Untitled

June 27, 2025

1 Project Report: News Article Classification (Fake/Real)

I. Introduction In an era saturated with information, the ability to discern between genuine and fabricated news has become increasingly critical. "Fake news" can propagate misinformation, influence public opinion, and erode trust in traditional media. This project aims to address this challenge by developing a machine learning model capable of automatically classifying news articles as either fake or real. Leveraging Natural Language Processing (NLP) techniques, this system will analyze textual content to identify patterns indicative of authenticity, providing a valuable tool for content verification.

II. Abstract This report details the development of a Fake News Classification system using a Logistic Regression model trained on TF-IDF vectorized textual data. The project involved loading and combining a comprehensive dataset of fake and real news articles, rigorous text preprocessing using NLTK to clean and standardize the content, and feature extraction via TF-IDF to convert text into numerical representations. A Logistic Regression classifier was then trained and evaluated on the processed data. The model achieved an accuracy of approximately 98.89%, demonstrating high precision, recall, and F1-scores for both classes. Furthermore, an interactive web application was built using Streamlit, allowing users to input news articles and receive real-time predictions along with confidence scores and explanations, showcasing a complete end-to-end machine learning solution.

III. Tools Used Python: The primary programming language for all development.

Pandas: Utilized for efficient data loading, manipulation, and combination of the news datasets.

NLTK (Natural Language Toolkit): Employed for fundamental text preprocessing operations, specifically stop word removal.

Scikit-learn: A comprehensive machine learning library used for:

TfidfVectorizer: For converting raw text into numerical TF-IDF features.

train_test_split: For dividing data into training and testing sets.

LogisticRegression: The chosen classification algorithm for its effectiveness and interpretability.

accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report: For robust model evaluation.

Matplotlib & Seaborn: Used for data visualization, particularly for generating the confusion matrix heatmap to visually assess model performance.

Numpy: For numerical operations, especially within data manipulation and vectorization.

pickle: Python's standard library for serializing and deserializing Python object structures, used to save and load the trained model and vectorizer.

Streamlit: An open-source app framework for machine learning engineers, used to create the interactive web-based demo for the classifier.

IV. Steps Involved in Building the Project The project was developed following a structured machine learning pipeline:

Data Collection and Combination:

The project began by acquiring two distinct datasets, Fake.csv and True.csv, from Kaggle.

Each dataset was loaded into a Pandas DataFrame, and a new label column was added (0 for fake, 1 for true).

These two DataFrames were then concatenated to form a single, unified df_news dataset.

Crucially, the combined dataset was shuffled to ensure random distribution of fake and real articles, preventing positional bias during subsequent data splitting. Initial data inspection confirmed no missing values, simplifying preprocessing.

Text Preprocessing (NLTK):

To prepare the textual data for analysis, the title and text columns were merged into a full_text column.

A custom text cleaning function was applied to this column, performing several key operations: converting text to lowercase, removing URLs, HTML tags, punctuation, numbers embedded in words, and standard English stop words. This step significantly reduced noise and standardized the text.

Text Vectorization (TF-IDF):

Machine learning models require numerical input. Thus, the cleaned text was transformed into numerical features using the TfidfVectorizer.

The TfidfVectorizer was configured to extract the top 10,000 most relevant features (max_features=10000) and to ignore terms appearing in fewer than 5 documents (min_df=5), focusing on words with significant discriminative power. The output was a sparse matrix X_tfidf.

Model Training (Logistic Regression):

The vectorized features (X_tfidf) and the corresponding labels (y) were split into training (80%) and testing (20%) sets using train_test_split. The stratify=y parameter ensured that the class distribution (fake/real) was maintained in both subsets.

A LogisticRegression model, a robust linear classifier, was initialized with max_iter=1000 to ensure convergence, and then trained on the X_train and y_train data.

Model Evaluation:

The performance of the trained LogisticRegression model was rigorously evaluated on the unseen X_test dataset.

Key metrics reported included overall Accuracy (0.9889), Precision, Recall, and F1-Score for both 'fake' and 'true' classes, all of which were approximately 0.99, indicating high model effectiveness.

A Confusion Matrix was generated and visualized as a heatmap, providing a detailed breakdown of True Positives, True Negatives, False Positives, and False Negatives, clearly showing the minimal number of misclassifications.

Model Persistence:

To enable the deployment of the classifier, the trained LogisticRegression model and the fitted TfidfVectorizer were saved to disk using Python's pickle module (logistic_regression_model.pkl and tfidf_vectorizer.pkl). This ensures that the model can be loaded and used for predictions without requiring retraining.

Interactive Web Application (Streamlit Deployment):

An interactive web application (fake_news_app.py) was developed using the Streamlit framework.

This application loads the saved model and vectorizer.

It provides a user-friendly interface with a text area for users to input news articles.

Upon user submission, the app preprocesses the text, vectorizes it using the loaded TfidfVectorizer, makes a prediction with the LogisticRegression model, and displays the classification outcome (FAKE or TRUE) along with confidence scores and a concise explanation. This creates a "live web demo" as required.

V. Conclusion This project successfully developed a robust and highly accurate Fake News Classification system. By meticulously following a standard machine learning workflow—encompassing data preparation, text preprocessing, feature engineering, model training, evaluation, and deployment—a Logistic Regression model was developed that can reliably distinguish between fake and real news articles. The impressive accuracy (approx. 98.89%) and strong performance metrics highlight the model's efficacy. The creation of an interactive Streamlit application demonstrates practical deployment skills, making the model accessible and usable. This project provides valuable experience in Natural Language Processing and supervised learning, crucial for understanding and combating misinformation in digital media.