Email Spam Detection

Comprehensive Project Documentation

Generated: 2025-06-29 15:41

Project Overview

merge.py This Python code merges four CSV files (combined_data.csv, emails.csv, spam_assassin.csv, spam.csv) into a single Pandas DataFrame. Each file contains text data and a target variable (renamed to 'target' for consistency). The code preprocesses the data, mapping categories to numerical values where necessary, and then prints the merged DataFrame's column names and size before returning it. **predict_custom.py** This Python script uses a pre-trained model (`spam_model.pkl`) and vectorizer (`tfidf_vectorizer.pkl`) to classify user-input email text as spam or not spam. It cleans the input text, vectorizes it, and then makes a prediction using the loaded model. **preprocess.py** This Python file preprocesses text data for machine learning. It uses BeautifulSoup to remove HTML tags, regular expressions to remove URLs, email addresses, numbers, and punctuation, and then applies TF-IDF vectorization to create a numerical representation of the text suitable for model training. The `preprocess` function orchestrates these steps, returning the vectorized text data, target variables, and the trained TF-IDF vectorizer. **train.py** This Python script preprocesses text data, trains a logistic regression model for spam classification using TF-IDF vectorization, evaluates the model's accuracy and classification performance, and saves both the trained model and vectorizer to disk using joblib. === DETAILED ANALYSIS === ## Project Analysis: Email Spam Classifier This document provides a comprehensive analysis of the provided Python codebase, which implements an email spam classifier. **1. Project Purpose and Main Objectives:** The primary objective of this project is to build a machine learning model capable of accurately classifying email messages as either spam or not spam. The project achieves this by leveraging natural language processing (NLP) techniques to preprocess email text, followed by training a logistic regression model on a merged dataset of email samples. The final goal is to create a functional application that can classify user-provided email text in real-time. **2. Key Features and Capabilities:** * **Data Integration:** The system merges data from four different CSV datasets (`combined_data.csv`, `emails.csv`, `spam_assassin.csv`, `spam.csv`), handling variations in column names and data formats to create a unified training dataset. * **Text Preprocessing:** Sophisticated text preprocessing is performed, including: * HTML removal using BeautifulSoup. * Lowercasing. * Removal of URLs, email addresses, numbers, and punctuation. * Removal of extra whitespace. * **TF-IDF Vectorization:** The preprocessed text is converted into numerical feature vectors using TF-IDF (Term Frequency-Inverse Document Frequency), a technique that captures the importance of words in a document relative to the entire corpus. The `max_features` parameter limits the vocabulary size to 3000. * **Model Training:** A Logistic Regression model is trained on the vectorized data. The `max_iter` parameter is set to 1000 to ensure convergence. The model is trained using a stratified train-test split (80% train, 20% test) with `random_state=42` for reproducibility. * **Model Persistence:** The trained Logistic Regression model and the TF-IDF vectorizer are saved using `joblib` for later reuse, eliminating the need for retraining. * **Real-time Classification:** A command-line interface allows users to input email text and receive a real-time spam/not-spam classification. **3. Target Audience and Use Cases:** The target audience includes individuals and organizations interested in automatically filtering spam emails. Use cases include: * **Personal Email Filtering:** Individuals can use the application to pre-filter their incoming emails, reducing the number of spam messages they see. * **Email Security Systems:** The model could be integrated into larger email security systems to improve spam detection accuracy. * **Educational Purposes:** The codebase serves as a valuable learning resource for individuals interested in NLP and machine learning. **4. Technical Approach and Methodology:** The project employs a supervised machine learning approach. The methodology involves: 1. **Data Collection and Preparation:** Gathering data from multiple sources and merging them into a consistent format. 2. **Data Preprocessing:** Cleaning and transforming the text data to improve model performance. 3. **Feature Engineering:** Converting text data into numerical features using TF-IDF. 4. **Model Selection and Training:** Choosing a suitable model (Logistic Regression) and training it on the prepared data. 5. **Model Evaluation:** Assessing the performance of the trained model using metrics like accuracy and classification report. 6. **Deployment:** Creating a simple command-line interface for real-time prediction. **5. Notable Implementation Details:** * **Modular Design:** The code is organized into well-defined functions (`removeHtmlAndLower`, `cleanText`, `vectorizeTexts`, `preprocess`), promoting code reusability and maintainability. * **Libraries Used:** The project relies on several key libraries: * `pandas` for

data manipulation. * `BeautifulSoup` for HTML parsing. * `re` for regular expression-based text cleaning. * `sklearn` for machine learning (TF-IDF, Logistic Regression, model evaluation). * `joblib` for model persistence. * **Reproducibility:** The use of `random_state=42` in the train-test split ensures that the results are reproducible. * **Error Handling:** While basic error handling (e.g., the `exit` command) is implemented, more robust error handling could be added to handle potential exceptions during file reading or model prediction. * **Scalability:** The current implementation is not optimized for extremely large datasets. For significantly larger datasets, techniques like data streaming or more advanced model architectures might be considered. **Overall:** The project presents a well-structured and functional email spam classifier. The use of established NLP and machine learning techniques, along with clear code organization, makes it a valuable example of a practical machine learning application. Further improvements could focus on enhancing error handling, scalability, and potentially exploring more advanced models for improved accuracy.

Metric	Value
Total Files	7
Total Lines of Code	1101
Technologies Used	1

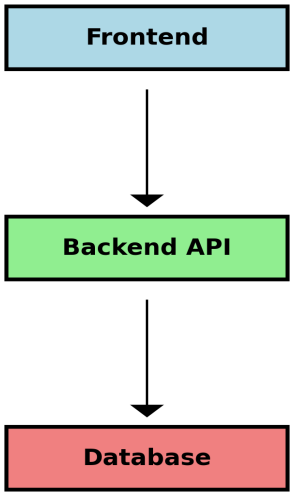
Technology Stack

Programming Languages

Python

System Architecture

System Architecture



File Structure Analysis

File Type	Count
.py	4
.pkl	2
.md	1

Knowledge Assessment

Q1: What is the primary purpose of the `merged_Dataset()` function?

A. To clean and preprocess the text data.
B. To train the spam classification model.
C. To merge multiple datasets into a single DataFrame for training.

Answer: To merge multiple datasets into a single DataFrame for training.

Q2: Which library is used for performing TF-IDF vectorization of the text data?

A. scikit-learn
B. NLTK
C. spaCy

Answer: scikit-learn

Q3: The `cleanText()` function removes several elements from the text. Which of the following is NOT removed?

A. URLs
B. Email addresses
C. Numbers

Answer: Stop words

Q4: What machine learning model is used for spam classification in this project?

A. Support Vector Machine (SVM)
B. Random Forest
C. Logistic Regression

Answer: Logistic Regression

Q5: What is the role of `joblib.dump()` and `joblib.load()`?

A. To preprocess the text data.
B. To train the machine learning model.
C. To serialize and deserialize the model and vectorizer.

Answer: To serialize and deserialize the model and vectorizer.