Import Libraries

```
import pandas as pd
import numpy as np
import re
import string
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
```

2. Load Dataset

```
df = pd.read_csv("Fake_New_data.csv")

print(df.head())
print(df.isnull().sum())
```

```
   Unnamed: 0                                              title  \
0        8476                       You Can Smell Hillary’s Fear
1       10294  Watch The Exact Moment Paul Ryan Committed Pol...
2        3608        Kerry to go to Paris in gesture of sympathy
3       10142  Bernie supporters on Twitter erupt in anger ag...
4         875   The Battle of New York: Why This Primary Matters

                                                text label
0  Daniel Greenfield, a Shillman Journalism Fello...  FAKE
1  Google Pinterest Digg Linkedin Reddit Stumbleu...  FAKE
2  U.S. Secretary of State John F. Kerry said Mon...  REAL
3  — Kaydee King (@KaydeeKing) November 9, 2016 T...  FAKE
4  It's primary day in New York and front-runners...  REAL
Unnamed: 0    0
title         0
text          0
label         0
dtype: int64
```

```
print(df.head())
print(df.isnull().sum())
```

```
   Unnamed: 0                                              title  \
0        8476                       You Can Smell Hillary’s Fear
1       10294  Watch The Exact Moment Paul Ryan Committed Pol...
2        3608        Kerry to go to Paris in gesture of sympathy
3       10142  Bernie supporters on Twitter erupt in anger ag...
4         875   The Battle of New York: Why This Primary Matters

                                                text label
0  Daniel Greenfield, a Shillman Journalism Fello...  FAKE
1  Google Pinterest Digg Linkedin Reddit Stumbleu...  FAKE
2  U.S. Secretary of State John F. Kerry said Mon...  REAL
3  — Kaydee King (@KaydeeKing) November 9, 2016 T...  FAKE
4  It's primary day in New York and front-runners...  REAL
Unnamed: 0    0
title         0
text          0
label         0
dtype: int64
```

3. Preprocessing

```
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r"http\S+", "", text)                 # remove URLs
    text = re.sub(r"[^a-zA-Z\s]", "", text)             # remove numbers, punctuation
    text = text.translate(str.maketrans("", "", string.punctuation))
    text = re.sub(r"\s+", " ", text).strip()            # remove extra spaces
    return text

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

# Convert labels (FAKE/REAL) to 0/1
df["label"] = df["label"].map({"FAKE": 0, "REAL": 1})
```

4. Tokenization + Sequencing

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

tokenizer = Tokenizer(num_words=50000, oov_token="<OOV>")
tokenizer.fit_on_texts(X)
```

```
sequences = tokenizer.texts_to_sequences(X)
padded = pad_sequences(sequences, maxlen=300)
```

5. Train-Test Split

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

6. LSTM Model

```
model = Sequential([
    Embedding(input_dim=50000, output_dim=128, input_length=300),
    LSTM(128, return_sequences=False),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

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

model.summary()
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just re
  warnings.warn(
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding (Embedding)           │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm (LSTM)                     │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ ?                      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ ?                      │   0 (unbuilt) │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ ?                      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ ?                      │   0 (unbuilt) │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 0 (0.00 B)
```

7. Train Model

```
history = model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=5,
    batch_size=64,
    verbose=1
)
```

```
Epoch 1/5
64/64 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.5911 - loss: 0.6668 - val_accuracy: 0.7633 - val_loss: 0.4650
Epoch 2/5
64/64 ━━━━━━━━━━━━━━━━━━━━ 59s 905ms/step - accuracy: 0.8833 - loss: 0.2966 - val_accuracy: 0.8629 - val_loss: 0.3547
Epoch 3/5
64/64 ━━━━━━━━━━━━━━━━━━━━ 46s 716ms/step - accuracy: 0.9690 - loss: 0.1085 - val_accuracy: 0.8462 - val_loss: 0.4242
Epoch 4/5
64/64 ━━━━━━━━━━━━━━━━━━━━ 47s 741ms/step - accuracy: 0.9963 - loss: 0.0141 - val_accuracy: 0.8383 - val_loss: 0.6100
Epoch 5/5
64/64 ━━━━━━━━━━━━━━━━━━━━ 48s 746ms/step - accuracy: 0.9994 - loss: 0.0042 - val_accuracy: 0.8442 - val_loss: 0.6027
```

8. Evaluation

```
pred = (model.predict(X_test) > 0.5).astype("int32")

acc = accuracy_score(y_test, pred)
prec = precision_score(y_test, pred)
rec = recall_score(y_test, pred)
f1 = f1_score(y_test, pred)

print("Accuracy:", acc)
print("Precision:", prec)
print("Recall:", rec)
print("F1 Score:", f1)
```

```
40/40 ━━━━━━━━━━━━━━━━━━━━ 5s 124ms/step
Accuracy: 0.8492501973164956
Precision: 0.82
Recall: 0.8982785602503912
F1 Score: 0.8573562359970127
```

9. Make Prediction

```
def predict_news(text):
    cleaned = clean_text(text)
    seq = tokenizer.texts_to_sequences([cleaned])
    pad = pad_sequences(seq, maxlen=300)
    prob = model.predict(pad)[0][0]
    return "REAL" if prob > 0.5 else "FAKE"

print(predict_news("Breaking! Scientists discover new planet."))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 66ms/step
FAKE
```