MEKELLE UNVIERSITY

EITM
SCHOOL OF COMPUTING

DEPARTMENT OF SOFTWARE ENGINEERING

Fundamentals of Machine Learning
Individual Assignment

- Name: Semere Herruy
- ID: ugr/177912/12

Submission date: 19/02/2017 E.c
Submitted to : Dr. Guesh

Spam Detection Project Report

Submitted by: [semere]

1. Project Description

The objective of this project is to build a spam detection classifier that can automatically classify messages as either "Spam" or "Not Spam." Spam detection is essential in filtering unwanted and potentially harmful messages, improving email security, and enhancing user experience. This project utilizes the Naive Bayes algorithm, which is a popular machine learning technique for text classification tasks, due to its simplicity, efficiency, and proven success with categorical text data. We trained and evaluated the model on a labeled dataset, aiming to achieve high accuracy and reliable performance.

2. Dataset Description

The dataset used in this project consists of labeled text messages with two categories:

- **Spam:** Messages with irrelevant or unwanted content, often promotional.
- **Not Spam:** Legitimate messages with useful information.

Each row in the dataset includes:

- **Text:** The content of the message.
- **Label:** The classification label, either "Spam" or "Not Spam".

The dataset contains **5,572** messages, with a balanced distribution of spam and not spam messages.

3. Methodology and Model Choice

3.1 Text Classification Approach

For text classification, we use the TF-IDF (Term Frequency-Inverse Document Frequency) representation to convert text into a numerical format that can be processed by machine learning models. Specifically, the `TfidfVectorizer` technique from scikit-learn is employed to create a feature matrix based on weighted word frequencies in each message.

3.2 Model: Naive Bayes

We chose **Multinomial Naive Bayes (MNB)** as the classifier because:

- **Simplicity and Efficiency:** MNB is computationally efficient, making it suitable for high-dimensional data such as text.
- **Performance:** It performs well with sparse data, which is typical in text classification, as each text sample contains only a fraction of possible words. Naive Bayes assumes independence between features (words in this case), which, while not strictly true, works surprisingly well in practice for many text classification problems.

4. Implementation

4.1 Importing Libraries

The following libraries are utilized in this project:

- **pandas:** For data handling.
- **scikit-learn:** For splitting data, transforming text, training the model, and evaluating results.

4.2 Data Loading and Preprocessing

The dataset is loaded from a CSV file (`spam_not_spam.csv`). The columns are accessed directly as 'Message' and 'Category' based on the dataset format.

4.3 Data Splitting

The dataset is split into a training set (80%) and a test set (20%). The training set is used to train the model, and the test set evaluates performance on unseen data.

4.4 Text Vectorization

The `TfidfVectorizer` from `sklearn.feature_extraction.text` converts text into a numerical matrix based on weighted word frequencies. This process, called TF-IDF, is essential for transforming the data into a format usable by the Naive Bayes classifier.

4.5 Model Training

We initialize the `MultinomialNB` classifier and train it on the vectorized text data.

4.6 Model Evaluation

We evaluate the model's accuracy, precision, recall, and F1 score using the `accuracy_score`, `classification_report`, and `confusion_matrix` functions. These metrics help gauge how well the classifier differentiates between spam and not spam messages.

5. Code Implementation

Here is the code used in this project:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
df = pd.read_csv('spam_not_spam.csv')

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['Message'],
df['Category'], test_size=0.2, random_state=42)

# Vectorize the text data
vectorizer = TfidfVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

# Train the Naive Bayes model
model = MultinomialNB()
model.fit(X_train_vectorized, y_train)

# Make predictions on test data
y_pred = model.predict(X_test_vectorized)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

6. Experimental Results

6.1 Initial Results

After the first training run, the model achieved an accuracy of approximately **99%** on the test set. The classification report provides detailed insights into the model's performance for both classes:

```
Accuracy: 0.991
Classification Report:
              precision    recall  f1-score   support

         ham       0.99      1.00      0.99       966
        spam       1.00      0.93      0.97       149

    accuracy                           0.99      1115
   macro avg       0.99      0.97      0.98      1115
weighted avg       0.99      0.99      0.99      1115
```

Key Metrics:

- **Precision:** The ratio of correctly predicted positive observations to the total predicted positives. For spam, it is **1.00**, and for not spam, it is **0.99**.
- **Recall:** The ratio of correctly predicted positive observations to all actual positives. For spam, it is **0.93**, and for not spam, it is **1.00**.
- **F1 Score:** The weighted average of precision and recall. For spam, it is **0.97**, and for not spam, it is **0.99**.

6.2 Hyperparameter Tuning

Further adjustments in parameters, like alpha in MultinomialNB, may improve performance. However, the initial results are satisfactory for this task.

6.3 Conclusion from Results

The model performs exceptionally well in distinguishing spam from not spam, indicating high accuracy and reliability. Its simplicity makes it suitable for practical applications without heavy computational resources. Improvements could be achieved by using additional features or trying advanced models like Support Vector Machines or ensemble methods.

7. Conclusion and Future Work

This project demonstrates that Naive Bayes is effective for spam detection with a straightforward implementation and high accuracy. Future work could include:

1. **Data Augmentation:** Using more varied datasets or pre-trained embeddings to increase data richness.
2. **Alternative Models:** Testing models like Support Vector Machines or neural networks to further enhance performance.
3. **Model Deployment:** Integrating the trained model into a web or mobile application for real-time spam filtering.

8. References

- Scikit-learn Documentation: https://scikit-learn.org/stable/
- Text Classification with Naive Bayes: A Practical Introduction