# Fake News Detector

## Objective: Detect if a news headline is fake or real.

### Abstract

This project describes the development of a real-time Fake News Detector application designed for rapid classification of news headlines. The solution employs classical Natural Language Processing (NLP) and Machine Learning techniques, specifically **TF-IDF Vectorization** for feature extraction and **Logistic Regression** for binary classification (Real vs. Fake). The entire pipeline is wrapped in an interactive web application built with **Streamlit**, providing capabilities for single-headline prediction and batch processing via CSV uploads. The core model is trained on a small, self-contained dataset and cached for instant deployment, demonstrating a proof-of-concept for text classification in a web environment.

### 1. Introduction

The rapid propagation of misinformation and unverified content online necessitates immediate tools for content verification. This prototype focuses on establishing a functional and explainable pipeline for text classification. The application serves as a demonstrative proof-of-concept, highlighting the efficacy of using classical machine learning techniques to address this pressing issue.

**Objective:** To create a user-friendly application capable of classifying news headlines as either "REAL" (Label 1) or "FAKE" (Label 0) and displaying the model's confidence in that prediction.

### 2. Tools Used

The project relies on the following key libraries and frameworks:

- **Application Development: Streamlit** is used to create the interactive, front-end user interface and handle data input/output, file management, and results visualization.
- **Data Handling: Pandas** and **NumPy** are essential for data loading, cleaning, manipulation, and numerical array operations during vectorization and prediction.
- **Machine Learning/NLP: Scikit-learn** forms the backend for the machine learning pipeline, specifically utilizing:
    - **TfidfVectorizer:** The feature extraction tool.

- **Logistic Regression:** The binary classification algorithm.

### 3. Steps Involved in Building the Project

The detector was constructed using a four-phase pipeline: Data Preparation, Model Training, Feature Engineering, and Application Deployment.

### 3.1 Data Preparation and Model Training

1. **Dataset Initialization:** Due to environmental constraints, a minimal, hardcoded dataset of headlines and their corresponding binary labels (1: Real, 0: Fake) is loaded.
2. **Model Instantiation:** The LogisticRegression classifier is initialized for the binary classification task.
3. **Caching:** The entire training process is wrapped in Streamlit's @st.cache_resource decorator, ensuring that the computationally intensive steps run only once upon application startup.

### 3.2 Feature Engineering (TF-IDF)

The raw text headlines are transformed into numerical features using **TF-IDF (Term Frequency-Inverse Document Frequency)**.

1. **Vectorizer Configuration:** TfidfVectorizer is initialized to automatically remove English stop words (e.g., "a," "the") and filter out overly common terms (max_df=0.7).
2. **Fitting and Transformation:** The vectorizer is fitted exclusively on the training corpus and then used to transform the training data into a sparse numerical matrix. This fitted vectorizer is saved alongside the model for future use.

### 3.3 Application Interface and Prediction

The Streamlit UI provides two primary functionalities:

1. **Single Headline Analysis:**
    - The user inputs text.
    - The input is converted to a vector using the **saved TF-IDF Vectorizer**.
    - The vector is fed to the **trained Logistic Regression model** to obtain a prediction (Real/Fake) and its associated probability (confidence).

- The result is presented with visual cues (colors and icons).

2. **Batch Processing:**
   - The user uploads a CSV file.
   - The application automatically detects the headline column.
   - The batch_predict function iterates through the headlines, applies the classification pipeline, and adds new columns for the **Prediction** and **Confidence** to the DataFrame.
   - The classified data is displayed and made available for download via a dedicated CSV export button.

**4. Conclusion**

The Streamlit Fake News Detector successfully implements a functional, full-stack text classification application. The use of robust, classical NLP techniques (TF-IDF) combined with a simple, high-performing classifier (Logistic Regression) provides a fast and effective foundation for distinguishing between real and fake news based on lexical signals.

While the application is functionally sound and demonstrates a complete pipeline, its current predictive power is limited by the training data's small scale and scope. Future enhancements should focus on integrating massive, real-world datasets and exploring advanced deep learning models to capture better contextual and semantic nuances inherent in sophisticated fake news.