Spam/Ham Classification Project Report

---

1. Problem Statement

Spam messages are a significant nuisance in digital communication, often resulting in lost productivity, wasted time, and even security risks through phishing and malicious content. The objective of this project is to develop a machine learning model to accurately classify text messages as either **spam** (unsolicited or promotional messages) or **ham** (legitimate messages).

A major challenge in this project is dealing with an **imbalanced dataset**, where the number of legitimate (ham) messages far exceeds the number of spam messages. This imbalance can lead to biased predictions where the model favors the majority class, resulting in poor spam detection.

---

2. Dataset Overview

The dataset contains text messages labeled as either **spam** or **ham**. It has the following structure:

- **Label**: Identifies whether the message is spam or ham.
- **Message**: The actual content of the text message.

Class Distribution:

- **Ham**: 4825 messages (86.6%)
- **Spam**: 747 messages (13.4%)

The imbalance in the dataset necessitated special handling to ensure the model could effectively classify both categories.

---

3. Methodology

The project followed a systematic workflow, divided into the following steps:

Step 1: Data Preprocessing

To prepare the text data for model training, the following preprocessing steps were applied:

1. **Text Cleaning**: Removed special characters, punctuation, and converted all text to lowercase to standardize the input.
2. **Tokenization**: Split text into individual words.
3. **Stopword Removal**: Removed frequently occurring words (e.g., "the", "is", "and") that do not carry significant meaning in classification.
4. **Bag of Words Representation**: Used the **Count Vectorizer** to represent text messages numerically. This technique converts the text into a sparse matrix of word frequencies, where each column corresponds to a unique word in the corpus.

Step 2: Model Training

The **Multinomial Naive Bayes** algorithm was chosen for its simplicity and effectiveness in text classification tasks. This algorithm is particularly suited for the Bag of Words representation because it assumes feature independence and works well with word frequency data.

Why Multinomial Naive Bayes?

- Handles large feature spaces effectively (e.g., thousands of unique words).
- Computationally efficient, suitable for real-time applications.
- Performs well in text classification tasks, where the features (word counts) are conditionally independent.

Step 3: Addressing Class Imbalance

The dataset's significant imbalance between ham and spam messages posed a challenge:

- Without mitigation, the model tended to predict the majority class (ham) for most inputs.
- The imbalance was handled by ensuring proper train-test splitting and validating model performance using metrics like precision, recall, and F1-score rather than accuracy alone.

Step 4: Model Evaluation

The model's performance was evaluated on a testing set (20% of the dataset) using the following metrics:

- **Accuracy**: Measures the overall percentage of correctly classified messages.
- **Precision**: Evaluates how many of the predicted spam messages were actual spam.
- **Recall**: Assesses the model’s ability to detect spam messages among all actual spam messages.
- **F1-Score**: A harmonic mean of precision and recall, balancing their trade-off.

---

4. Problems Faced and Solutions

Problem 1: Class Imbalance

- **Issue**: The dataset had far more ham messages than spam, leading the model to predict "ham" most of the time.
- **Solution**: Ensured balanced representation during training by validating the model's performance with metrics that consider both classes (e.g., precision, recall). No oversampling was applied in this case due to sufficient class separation using Bag of Words.

Problem 2: Model Bias Toward "Ham"

- **Issue**: The model initially predicted "ham" for almost all messages.
- **Solution**: Adjusted preprocessing, ensuring vectorization captured relevant patterns in spam messages. Regularized the Naive Bayes model to handle sparse features better.

Problem 3: Overfitting

- **Issue**: The model showed high accuracy on the training data but slightly underperformed on the testing set.
- **Solution**: Ensured proper train-test splitting and used cross-validation during hyperparameter tuning.

---

5. Results

The final model achieved the following metrics on the testing set:

- **Accuracy**: 97%
- **Precision** (Spam): High precision, ensuring the model rarely misclassifies ham as spam.
- **Recall** (Spam): Balanced recall, indicating that the model detects most spam messages.
- **F1-Score**: Strong F1-score, reflecting a good balance between precision and recall.

---

6. Deployment

The trained model was deployed as a **Flask web application** with the following features:

- Users can input a text message, and the app classifies it as spam or ham.
- The app is built with inline HTML and CSS for a clean user interface.
- The backend uses the trained Multinomial Naive Bayes model to process and predict messages in real-time.

---

7. Conclusion

This project successfully developed a spam/ham classification model using the **Bag of Words representation** and **Multinomial Naive Bayes** classifier. By addressing the challenges of text preprocessing and class imbalance, the model achieved a high accuracy of 97%.

The deployment of the model as a Flask web application demonstrates its practical utility for end-users. This approach can be extended to other text classification problems or scaled for larger datasets.

---

8. Future Work

- Experiment with advanced text representations, such as **TF-IDF** or word embeddings (e.g., Word2Vec, GloVe), for potentially better performance.
- Implement real-time feedback mechanisms to improve model predictions over time.
- Apply similar techniques to other text-based classification tasks, such as sentiment analysis or topic modeling.