

```
In [1]: import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns

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

```
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\B.O.S\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\B.O.S\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

Out[1]: True

```
In [2]: import pandas as pd
df = pd.read_csv('nlpdataset.csv')
df.head()
```

Out[2]:

	text	Sentiment
0	حقوق المرأة ❤️❤️❤️ https://t.co/Mzf90Ta5g1	1
1	RT @__IHAVENOIDEA: حقوق المرأة في الإسلام. ht...	1
2	RT @saud_talep: Retweeted بشيرا للتنمية (...)	1
3	RT @MojKsa: ... حقوق المرأة التي تضمنها لها وزارة	1
4	RT @abm112211: ...ولي امر الزوجة او ولي الزوجة او	1

```
In [4]: import re
import nltk
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('stopwords')

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

def preprocess_arabic_text(text):
    # إزالة التشكيل
    text = re.sub(r'[\u0617-\u061A\u064B-\u0652]', '', text)

    # توحيد الحروف
```

```
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\B.O.S\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data] C:\Users\B.O.S\AppData\Roaming\nltk_data...
[nltk_data] Unzipping tokenizers\punkt_tab.zip.
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\B.O.S\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

```
X = df["clean_text"]
y = df["Sentiment"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=42, stratify=y)

print("X_train shape:",X_train.shape)
print("X_test shape:",X_test.shape)
print("y_train shape:",y_train.shape)
print("y_test shape:",y_test.shape)
```

```
bow = CountVectorizer()
X_train_bow = bow.fit_transform(X_train)
X_test_bow = bow.transform(X_test)
print("X_train_bow shape:", X_train_bow.shape)
print("X_test_bow shape:", X_test_bow.shape)
```

2/4

```
In [19]: import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(y_true, y_pred, title):
    cm = confusion_matrix(y_true, y_pred)

    plt.figure(figsize=(5,4))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title(title)
    plt.show()
```

```
In [26]: from sklearn.feature_extraction.text import TfidfVectorizer

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

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# إنشاء الموديل
lr_model = LogisticRegression(max_iter=1000)

# تدريب الموديل
lr_model.fit(X_train_vec, y_train)

# التوقع
y_pred_lr = lr_model.predict(X_test_vec)

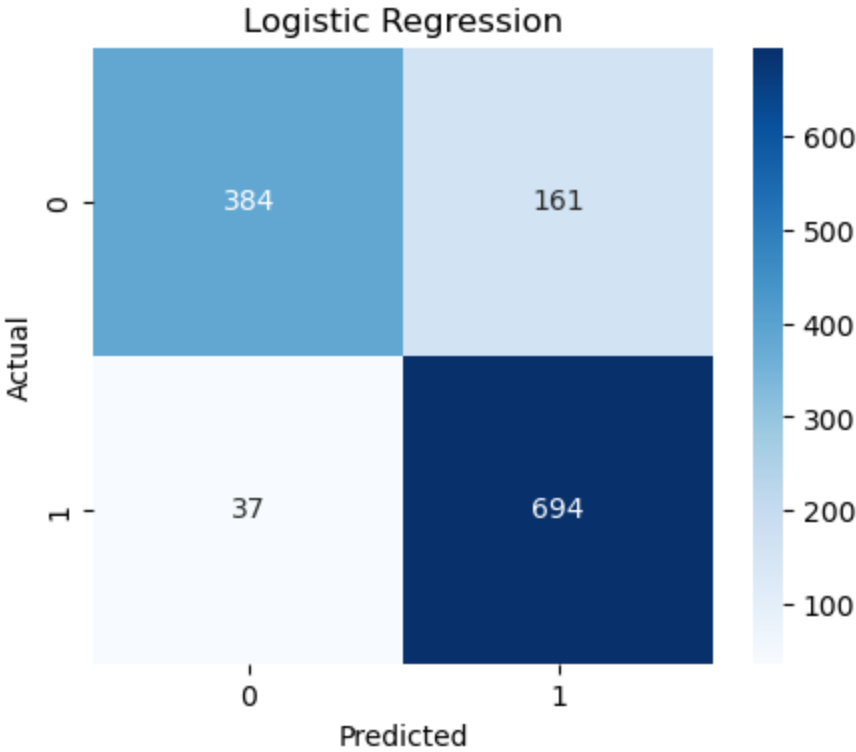
# التقييم
print("accuracy:", accuracy_score(y_test, y_pred_lr))
print("precision:", precision_score(y_test, y_pred_lr, average='weighted'))
print("recall:", recall_score(y_test, y_pred_lr, average='weighted'))
print("f1_score:", f1_score(y_test, y_pred_lr, average='weighted'))

print("\nclassification_report:")
print(classification_report(y_test, y_pred_lr))

# رسم Confusion Matrix
plot_confusion_matrix(y_test, y_pred_lr, "Logistic Regression")
```

accuracy: 0.8448275862068966
precision: 0.8545860857128214
recall: 0.8448275862068966
f1_score: 0.8409342887554601

classification_report:				
	precision	recall	f1-score	support
0	0.91	0.70	0.80	545
1	0.81	0.95	0.88	731
accuracy			0.84	1276
macro avg	0.86	0.83	0.84	1276
weighted avg	0.85	0.84	0.84	1276



In []: