SMS Spam Classifier using Naive Bayes

Rabiul Hassan
Hangzhou Dianzi University

May 2025

Contents

1 Introduction

With the rapid growth of digital communication, especially via mobile messaging, identifying and filtering out spam messages has become a crucial task. This project applies the Naive Bayes algorithm to detect spam in SMS messages with high accuracy and efficiency. It culminates in a lightweight, interactive web-based tool that allows end users to test their own messages for spam detection. The full project, including code and documentation, is available on my GitHub: github.com/rabiulhassandev/ML-SMS-Spam-Detector.git.

2 Mathematical Foundations

2.1 Bayes' Theorem

Naive Bayes is built on Bayes' Theorem, which describes the probability of a class C given some data X:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

In this context:

- $P(C|X)$ is the posterior probability of class C (spam or ham) given the text X
- $P(X|C)$ is the likelihood of observing text X given class C
- $P(C)$ is the prior probability of class C
- $P(X)$ is the prior probability of text X

2.2 Multinomial Naive Bayes

This variant of Naive Bayes is especially suited for text classification where input features represent frequencies of words. For each message, the model calculates:

$$P(C) \prod_{i=1}^{n} P(x_i|C)$$

Where x_i represents word occurrences in a message. Laplace smoothing is used to avoid zero probabilities.

2.3 Strengths and Weaknesses

- **Advantages:** Fast training, scalable, simple to implement, works well with text data.
- **Limitations:** Assumes feature independence, which may not hold true in natural language.

3 Real-World Applications

3.1 Common Use Cases

- Spam detection in SMS and email systems
- News categorization
- Sentiment analysis of customer reviews
- Topic classification in blogs or forums

3.2 Dataset Description

This project uses the publicly available **SMS Spam Collection Dataset** from UCI Machine Learning Repository. It includes:

- 5,572 total SMS messages
- 4,825 labeled as **ham** (non-spam)
- 747 labeled as **spam**

3.3 Label Distribution

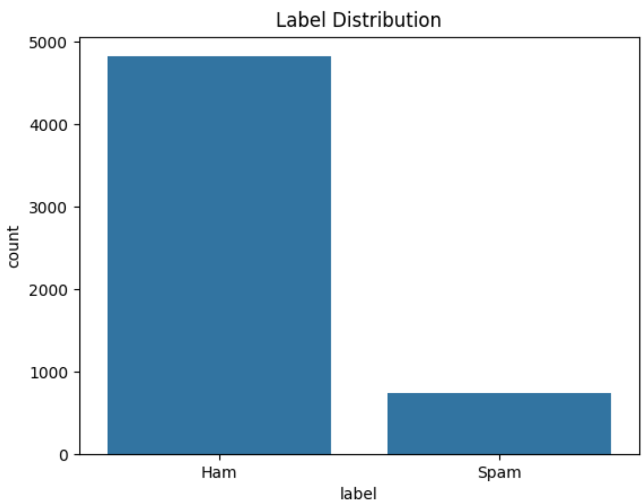

Figure 1: Label distribution showing class imbalance

4 Implementation

4.1 Tools and Libraries

- `Python` and `Flask` for backend and web app
- `scikit-learn` for machine learning
- `Pandas`, `Matplotlib`, `Seaborn` for data processing and visualization

4.2 Pipeline Overview

1. Load and clean the dataset
2. Convert labels to binary form (0 for ham, 1 for spam)
3. Split the data into training and test sets
4. Vectorize messages using `CountVectorizer`
5. Train the Naive Bayes classifier
6. Evaluate with classification metrics
7. Build a Flask-based web UI for real-time testing

4.3 Sample Code

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(messages)
model = MultinomialNB()
model.fit(X_train, y_train)
```

5 Model Evaluation

5.1 Performance Report

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       966
           1       1.00      0.74      0.85       149

    accuracy                           0.97      1115
   macro avg       0.98      0.87      0.92      1115
weighted avg       0.97      0.97      0.96      1115
```

5.2 Accuracy

$$\text{Accuracy} = 96.6\%$$

This demonstrates excellent performance, particularly in identifying ham messages. The slightly lower recall for spam is expected due to class imbalance.

6 Web Interface

The deployed Flask web application allows users to:

- Explore summary statistics and charts
- View the classification report
- Enter their own SMS message and receive instant spam/ham prediction

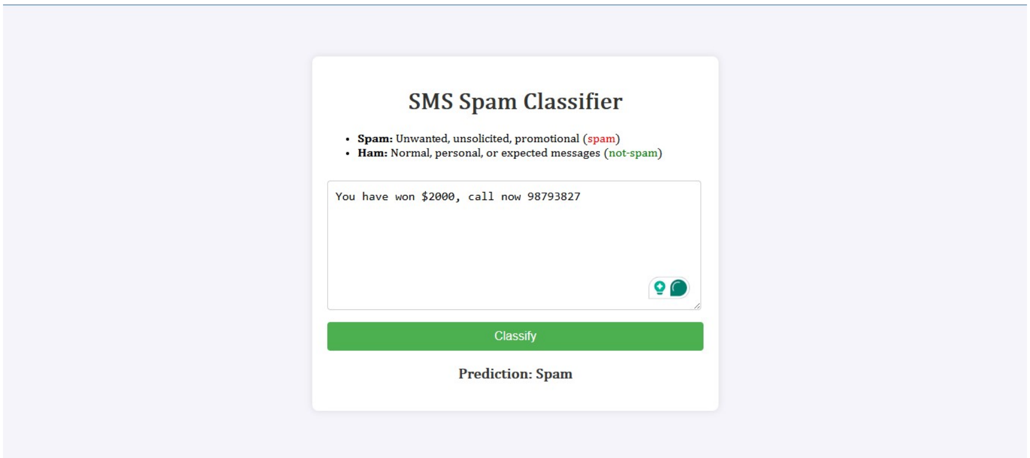

Figure 2: Web Interface Screenshot

7 Conclusion

This project successfully implements and deploys a machine learning solution for spam classification using Naive Bayes. It showcases how a classic algorithm, combined with modern tools like Flask and scikit-learn, can deliver a powerful real-world application. The classifier achieved over 96% accuracy and is capable of real-time spam detection through a browser.

For complete source code and documentation, visit my GitHub repository:
`https://github.com/rabiulhassandev/ML-SMS-Spam-Detector.git`