

Start coding or [generate](#) with AI.

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)

Library install

```
import pandas as pd
import numpy as np
import nltk
import sklearn
import tensorflow as tf

print("Pandas:", pd.__version__)
print("NumPy:", np.__version__)
print("NLTK:", nltk.__version__)
print("Scikit-learn:", sklearn.__version__)
print("TensorFlow:", tf.__version__)
```

Pandas: 2.2.2
NumPy: 2.0.2
NLTK: 3.9.1
Scikit-learn: 1.6.1
TensorFlow: 2.19.0

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

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

Upload files

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

اختيار ملفات 2 من الملفات

Fake.csv(text/csv) - 62789876 bytes, last modified: 100% - ٢٠٢٥/٤/٢٨ done
True.csv(text/csv) - 53582940 bytes, last modified: 100% - ٢٠٢٥/٤/٢٨ done
Saving Fake.csv to Fake (1).csv
Saving True.csv to True (1).csv

Reading and merging files

```
import pandas as pd

df_true = pd.read_csv("True.csv")
df_fake = pd.read_csv("Fake.csv")

df_true["label"] = 1 # Real
df_fake["label"] = 0 # Fake

df = pd.concat([df_true, df_fake], ignore_index=True)

df.head()
```

	title	text	subject	date	label	
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017	1	
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017	1	
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017	1	
3	FBI Russia probe helped by Australian	WASHINGTON (Reuters) - Trump campaign	...	December 30,	1	

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

Verification

```
df.info()
df['label'].value_counts()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44898 entries, 0 to 44897
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ---
 0   title       44898 non-null  object
 1   text        44898 non-null  object
 2   subject     44898 non-null  object
 3   date        44898 non-null  object
 4   label       44898 non-null  int64
dtypes: int64(1), object(4)
memory usage: 1.7+ MB
```

count

	label
0	23481
1	21417

dtype: int64

```
import nltk
nltk.download('punkt_tab')
```

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

Text Preprocessing

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

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

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

def clean_text(text):

    text = text.lower()

    text = re.sub(r'http\S+|www\S+|https\S+', '', text)

    text = re.sub(r'^\w\s', '', text)

    text = re.sub(r'\d+', '', text)

    text = re.sub(r'^a-zA-Z\s', '', text)

    tokens = nltk.word_tokenize(text)

    tokens = [word for word in tokens if word not in stop_words]

    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    text = " ".join(tokens)

    return text
```

```
df["clean_text"] = df["text"].apply(clean_text)
```

```
df[["text", "clean_text"]].head()
```

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

	text	clean_text	
0	WASHINGTON (Reuters) - The head of a conservat...	washington reuters head conservative republica...	
1	WASHINGTON (Reuters) - Transgender people will...	washington reuters transgender people allowed ...	
2	WASHINGTON (Reuters) - The special counsel inv...	washington reuters special counsel investigati...	
3	WASHINGTON (Reuters) - Trump campaign adviser ...	washington reuters trump campaign adviser geor...	
4	SEATTLE/WASHINGTON (Reuters) - President Donal...	seattlewashington reuters president donald tru...	

Train/Test Split

```
from sklearn.model_selection import train_test_split
```

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

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

```
print("Training samples:", len(X_train))
print("Testing samples:", len(X_test))
```

```
Training samples: 35918
Testing samples: 8980
```

Verify the division

```
print(y_train.value_counts())
print(y_test.value_counts())
```

```
label
0    18785
1    17133
Name: count, dtype: int64
label
0     4696
1     4284
Name: count, dtype: int64
```

Feature Representation using TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
vectorizer = TfidfVectorizer(  
    max_features=20000,  
    ngram_range=(1,2),  
    stop_words='english'  
)
```

```
X_train_tfidf = vectorizer.fit_transform(X_train)
```

```
X_test_tfidf = vectorizer.transform(X_test)
```

```
print("TF-IDF train shape:", X_train_tfidf.shape)  
print("TF-IDF test shape:", X_test_tfidf.shape)
```

```
TF-IDF train shape: (35918, 20000)  
TF-IDF test shape: (8980, 20000)
```

Logistic Regression Model Training

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

```
log_reg = LogisticRegression(max_iter=2000)
```

