SMS Spam Detection Project Documentation

Introduction

The SMS Spam Detection project is a machine learning-based solution designed to classify text messages as either "ham" (legitimate messages) or "spam" (unwanted or malicious messages). This project leverages natural language processing (NLP) techniques and machine learning algorithms to analyse and categorize SMS messages, helping users filter out unwanted content.

Objectives

1. **Data Cleaning**: Preprocess the raw SMS data to remove noise, handle missing values, and standardize the format.
2. **Exploratory Data Analysis (EDA)**: Analyse the dataset to understand its structure, distribution, and key characteristics.
3. **Text Preprocessing**: Convert raw text into a format suitable for machine learning models by tokenizing, removing stop words, and applying other NLP techniques.
4. **Model Building**: Train and evaluate machine learning models to classify messages as spam or ham.
5. **Evaluation**: Assess the performance of the models using appropriate metrics.
6. **Improvement**: Optimize the model for better accuracy and efficiency.
7. **Deployment**: Create a user-friendly interface or API to deploy the model for real-world use.

Dataset

The dataset used in this project contains SMS messages labeled as either "ham" or "spam". The original dataset has 5,572 entries with 5 columns, but after cleaning, it is reduced to 5,169 entries with 2 main columns:

- **Target**: Indicates whether the message is ham (0) or spam (1).
- **Text**: The content of the SMS message.

Key Statistics

- **Ham messages**: 4,516 (87.37%)
- **Spam messages**: 653 (12.63%)

Data Cleaning

1. **Removing Unnecessary Columns**: The dataset initially contained three unnamed columns with mostly missing values, which were dropped.
2. **Renaming Columns**: The columns were renamed to "Target" and "Text" for clarity.

3. **Handling Missing Values**: No missing values were found in the cleaned dataset.
4. **Removing Duplicates**: 403 duplicate entries were identified and removed to ensure data quality.
5. **Encoding Labels**: The "Target" column was encoded using Label Encoder, converting "ham" to 0 and "spam" to 1.

Exploratory Data Analysis (EDA)

1. **Class Distribution**: The dataset is imbalanced, with ham messages significantly outnumbering spam messages.
    - Visualization: A pie chart shows the distribution (87.37% ham vs. 12.63% spam).
2. **Message Length Analysis**:
    - **Number of Characters**: The average message length is 78.98 characters.
    - **Number of Words**: The average message contains 18.46 words.
    - **Number of Sentences**: The average message has 1.97 sentences.
3. **Comparison Between Ham and Spam**:
    - Spam messages tend to be longer (average 137.89 characters) compared to ham messages (average 70.46 characters).
    - Spam messages also contain more words (average 27.67) and sentences (average 2.97) than ham messages.

Text Preprocessing

1. **Tokenization**: Messages are split into individual words or sentences using NLTK's word_tokenize and sent_tokenize functions.
2. **Feature Engineering**: Additional features like the number of characters, words, and sentences were extracted for analysis.

Model Building (Planned)

While the notebook provided does not include the actual model building, the following steps are typically involved:

1. **Text Vectorization**: Convert text data into numerical features using techniques like TF-IDF or CountVectorizer.
2. **Model Selection**: Choose appropriate algorithms such as Naive Bayes, Logistic Regression, or Random Forest.
3. **Training**: Fit the model on the training data.

4. **Evaluation**: Use metrics like accuracy, precision, recall, and F1-score to assess performance.

Evaluation (Planned)

- **Metrics**: Accuracy alone may not be sufficient due to class imbalance. Precision, recall, and F1-score are crucial for evaluating spam detection.
- **Cross-Validation**: Ensure the model generalizes well to unseen data.

Improvement (Planned)

- **Handling Imbalance**: Techniques like oversampling, undersampling, or using class weights can address the imbalance.
- **Hyperparameter Tuning**: Optimize model parameters for better performance.
- **Advanced Techniques**: Experiment with deep learning models like LSTM or transformers for improved accuracy.

Deployment (Planned)

- **Web Application**: Develop a simple interface where users can input messages and receive spam/ham predictions.
- **API**: Create an API for integration into other applications or services.

Challenges

1. **Class Imbalance**: The dataset is heavily skewed toward ham messages, which may bias the model.
2. **Text Variability**: Spam messages can vary widely in content and structure, making them harder to detect.
3. **Feature Extraction**: Converting text into meaningful numerical features requires careful preprocessing.

Conclusion

The SMS Spam Detection project demonstrates the process of cleaning, analyzing, and preprocessing text data for machine learning. While the notebook covers initial steps like data cleaning and EDA, the next phases (model building, evaluation, and deployment) are critical for creating a functional spam detection system. By leveraging NLP techniques and machine learning, this project can effectively filter spam messages, enhancing user experience and security. Future work includes implementing and optimizing models, addressing class imbalance, and deploying the solution for real-world use.

Future Work

1. **Experiment with Models**: Test various algorithms to find the best-performing one.

2. **Real-Time Processing**: Extend the project to handle real-time message classification.
3. **User Feedback**: Incorporate feedback mechanisms to improve model accuracy over time.
4. **Multilingual Support**: Expand the system to detect spam in multiple languages.

This project serves as a foundation for building robust spam detection systems, with potential applications in email filtering, social media moderation, and more.