Spam Email Detection README

1 Introduction

This project implements a spam email classifier using a Naive Bayes model in a Jupyter Notebook. The classifier uses the Scikit-learn library to process text data and classify messages as either "ham" (non-spam) or "spam" based on a publicly available SMS dataset.

2 Features

- Loads and preprocesses a dataset of SMS messages labeled as "ham" or "spam".
- Uses CountVectorizer for text feature extraction.
- Trains a Multinomial Naive Bayes model for classification.
- Evaluates the model using accuracy, classification report, and a confusion matrix visualized with a heatmap.
- Achieves high accuracy in distinguishing spam from ham messages.

3 Requirements

- Python 3.x
- Jupyter Notebook (`pip install notebook`)
- Required Python libraries:

```
pip install pandas numpy scikit-learn seaborn matplotlib
```

4 Installation

1. Clone or download the repository containing the `Spam_Email_Detection.ipynb` notebook.
2. Install the required dependencies:

```
pip install pandas numpy scikit-learn seaborn matplotlib
```

3. Ensure you have Jupyter Notebook installed:

```
pip install notebook
```

5 Usage

1. Launch Jupyter Notebook:

   ```
   jupyter notebook
   ```

2. Open the `Spam_Email_Detection.ipynb` file in the Jupyter interface.
3. Run all cells in the notebook to:
   - Load the SMS dataset from a public URL.
   - Preprocess the data (convert labels to binary, split into training/test sets).
   - Vectorize text using CountVectorizer.
   - Train a Multinomial Naive Bayes model.
   - Evaluate the model and display accuracy, classification report, and a confusion matrix heatmap.

6 Dataset

The notebook uses the SMS Spam Collection dataset, which is publicly available at:

`https://raw.githubusercontent.com/justmarkham/pycon-2016-tutorial/master/data/sms.tsv`

- The dataset contains two columns: `label` ("ham" or "spam") and `message` (the text content).
- The dataset is automatically downloaded when the notebook is run.

7 How It Works

- **Data Loading**: Reads the SMS dataset into a Pandas DataFrame.
- **Preprocessing**: Maps "ham" to 0 and "spam" to 1 for binary classification.
- **Data Splitting**: Splits data into 80% training and 20% test sets.
- **Text Vectorization**: Converts text messages into numerical features using CountVectorizer.
- **Model Training**: Trains a Multinomial Naive Bayes classifier on the training data.
- **Evaluation**: Computes accuracy, generates a classification report, and visualizes a confusion matrix using Seaborn.

8 Example Output

Upon running the notebook, you will see:

- **Accuracy**: A value around 0.99 (e.g., 99.19%), indicating high model performance.
- **Classification Report**: Metrics like precision, recall, and F1-score for both "ham" and "spam" classes.
- **Confusion Matrix**: A heatmap showing true positives, true negatives, false positives, and false negatives.

9 Limitations

- The model relies on a simple Naive Bayes algorithm, which may not capture complex patterns in text.
- The dataset is relatively small and specific to SMS messages, so performance may vary on other types of email data.
- No advanced text preprocessing (e.g., stopword removal, lemmatization) is applied, which could improve results.

10 Future Improvements

- Incorporate advanced text preprocessing (e.g., NLTK for tokenization, lemmatization).
- Experiment with other models like SVM, Random Forest, or deep learning approaches (e.g., LSTM).
- Use a larger or more diverse dataset for improved generalization.
- Add cross-validation for more robust model evaluation.