

```
from google.colab import files
```

```
uploaded = files.upload()
```

[Choose Files](#) unbalanceddataset.csv

unbalanceddataset.csv(text/csv) - 212273 bytes, last modified: 8/6/2026 - 100% done
Saving unbalanceddataset.csv to unbalanceddataset.csv

```
import re
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score, f1_score
```

```
# 1. قراءة البيانات
df = pd.read_csv('unbalanceddataset.csv')
```

```
# 2. تنظيف ومعالجة النصوص
STOP_WORDS = set([
    "a", "about", "above", "after", "again", "against", "all", "am", "an", "and", "any", "are", "aren't",
    "as", "at", "be", "because", "been", "before", "being", "below", "between", "both", "but", "by", "can't",
    "cannot", "could", "couldn't", "did", "didn't", "do", "does", "doesn't", "doing", "don't", "down", "during",
    "each", "few", "for", "from", "further", "had", "hadn't", "has", "hasn't", "have", "haven't", "having",
    "he", "he'd", "he'll", "he's", "her", "here", "here's", "hers", "herself", "him", "himself", "his", "how",
    "how's", "i", "i'd", "i'll", "i'm", "i've", "if", "in", "into", "is", "isn't", "it", "it's", "its", "itself",
    "let's", "me", "more", "most", "mustn't", "my", "myself", "no", "nor", "not", "of", "off", "on", "once",
    "only", "or", "other", "ought", "our", "ours", "ourselves", "out", "over", "own", "same", "shan't", "she",
    "she'd", "she'll", "she's", "should", "shouldn't", "so", "some", "such", "than", "that", "that's", "the",
    "their", "theirs", "them", "themselves", "then", "there", "there's", "these", "they", "they'd", "they'll",
    "they're", "they've", "this", "those", "through", "to", "too", "under", "until", "up", "very", "was",
    "wasn't", "we", "we'd", "we'll", "we're", "we've", "were", "weren't", "what", "what's", "when", "when's",
    "where", "where's", "which", "while", "who", "who's", "whom", "why", "why's", "with", "won't", "would",
    "wouldn't", "you", "you'd", "you'll", "you're", "you've", "your", "yours", "yourself", "yourselves"
])
```

```
def preprocess(text):
    if not isinstance(text, str):
        return ""
    text = text.lower()
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    text = re.sub(r'@\w+|\#\w+', '', text)
    text = re.sub(r'^a-zA-Z\s$', '', text)
    tokens = text.split()
    tokens = [word for word in tokens if word not in STOP_WORDS and len(word) > 1]
    return ' '.join(tokens)
```

```
df['clean_text'] = df['text'].apply(preprocess)
df['label'] = df['sentiment'].map({'negative': 0, 'positive': 1})
```

```
# 3. تقسيم البيانات (80% تدريب / 20% اختبار)
X_train_raw, X_test_raw, y_train, y_test = train_test_split(
    df['clean_text'], df['label'], test_size=0.2, random_state=42, stratify=df['label']
)
```

```
# 4. تمثيل الخصائص (Bag of Words & TF-IDF)
cv = CountVectorizer(max_features=5000)
X_train_cv = cv.fit_transform(X_train_raw)
X_test_cv = cv.transform(X_test_raw)

tfidf = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf.fit_transform(X_train_raw)
X_test_tfidf = tfidf.transform(X_test_raw)
```

```
# 5. تدريب وتقييم النماذج
models = {
    'Naive Bayes': MultinomialNB(),
```

```

'Logistic Regression': LogisticRegression(class_weight='balanced', random_state=42),
'Random Forest': RandomForestClassifier(class_weight='balanced', random_state=42)
}

results = []
for name, model in models.items():
    # Bag of Words
    model.fit(X_train_cv, y_train)
    preds_cv = model.predict(X_test_cv)
    acc_cv = accuracy_score(y_test, preds_cv)
    f1_cv = f1_score(y_test, preds_cv, average='macro')
    results.append({'Model': name, 'Feature': 'Bag of Words', 'Accuracy': f"{acc_cv:.4f}", 'F1-Macro': f"{f1_cv:.4f}"})

    # TF-IDF
    model.fit(X_train_tfidf, y_train)
    preds_tfidf = model.predict(X_test_tfidf)
    acc_tfidf = accuracy_score(y_test, preds_tfidf)
    f1_tfidf = f1_score(y_test, preds_tfidf, average='macro')
    results.append({'Model': name, 'Feature': 'TF-IDF', 'Accuracy': f"{acc_tfidf:.4f}", 'F1-Macro': f"{f1_tfidf:.4f}"})

# عرض النتائج في جدول
results_df = pd.DataFrame(results)
print("=== جدول نتائج التقييم ===")
print(results_df.to_string(index=False))

```

```

=== جدول نتائج التقييم ===

```

Model	Feature	Accuracy	F1-Macro
Naive Bayes	Bag of Words	0.8538	0.7328
Naive Bayes	TF-IDF	0.8000	0.5601
Logistic Regression	Bag of Words	0.8558	0.8082
Logistic Regression	TF-IDF	0.8519	0.8031
Random Forest	Bag of Words	0.8654	0.7866
Random Forest	TF-IDF	0.8596	0.7782