: {len(df)}")
df.head()
```

تم تحميل الملف بنجاح
عدد العينات: 2600

1 to 5 of 5 entries [Filter](#) [Table](#) [?](#)

index	text	sentiment
0	Java Concurrency in Practice is probably the best Java book I've ever bought. There's a recipe in there for interrupting blocking IO ops	positive
1	haha aww hun i bet you are more creative than me	positive
2	_pickle lol, thank you very much Hope you're having a great day!	positive
3	Out for an evening on the town with jeremy. Sad Carrie can't come	negative
4	- just took over the #1 Most Endorsed spot on twindexx.com - thanks to the endorsement by	positive

Show per page

Like what you see? Visit the [data table notebook](#) to learn more about interactive tables.

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
print("توزيع المشاعر:")
print(df['sentiment'].value_counts())
```

```
توزيع المشاعر:
sentiment
positive    2000
```

```
negative      600
Name: count, dtype: int64
```

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

# تحميل كل الحزم المطلوبة
nltk.download('punkt')
nltk.download('punkt_tab') # <-- الحزمة الجديدة المطلوبة
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)
    text = re.sub(r'^a-zA-Z\s', '', text)
    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("تم تنظيف النصوص")
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] Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
تم تنظيف النصوص
```

	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"حجم التدريب: {len(X_train)}")
print(f"حجم الاختبار: {len(X_test)}")
print("توزيع التدريب:\n", y_train.value_counts())
```

```
حجم التدريب: 2080
حجم الاختبار: 520
توزيع التدريب:
sentiment
positive    1600
negative     480
Name: count, dtype: int64
```

```

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)

print(f"شكل بيانات التدريب: {X_train_vec.shape}")

```

شكل بيانات التدريب: (4020 , 2080)

```

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

model = LogisticRegression(max_iter=1000)
model.fit(X_train_vec, y_train)

y_pred = model.predict(X_test_vec)

print("✅ Logistic Regression")
print(f"الدقة : {accuracy_score(y_test, y_pred):.4f}")
print("\n\tقرير التصنيف:\n", classification_report(y_test, y_pred))

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

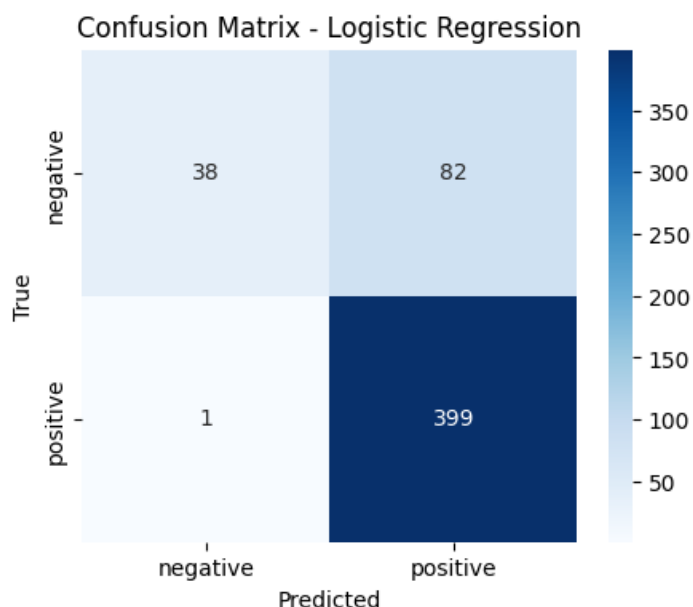
```

✔ Logistic Regression

الدقة: 0.8404

تقرير التصنيف:

	precision	recall	f1-score	support
negative	0.97	0.32	0.48	120
positive	0.83	1.00	0.91	400
accuracy			0.84	520
macro avg	0.90	0.66	0.69	520
weighted avg	0.86	0.84	0.81	520



```
from sklearn.ensemble import RandomForestClassifier

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

y_pred_rf = rf.predict(X_test_vec)

print("✔ Random Forest")
print(f"الدقة: {accuracy_score(y_test, y_pred_rf):.4f}")
print("\n\tقرير التصنيف:\n", classification_report(y_test, y_pred_rf))

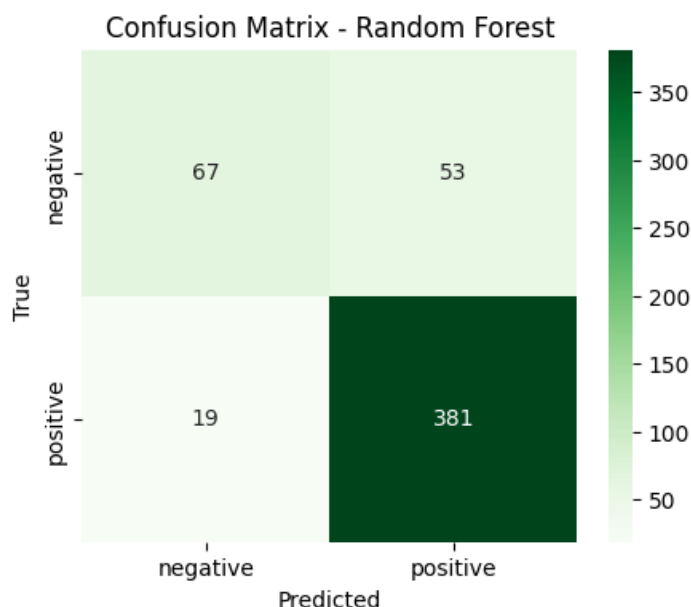
# مصفوفة الارتباك
cm_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(5,4))
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Greens', xticklabels=rf.classes_, yticklabels=rf.classes_)
plt.title("Confusion Matrix - Random Forest")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```

✓ Random Forest

الدقة: 0.8615

تقرير التصنيف:

	precision	recall	f1-score	support
negative	0.78	0.56	0.65	120
positive	0.88	0.95	0.91	400
accuracy			0.86	520
macro avg	0.83	0.76	0.78	520
weighted avg	0.86	0.86	0.85	520



