Misinformation Classifier Deployment

Om Patel Nirmal Nelson

**Description**:

The primary objective of this project is to develop and deploy a robust machine learning model capable of classifying news articles as either **"misinformation"** or **"trustworthy."** The goal includes creating a fully operational **web application** using Flask that allows end-users to input text and receive an immediate classification and confidence score.

**Significance**:

The proliferation of misinformation (or "fake news") is a significant societal and political challenge. This project offers a practical, automated tool to assist individuals and platforms in quickly vetting text sources, thereby contributing to **digital literacy** and promoting a more informed public discourse.

While news classification is a common task, the project's novelty lies in its **end-to-end integration**, combining a high-performance text classification pipeline (TF-IDF + Logistic Regression) with a lightweight, user-friendly **Flask web deployment**. This demonstrates a complete proof-of-concept for moving from raw data to a scalable, real-time prediction service.

**Code Structure:**

The project utilizes a modular structure for clear separation of concerns: data handling, model training, and web deployment.

| Directory/File | Description |
|---|---|
| misinfo_train.py | **Model Development Script.** Handles data loading, cleaning, feature engineering (TF-IDF), model training (Logistic Regression), evaluation, and saving the final pipeline. |
| misinfo_app.py | **Flask Deployment Script.** Loads the pre-trained model, defines the web routes, handles user input, performs real-time classification, and renders the results. |
| data/ | Contains the raw input datasets (True.csv, Fake.csv). |
| models/ | Stores the serialized, trained machine learning pipeline (misinfo_model.pkl). |
| webapp/templates/ | Stores the HTML templates (e.g., index.html) required by the Flask application. |

**Functionalities and Test Results:**

The project delivers core functionalities across its training and deployment phases:

| Functionality | Simple Description | Verification Result |
|---|---|---|
| **Data Preparation** | Loads, labels, combines, and shuffles the raw "True" and "Fake" datasets. | **Verification:** Combined dataset size confirmed (e.g., 44,898 entries). Labels ("trustworthy" and "misinfo") correctly assigned. |
| **Text Cleaning** | Removes URLs, punctuation, and standardizes text before training/prediction. | **Verification:** Input: "Read this: https://bad.link! It's BIG." Output: "read this it s big" |
| **Model Training** | Builds a classification pipeline and fits it to the training data. | **Verification:** Training completed successfully in a few seconds. |
| **Real-time Prediction** | The Flask app accepts text and returns a predicted label. | **Verification:** Entering a clearly fake headline yields **"misinfo"** prediction. |
| **Probability Output** | Returns the confidence level for the prediction. | **Verification:** A highly confident prediction shows probabilities near 98% for one class. |

**Data Collection**

The project utilizes the widely-used **Fake and Real News Dataset** from Kaggle, which is optimal for binary classification tasks. It provides a large, balanced set of articles sourced from credible news outlets and known unreliable sources. The data was acquired by downloading two separate CSV files, True.csv and Fake.csv. True.csv: ~21,417 articles

Fake.csv: ~23,481 articles **Total:** ~44,898 articles. Each article contains **title**, **text**, **subject**, and **date**. The key features used are the **title** and **text**, which are combined into a single **content** feature for the model.

**Data Processing and Feature Engineering:**

**Data Preprocessing**

1. **Labeling:** DataFrames were loaded and explicitly labeled: True.csv received the **"trustworthy"** label, and Fake.csv received the **"misinfo"** label.
2. **Data Consolidation and Shuffling:** The two sets were combined and randomly **shuffled** to ensure uniform distribution of classes during training and testing.
3. **Content Creation:** The **title** and **text** columns were concatenated to form a single input feature called **content**.
4. **Text Cleaning:** The custom clean_text function was applied to the content feature, performing URL removal, punctuation stripping, and lowercasing.

**Feature Engineering**

1. **Train/Test Split:** The dataset was split into **80% for training** and **20% for testing**.
2. **TF-IDF Vectorization:** A **Term Frequency-Inverse Document Frequency (TF-IDF)** vectorizer was used as the feature extractor, integrated into the scikit-learn pipeline.
    - TF-IDF converts text documents into a numerical feature matrix.
    - **Key parameters used:**
        - stop_words="english": Removes common English words.

- max_features=30000: Limits the vocabulary size to the most distinguishing 30,000 terms.
- ngram_range=(1, 2): Captures both single words (unigrams) and two-word phrases (bigrams) to preserve local context.

**Model Development (10%)**

- **Model Input and Output:**
  - **Input:** The cleaned text, transformed by the TF-IDF vectorizer into a **30,000-dimensional sparse feature vector**.
  - **Output:** The predicted class label: **"misinfo"** or **"trustworthy"**.
- **Type of Algorithm:** The algorithm chosen is **Logistic Regression** (LogisticRegression), which is a linear model highly effective for high-dimensional, sparse text data. It offers fast training and strong baseline performance.
- **Test Results:**

| Metric | Misinfo | Trustworthy | Overall Accuracy |
|---|---|---|---|
| **Precision** | 0.98 | 0.99 | **0.985** |
| **Recall** | 0.99 | 0.98 | |
| **F1-Score** | 0.98 | 0.98 | |

**Discussion**:

The project successfully developed and deployed a text classification pipeline achieving high predictive accuracy using a Logistic Regression model on TF-IDF features. The end-to-end implementation demonstrates a robust solution for automated misinformation detection. Model Robustness: The model is linear and may struggle with highly nuanced text, such as sophisticated satire or propaganda that mimics credible writing styles. Preprocessing Maintenance: The preprocessing steps, especially the cleaning function, must be kept perfectly consistent between the training script and the deployment script to prevent prediction errors. The project applied several core concepts: Supervised Learning, advanced Feature Engineering (TF-IDF and -grams), Model Evaluation (using classification metrics), and Web Deployment (using the Flask framework for a practical demonstration).