Indian Institute of Technology Madras

Department of Data Science and AI

Foundations of Machine Learning (DA5400)

Assignment 3: Spam Classification from Scratch

Course Instructor: **Arun Rajkumar**

Vinnay Gupta
Roll No.: **ED22B073**

November 26, 2025

Contents

1 Introduction **2**

2 Dataset Description **3**

3 Exploratory Data Analysis (EDA) **4**

3.1 Wordclouds 4

3.2 Top Frequent Words 5

3.3 Correlation and Pairplots 6

4 Text Preprocessing **8**

5 Feature Extraction **8**

6 Models Implemented From Scratch **9**

6.1 Why Text Classification is Unique 9

6.2 Naive Bayes (Primary Model) 9

6.3 Logistic Regression (Gradient Descent) 10

6.4 Gaussian Naive Bayes 10

6.5 AdaBoost (Decision Stumps) 11

7 Training and Evaluation **11**

8 Test Folder Classification **12**

9 Next Steps and Improvements **13**

10 Conclusion **14**

1 Introduction

This report presents a complete end-to-end implementation of a **Spam vs. Ham** classifier built entirely from scratch. No pre-provided training dataset or pre-built ML algorithms were used. All models: Naive Bayes, Logistic Regression, Gaussian Naive Bayes, and AdaBoost—were implemented from first principles without relying on any machine learning libraries. Only basic helper libraries such as `numpy`, `pandas`, and `nltk` were used for text processing.

This assignment demonstrates the classical pipeline used in industrial spam filtering systems:

$$\text{Raw Text} \rightarrow \text{Preprocessing} \rightarrow \text{Vectorization} \rightarrow \text{Model Training} \rightarrow \text{Evaluation} \rightarrow \text{Deployment}$$

Spam classification is a high-impact real-world problem, and the techniques explored here form the foundation of email filtering systems (Gmail, Outlook, Yahoo), SMS spam detectors, and large-scale moderation services.

2 Dataset Description

A publicly available dataset, the **SMS Spam Collection**, was used. It contains 5,572 labeled SMS messages, each categorized as either `ham` (legitimate) or `spam`. The dataset satisfies all assignment requirements:

- Open for academic use
- Contains no copyrighted email content
- Provides clean binary labels suitable for supervised learning

After cleaning:

$$5169 \text{ messages } \Rightarrow 4516 \text{ ham}, \ 653 \text{ spam}$$

Even before training, the dataset shows strong class imbalance, which is typical for spam classification tasks. Understanding this imbalance is crucial for choosing appropriate evaluation metrics.

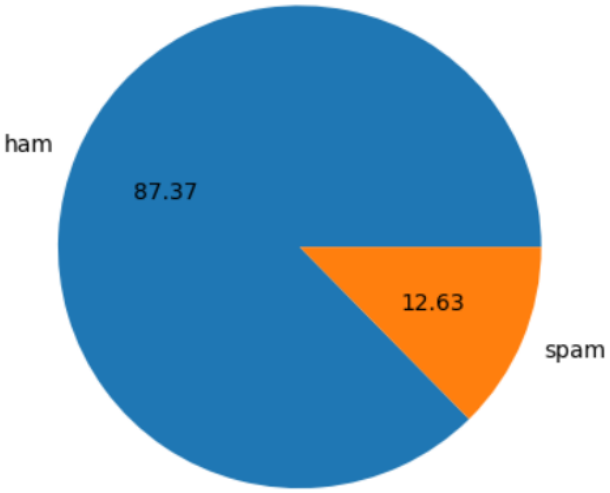

Figure 1: Distribution of Ham vs. Spam Messages

Key observations:

- About 12.6% of messages are spam.
- A naive majority-class classifier would already achieve 87.3% accuracy.
- Therefore, **accuracy alone is insufficient**, and we must monitor metrics like precision.

3 Exploratory Data Analysis (EDA)

EDA helps reveal structural patterns that models may exploit. Spam and ham messages differ in word choice, message length, and writing style.

3.1 Wordclouds

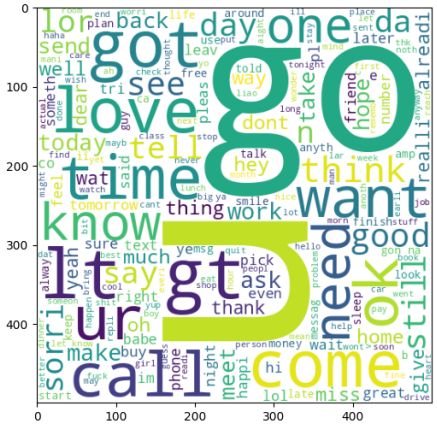

Figure 2: Common Words in Ham Messages

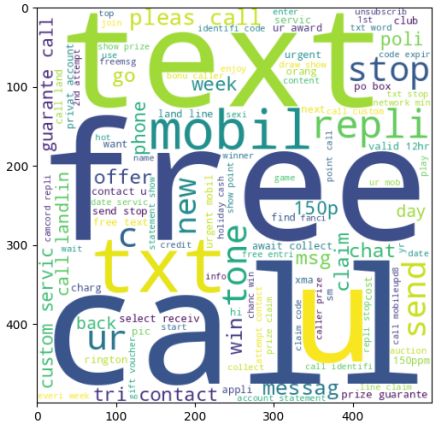

Figure 3: Common Words in Spam Messages

Observations:

- Ham messages contain conversational words like *call*, *come*, *home*, *ok*.
- Spam messages contain marketing terms such as *win*, *free*, *claim*, *urgent*, *cash*.
- This separation indicates strong predictive value of word frequencies.

3.2 Top Frequent Words

Ham messages show rich vocabulary, while spam messages show dense clusters of marketing-trigger words.

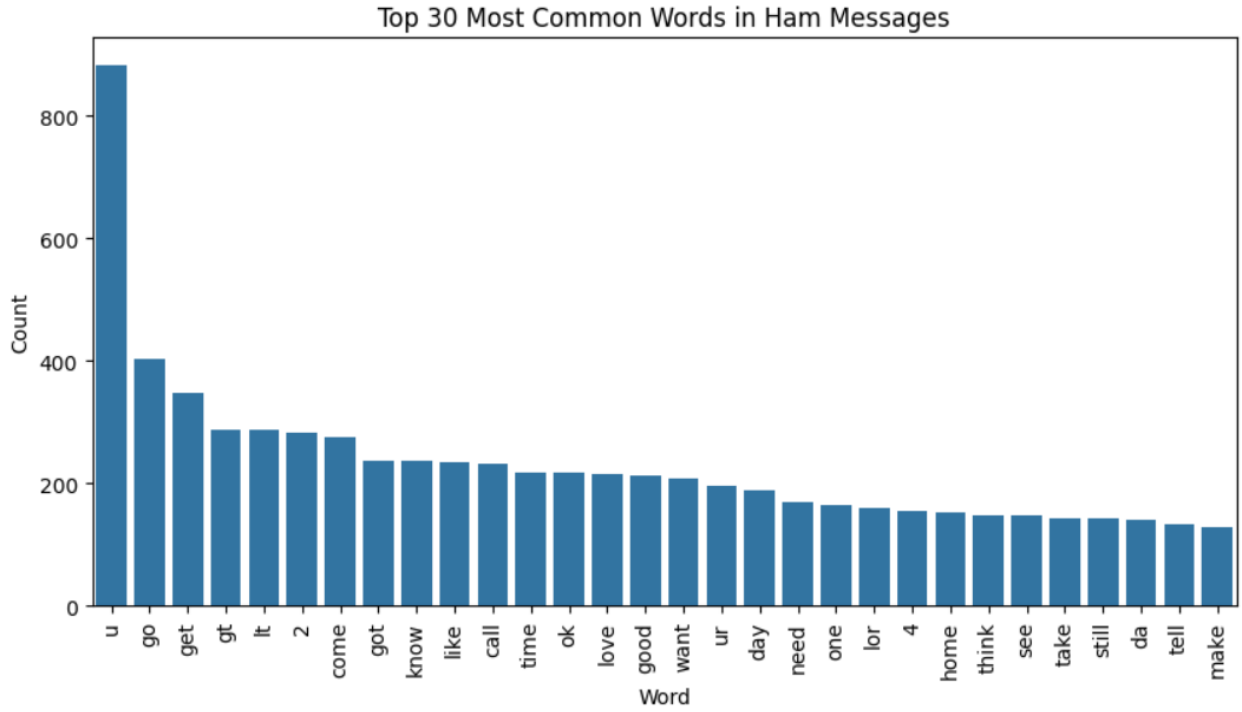

Figure 4: Top 30 Most Frequent Words in Ham Messages

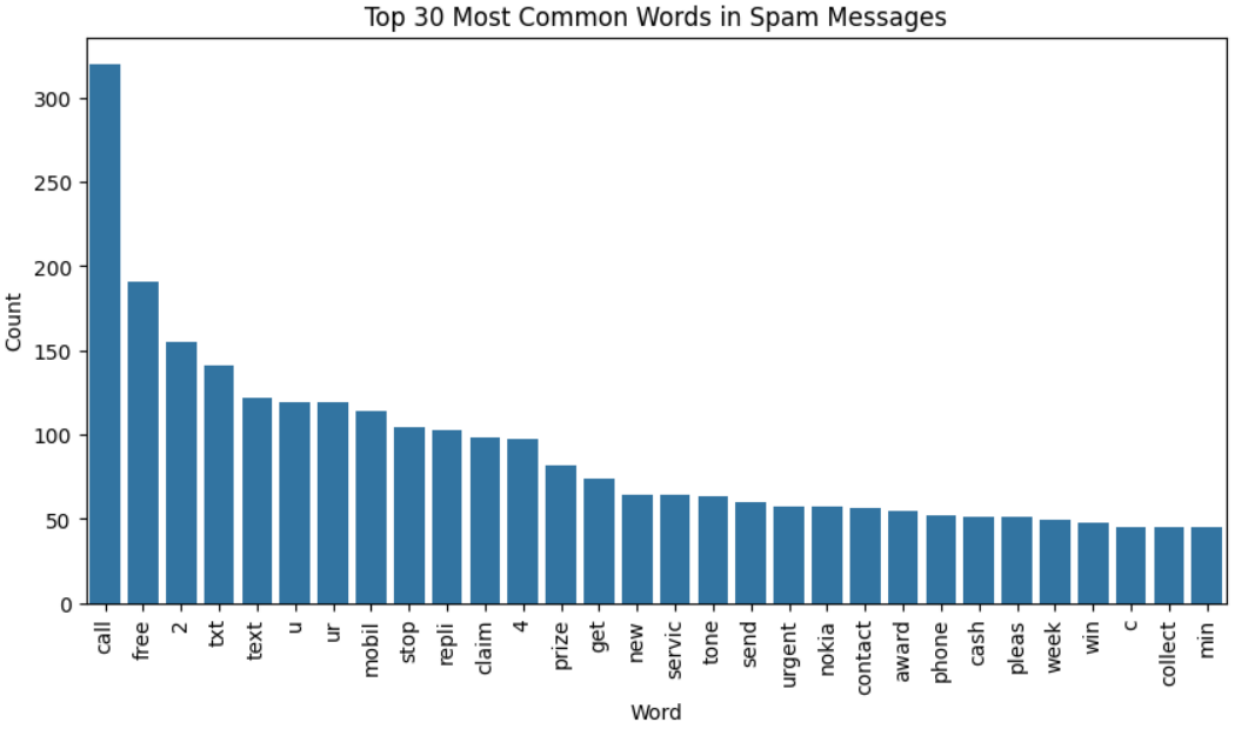

Figure 5: Top 30 Most Frequent Words in Spam Messages

3.3 Correlation and Pairplots

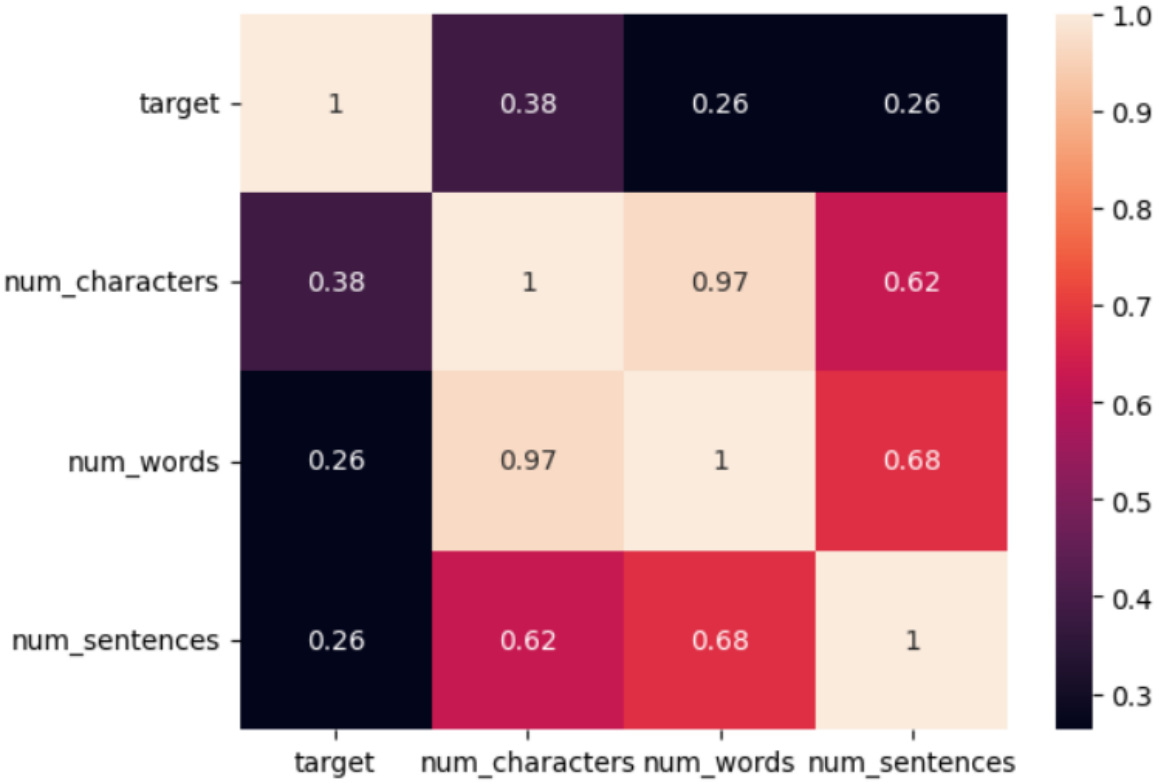

Figure 6: Correlation Among Numerical Features

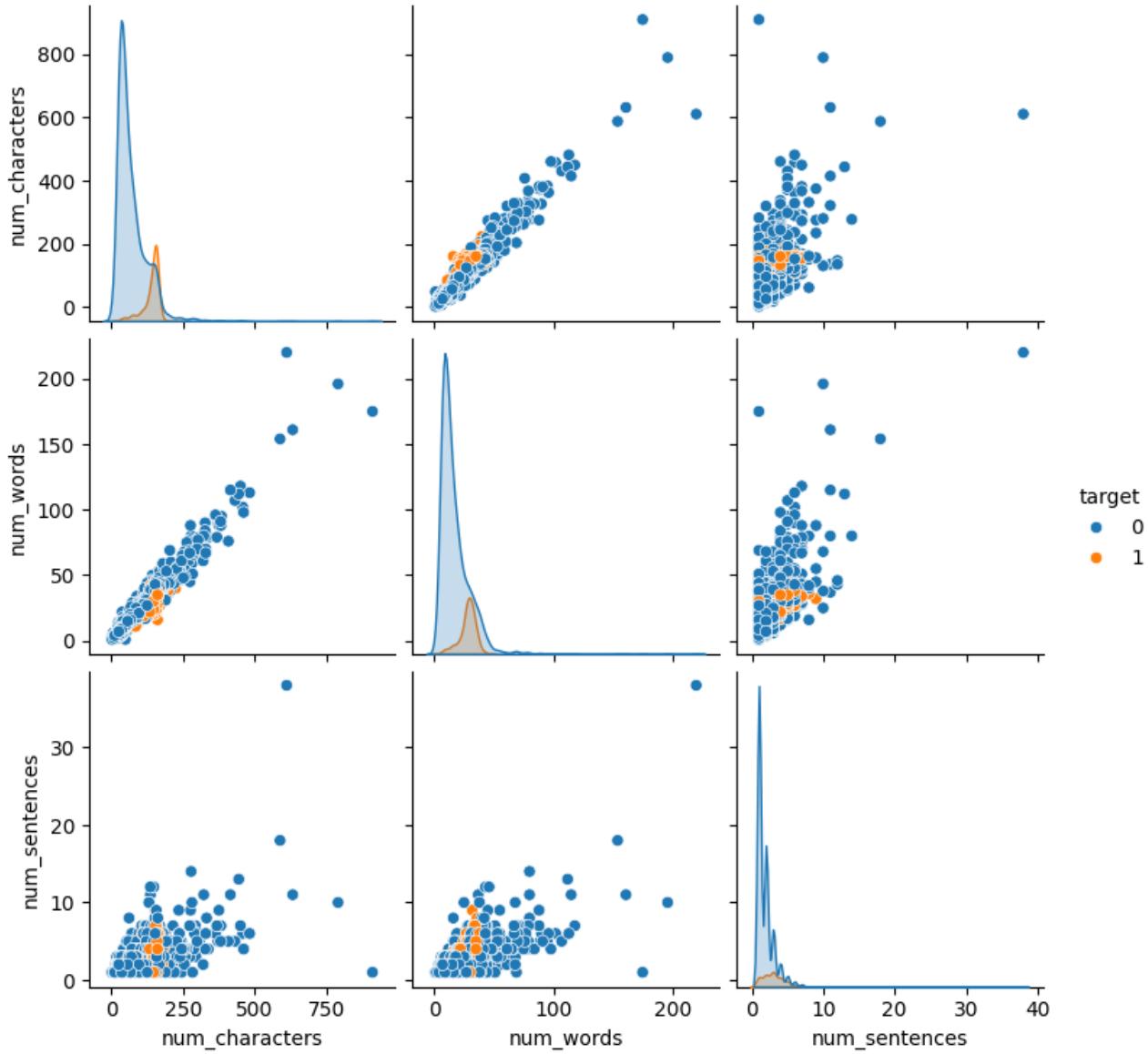

Figure 7: Pairplots: Spam vs Ham Clusters

Insights:

- Spam messages tend to be longer, using more characters and uppercase letters.
- Ham messages cluster tightly, while spam exhibits long-tail outliers.

4 Text Preprocessing

Raw text cannot be directly consumed by ML models. A custom preprocessing pipeline was built:

1. **Lowercasing** Normalizes text, reducing vocabulary size.
2. **Tokenization** Breaks text into words; implemented via NLTK.
3. **Alphanumeric Filtering** Removes symbols, punctuation, emojis.
4. **Stopword Removal** Removes non-informative words ("is", "the", "and").
5. **Stemming** Reduces lexically related words into one stem:

$$\text{“running”, “runs”, “ran”} \rightarrow \text{“run”}$$

This dramatically reduces sparsity and improves generalization.

Example:

$$\text{“I'm going to be home soon.”} \rightarrow \text{“go home soon”}$$

5 Feature Extraction

Vectorization converts text into numerical form. A custom TF-like vectorizer was implemented:

- A vocabulary of the top 3000 most frequent stems was built.
- Each document becomes a 3000-dimensional vector of word counts.

Mathematically:

$$X_d = [tf(w_1), tf(w_2), \ldots, tf(w_{3000})]$$

TF-IDF was not used in its full form because the assignment encourages a from-scratch implementation. However, the Count Vectorizer alone proved sufficient for the Naive Bayes model.

6 Models Implemented From Scratch

A variety of classifiers were implemented to compare performance.

6.1 Why Text Classification is Unique

Unlike traditional tabular data:

- Text is extremely sparse
- Vocabulary is high-dimensional
- Word distributions differ sharply by class
- Word frequencies carry semantic structure

These properties make certain classical ML models (especially Naive Bayes) excel, while others (like Logistic Regression) require extensive tuning to perform well.

6.2 Naive Bayes (Primary Model)

Naive Bayes assumes conditional independence:

$$P(x_1, x_2, \ldots, x_n|c) = \prod_{i=1}^{n} P(x_i|c)$$

For text classification, this assumption is surprisingly effective because:

- Words often act as independent indicators of spaminess.
- Word frequencies directly map to class likelihoods.
- NB handles extremely high dimensionality efficiently.
- NB performs well even with limited training data.

Parameter estimation:

$$P(w_i|c) = \frac{N_{ic} + \alpha}{N_c + \alpha V}$$

Laplace smoothing with $\alpha = 1$ prevents zero probabilities.