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

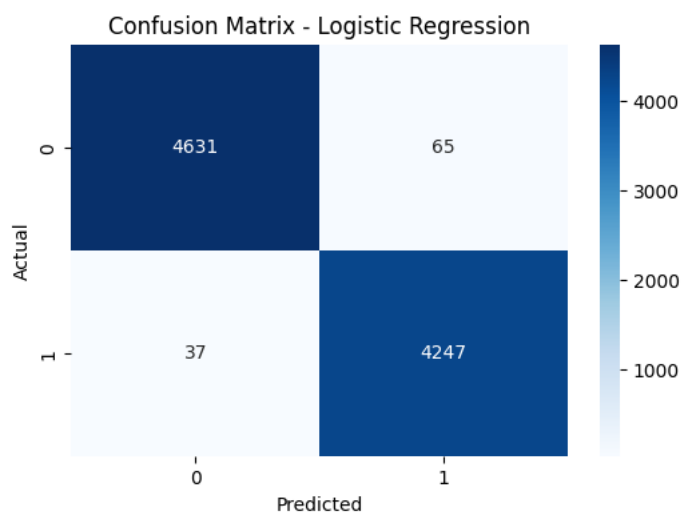
```
y_pred_lr = log_reg.predict(X_test_tfidf)
```

```
print("Classification Report:")  
print(classification_report(y_test, y_pred_lr))
```

```
cm = confusion_matrix(y_test, y_pred_lr)
```

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

```
Classification Report:  
              precision    recall  f1-score   support  
  
    0       0.99         0.99         0.99         4696  
    1       0.98         0.99         0.99         4284  
  
accuracy          0.99         0.99         0.99         8980  
macro avg         0.99         0.99         0.99         8980  
weighted avg         0.99         0.99         0.99         8980
```



Tokenization + Padding

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

max_words = 30000
max_len = 300

tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)

X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

X_train_pad = pad_sequences(X_train_seq, maxlen=max_len, padding='post', truncating='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len, padding='post', truncating='post')

print("Train padded shape:", X_train_pad.shape)
print("Test padded shape:", X_test_pad.shape)
```

```
Train padded shape: (35918, 300)
Test padded shape: (8980, 300)
```

LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

embedding_dim = 128

model = Sequential()
model.add(Embedding(input_dim=max_words, output_dim=embedding_dim, input_length=max_len))
model.add(LSTM(128, return_sequences=False))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(
    loss='binary_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)

model.build(input_shape=(None, max_len))

model.summary()
```

Model: "sequential_7"

Layer (type)	Output Shape	Param #
embedding_7 (Embedding)	(None, 300, 128)	3,840,000
lstm_10 (LSTM)	(None, 128)	131,584
dropout_12 (Dropout)	(None, 128)	0
dense_14 (Dense)	(None, 64)	8,256
dropout_13 (Dropout)	(None, 64)	0
dense_15 (Dense)	(None, 1)	65

```
Total params: 3,979,905 (15.18 MB)
Trainable params: 3,979,905 (15.18 MB)
Non-trainable params: 0 (0.00 B)
```

LSTM model training

```
history = model.fit(
    X_train_pad, y_train,
    validation_split=0.2,
    epochs=5,
```

```

        batch_size=64,
        verbose=1
    )

```

```

Epoch 1/5
449/449 ————— 356s 787ms/step - accuracy: 0.6865 - loss: 0.5617 - val_accuracy: 0.8685 - val_loss: 0.3705
Epoch 2/5
449/449 ————— 393s 876ms/step - accuracy: 0.8528 - loss: 0.3741 - val_accuracy: 0.8739 - val_loss: 0.3340
Epoch 3/5
449/449 ————— 369s 823ms/step - accuracy: 0.8672 - loss: 0.3397 - val_accuracy: 0.7597 - val_loss: 0.4394
Epoch 4/5
449/449 ————— 380s 818ms/step - accuracy: 0.8781 - loss: 0.3610 - val_accuracy: 0.7255 - val_loss: 0.4269
Epoch 5/5
449/449 ————— 389s 832ms/step - accuracy: 0.9464 - loss: 0.1985 - val_accuracy: 0.9641 - val_loss: 0.1412

```

Model evaluation

```

y_pred_proba = model.predict(X_test_pad)
y_pred_lstm = (y_pred_proba > 0.5).astype("int32")

from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred_lstm))

```

```

281/281 ————— 45s 158ms/step
              precision    recall  f1-score   support

         0           0.96       0.97       0.97         4696
         1           0.97       0.95       0.96         4284

   accuracy              0.96              8980
  macro avg           0.96       0.96       0.96         8980
weighted avg           0.96       0.96       0.96         8980

```

Loss & Accuracy

```

import matplotlib.pyplot as plt

# Loss curves
plt.figure(figsize=(12,5))

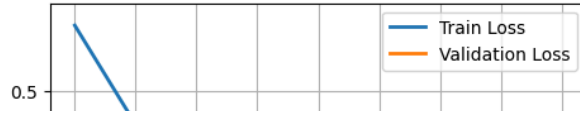
plt.subplot(1,2,1)
plt.plot(history.history['loss'], label='Train Loss', linewidth=2)
plt.plot(history.history['val_loss'], label='Validation Loss', linewidth=2)
plt.title('Training vs Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

# Accuracy curves
plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], label='Train Accuracy', linewidth=2)
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', linewidth=2)
plt.title('Training vs Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)

plt.show()

```

Training vs Validation Loss



Training vs Validation Accuracy