```
results = pd.DataFrame({
    'Model': ['Logistic Regression', 'Random Forest'],
    'Accuracy': [
        accuracy_score(y_test, y_pred),
        accuracy_score(y_test, y_pred_rf)
    ]
})
print(results)
```

	Model	Accuracy
0	Logistic Regression	0.840385
1	Random Forest	0.861538

```
from sklearn.feature_extraction.text import CountVectorizer

# بدلاً من TF-IDF استخدام 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"شكل بيانات التدريب باستخدام CountVectorizer: {X_train_count.shape}")

CountVectorizer: (2080, 5000)
```

```
from sklearn.svm import SVC

# تدريب SVM
svm = SVC(kernel='linear', random_state=42, probability=True) # ROC لمنحنى
svm.fit(X_train_count, y_train) # TF-IDF نستخدم أو يمكنك استخدام CountVectorizer
```

```

y_pred_svm = svm.predict(X_test_vec)

print("✅ SVM (Support Vector Machine)")
print(f"الدقة: {accuracy_score(y_test, y_pred_svm):.4f}")
print("\n\tقرير التصنيف:\n", classification_report(y_test, y_pred_svm))

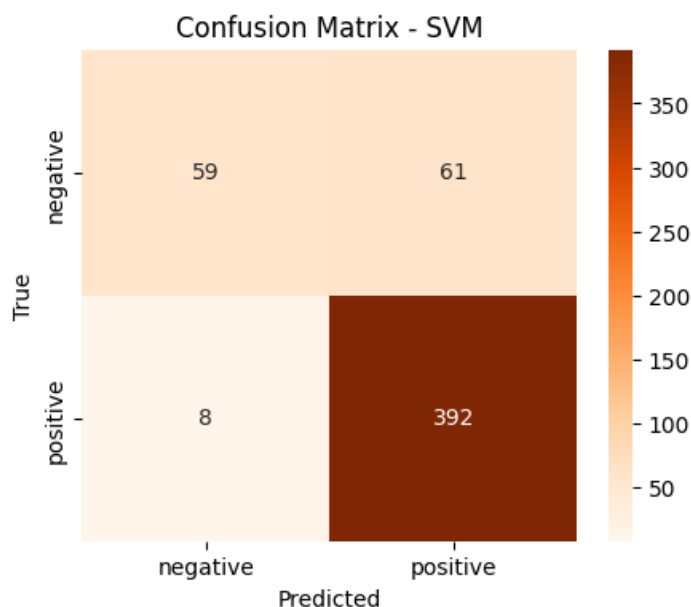
# مصفوفة الارتباك
cm_svm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(5,4))
sns.heatmap(cm_svm, annot=True, fmt='d', cmap='Oranges', xticklabels=svm.classes_, yticklabels=svm.classe
plt.title("Confusion Matrix - SVM")
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()

```

✅ SVM (Support Vector Machine)
الدقة: 0.8673

قرير التصنيف:

	precision	recall	f1-score	support
negative	0.88	0.49	0.63	120
positive	0.87	0.98	0.92	400
accuracy			0.87	520
macro avg	0.87	0.74	0.78	520
weighted avg	0.87	0.87	0.85	520



```

from imblearn.over_sampling import SMOTE
from collections import Counter

# (TF-IDF) على بيانات التدريب SMOTE تطبيق
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_vec, y_train)

print("SMOTE توزيع الفئات بعد:", Counter(y_train_smote))

# على البيانات المتوازنة Random Forest تدريب
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_vec)

```

```
print("✅ Random Forest بعد SMOTE")
print(f"الدقة: {accuracy_score(y_test, y_pred_smote):.4f}")
print("\n\tقرير التصنيف:\n", classification_report(y_test, y_pred_smote))