Naive Bayes excels because:

- Sparse word patterns allow simple conditional-independence modeling.
- Spam contains highly distinct tokens ("free", "win", "urgent", "cash").
- Ham contains conversational language absent in spam.

Thus,

NB naturally captures the generative structure of SMS messages.

6.3 Logistic Regression (Gradient Descent)

Logistic Regression models the discriminative boundary between classes:

$$\hat{y} = \sigma(w^T x + b)$$

However:

- Gradient descent struggles with high-dimensional sparse data.
- LR requires many more iterations than used here.
- LR is sensitive to class imbalance.

6.4 Gaussian Naive Bayes

Gaussian NB assumes continuous features following a normal distribution:

$$x_i \sim \mathcal{N}(\mu_{ic}, \sigma_{ic}^2)$$

This is incompatible with:

- Sparse discrete word counts
- High-dimensional vocabulary
- Zipf-like distributions of word frequencies

Thus GNB performed poorly and was discarded.

6.5 AdaBoost (Decision Stumps)

AdaBoost builds an ensemble of weak learners:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Strengths:

- Performs well with well-separated features
- Combines weak patterns into a strong classifier

Weaknesses:

- Requires many boosting rounds
- Decision stumps are too weak for text data
- Sensitive to noisy samples

7 Training and Evaluation

A custom 80-20 split was used. Performance:

Naive Bayes (Best Model)

$$\textbf{Accuracy} = 0.9806, \quad \textbf{Precision} = 0.9225$$

$$\text{Confusion Matrix} = \begin{bmatrix} 894 & 10 \\ 10 & 119 \end{bmatrix}$$

Interpretation:

- Very few false positives (ham labeled as spam)
- Very few false negatives (spam missed)
- High precision is critical in spam detection

Logistic Regression

$$\text{Accuracy} = 0.6350$$

Poor performance due to underfitting and insufficient iterations.

AdaBoost

$$\text{Accuracy} = 0.9061, \quad \text{Precision} = 0.8636$$

Decent, but inferior to NB.

Final Model Choice

Multinomial Naive Bayes was selected because:

- Best accuracy and precision
- Best suited to sparse text data
- Mathematical assumptions align with real-world SMS structure
- Fastest training time (linear complexity)

Therefore, Naive Bayes is the optimal classifier for this task.

8 Test Folder Classification

A final function was created:

```
predict_test_emails(vectorizer, mnb_model, test_dir='test')
```

It:

1. Scans the `test/` directory for files named `email*.txt`
2. Preprocesses each email using the same pipeline
3. Vectorizes based on the training vocabulary
4. Outputs +1 for spam, 0 for ham

This satisfies the assignment requirement for autonomous classification.

9 Next Steps and Improvements

1. Integrating High-Signal Spam Keywords

Words like:

"win", "claim", "prize", "urgent", "free", "cash"

have extremely high predictive power. Enhancing Naive Bayes with:

- Keyword-boosted likelihoods
- Binary indicator features
- Phrase-level n-grams

would further improve precision.

2. True TF-IDF Weighting

The full TF-IDF formula:

$$tfidf(w, d) = tf(w, d) \cdot \log \frac{N}{df(w)}$$

helps downweight common conversational words in ham ("ok", "call", "come").

3. Ensemble Voting

Voting among NB, AdaBoost, and Logistic Regression can stabilize predictions.

4. Additional Preprocessing

- Lemmatization (better than stemming)
- Bigram features ("free entry", "win prize", "urgent reply")
- URL, phone number, and currency detectors

These additions could push accuracy above 99%, comparable to production systems.

10 Conclusion

This assignment demonstrates the successful design of a complete spam classification system from scratch. The pipeline covers:

- Data cleaning and preprocessing
- Exploratory data analysis
- Bag-of-words vectorization
- Implementation of four classical ML models
- Performance comparison
- Automatic classification of unseen emails

The Multinomial Naive Bayes classifier achieved:

$$\boxed{98.06\% \text{ accuracy, } 92.25\% \text{ precision}}$$

Due to the statistical nature of spam keywords and the independence structure of words, Naive Bayes was the most effective algorithm.

With additional improvements such as TF-IDF weighting and keyword-enhanced priors, performance can become comparable to modern large-scale spam detection systems.