1. **OBJECTIVE:**

To train a Machine learning model that takes a news input in Fon, Yoruba, or French and predicts whether it's **true news (1)** or **false/misleading (0)**.

1. **TOOLS AND TECHNOLOGY:**
   * + - NUMPY AND PANDAS:

* These are the foundation of data handling in Python.
* pandas makes it easy to load your CSV, access the text and label, and structure the dataset.
* numpy provides efficient array operations—and also helpful when TensorFlow deals with inputs as NumPy arrays behind the scenes.
  + - * TOKENIZER AND PAD\_SEQUENCE (from keras):
* Neural networks can’t interpret raw text—only numbers.
* Tokenizer: Converts words into integer sequences based on word frequency.
* pad\_sequences: Ensures all inputs are the same length, which is essential for batching and feeding into RNN/LSTM layers.
* TRAIN\_TEST\_SPLIT(from sklearn.model\_selection):
* Helps prevent overfitting by separating data into train and test
* **Training set**: For learning.
* **Test set**: For evaluating generalization.
* SEQUENTIAL MODEL (from Keras):
* Provides a simple way to build layer-by-layer neural networks.
* It's best when you want to build a straightforward pipeline: from text to embedding to classification.
* EMBEDDING LAYER:
* Turns each word index into a **dense vector** that captures semantic relationships.
* Example: “government” and “policy” might have similar embeddings even if their word IDs are far apart.
* BIDIRECTIONAL LSTM:
* LSTM handles sequences and remembers long-term dependencies.
* **Bidirectional** gives the model the ability to understand **context from both past and future** tokens, improving accuracy in nuanced text.
* DROPOUT LAYER:
* Regularizes the model to prevent overfitting.
* It randomly drops neurons during training so the model doesn’t rely too heavily on any one pattern.
* DENSE LAYERS:
* These layers are the decision-makers.
* The final dense layer with a sigmoid activation produces a probability between 0 and 1 — perfect for binary classification (true vs. fake).
* BINARY CROSSENTROPY LOSS:
* This loss function is specifically designed for binary classification problems.
* It measures how far off the model’s predicted probabilities are from the actual labels (0 or 1).
* ADAM OPTIMIZER:
* It’s fast, reliable, and adapts learning rates while training.
* Helps the model converge efficiently even with noisy or sparse gradient updates.

1. **MODEL PERFORMANCE EVALUATION:**

Model used is called sequential deep learning model which performs better on sequential data like texts. Below are the metrics evaluation for the model

Accuracy Score: The model got **100%** of the predictions right on the training data. That's perfect accuracy.

Loss: That’s the training loss, which is extremely low (*about 0.00025*). It indicates the model's predictions closely match the target outputs.

Val\_accuracy: 1.0000 → 100% accuracy, but this time on the validation set, which wasn’t used during training. This means the model generalizes very well.

Val\_loss: 4.6587e-05 → Also a very low validation loss. The lower the loss, the better the model is doing in terms of minimizing errors.

1. **DATASET:**

* FORMAT: CSV
* FIELDS:
* text: The input message
* label: Binary class (1 = true news, 0 = fake news)
* LANGUAGE SOURCES: Fon, Yorùbá, French
* REFERENCE:
* Web Scraping from:
* Radio France Internationale
* BBC Afrique
* ORTB (Office de Radiodiffusion et Télévision du Bénin)
* Premium Times Nigeria
* Social media.