# مصفوفة الارتباك
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()
```

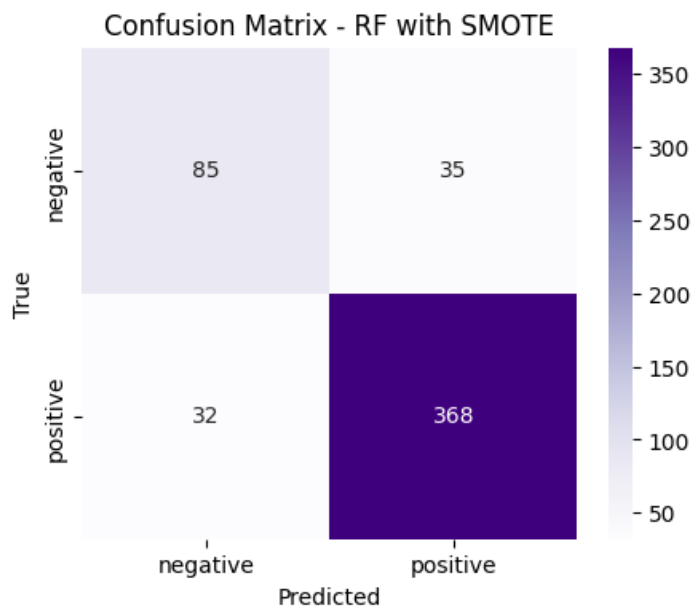
بعد SMOTE توزيع الفئات Counter({'positive': 1600, 'negative': 1600})

✅ Random Forest بعد SMOTE

الدقة: 0.8712

قرير التصنيف:

	precision	recall	f1-score	support
negative	0.73	0.71	0.72	120
positive	0.91	0.92	0.92	400
accuracy			0.87	520
macro avg	0.82	0.81	0.82	520
weighted avg	0.87	0.87	0.87	520



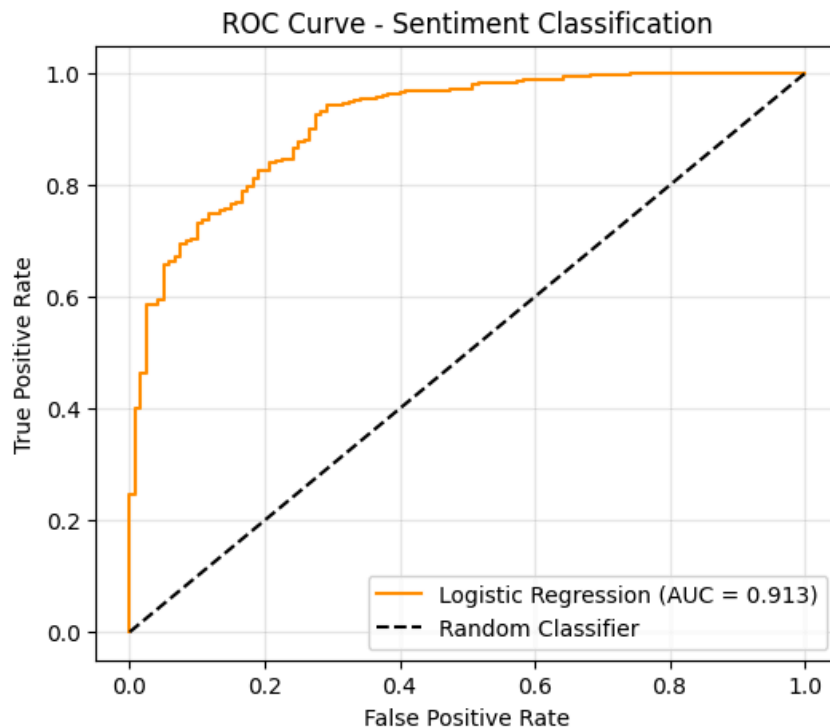
```
from sklearn.metrics import roc_curve, roc_auc_score
```

```
# نحتاج إلى احتمالات التنبؤ (وليس الفئات فقط)
# Logistic Regression تلقائياً يقدم predict_proba
lr_probs = model.predict_proba(X_test_vec)[: , 1] # احتمالات الفئة الموجبة
```

```
# حساب منحنى ROC
fpr, tpr, thresholds = roc_curve(y_test, lr_probs, pos_label='positive')
auc = roc_auc_score(y_test, lr_probs)
```

```
# رسم منحنى ROC
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.naive_bayes import MultinomialNB

nb = MultinomialNB()
nb.fit(X_train_vec, y_train)
y_pred_nb = nb.predict(X_test_vec)

print("✅ Naive Bayes")
print(f"الدقة: {accuracy_score(y_test, y_pred_nb):.4f}")
print("\n\tقرير التصنيف:\n", classification_report(y_test, y_pred_nb))
```

✅ Naive Bayes
الدقة: 0.8019

تقرير التصنيف:

	precision	recall	f1-score	support
negative	1.00	0.14	0.25	120
positive	0.80	1.00	0.89	400
accuracy			0.80	520
macro avg	0.90	0.57	0.57	520
weighted avg	0.84	0.80	0.74	520