Spam Email Classification Report

1. Title

Comparative Analysis of Logistic Regression (TF-IDF) and Multinomial Naïve Bayes (Bag-of-Words) for Email Spam Detection

2. Abstract

This report presents the design, implementation, and evaluation of two supervised machine learning pipelines for binary email spam detection ("spam" vs. "ham"). The first pipeline employs TF-IDF feature extraction with a Logistic Regression classifier; the second uses a custom pre-processing token analyzer with Count Vectorization and a Multinomial Naïve Bayes classifier. Both models were trained and tested on the *mail_data.csv* dataset. We compare their performance using accuracy, precision, recall, F1-score, and confusion matrices, and discuss trade-offs in speed, interpretability, and robustness. Recommendations and future enhancement directions are provided.

3. Introduction

Unsolicited bulk email (spam) poses security, productivity, and resource challenges. Machine learning offers scalable, data-driven filtering. This project builds two classical baselines that are commonly used in production-grade pipelines due to their efficiency and strong performance on sparse high-dimensional text data. The objective is to (1) preprocess raw email text, (2) transform text into numerical representations, (3) train and evaluate two models, and (4) provide comparative insights and improvement avenues.

4. Dataset Description

- **Source:** *mail_data.csv* (provided dataset).
- **Columns:**
- `Category`: Label string ('spam' or 'ham').
- `Message`: Raw email message content (text).
- **Derived Column:** `spam` (binary: 1 = spam, 0 = ham) in second pipeline; alternatively 0 = spam, 1 = ham in first pipeline (note: label encoding inconsistency—see Section 6.3).
- **Size:** *N_total* = **(fill actual number)** rows before cleaning.
- **Class Distribution (before cleaning):**
- Spam: **(count_spam)** (*p_spam%*)
- Ham: **(count_ham)** (*p_ham%*)
- **Duplicates Removed: (n_duplicates)**
- **Null Handling:** Replaced null/NaN text fields with empty string.

Recommendation: Standardize a single label convention across all experiments (e.g., spam = 1, ham = 0) to avoid confusion in downstream evaluation.

5. Exploratory Data Analysis (EDA)

- **Frequency Plot:** Count plot of spam vs. ham shows **(brief observation: e.g., class imbalance / near balance)**.
- **Average Message Length (optional):** Spam tends to have **(longer/shorter)** average length than ham (compute mean token length to confirm).
- **Vocabulary Size:** After pre-processing: **(vocab_size)** unique tokens (Count Vectorizer pipeline).
- **Stopwords & Punctuation:** High presence of promotional words (e.g., "free", "win", "offer") in spam messages (qualitative observation).

6. Methodology

6.1 Pre-Processing Steps

Step	Logistic Regression Pipeline	Naïve Bayes Pipeline
Text Normalization	Lowercasing (via TF-IDF `lowercase=True`)	Manual: punctuation removal, lowercasing in custom analyzer
Tokenization	Default word tokenization	Custom list comprehension splitting on whitespace
Stopword Removal	Built-in English stopword list in `TfidfVectorizer(stop_words='english')`	Manual removal using NLTK stopwords set
Punctuation Removal	Implicit (tokens containing punctuation often excluded)	Explicit list comprehension filter
Feature Weighting	TF-IDF (term frequency * inverse document frequency)	Raw counts
Minimum Frequency	`min_df=1` (no pruning)	Not explicitly set (defaults)

6.2 Feature Extraction

- **TF-IDF Vectorizer:** Produces weighted sparse matrix emphasizing discriminative terms.
- **Count Vectorizer (Custom Analyzer):** Produces sparse integer matrix of token counts.

6.3 Label Encoding Note

- Pipeline 1: `Category` mapped to (spam = 0, ham = 1).
- Pipeline 2: New column `spam` mapped to (spam = 1, ham = 0).

Action: Harmonize to a single mapping (e.g., spam = 1) before comparing confusion matrices to avoid misinterpretation of metrics (especially recall for spam class).

6.4 Train/Test Split

- Split ratio: 80% train / 20% test (`test_size=0.2`).
- `random_state` : 3 (Logistic Regression) and 0 (Naïve Bayes) — *different seeds introduce variance*.

  Recommendation: Use the **same** `random_state` for fair comparison.

6.5 Algorithms

1. **Logistic Regression** (L2 regularized linear classifier solving a discriminative optimization problem).
2. **Multinomial Naïve Bayes** (Generative probabilistic model assuming conditional independence of features given class).

6.6 Model Training & Serialization

- **Serialization:** Used `pickle` for both `vectorizer.pkl` and `model.pkl` . (Ensure versioning: record Python, scikit-learn, NLTK versions.)
- **Security Note:** Never unpickle untrusted files; restrict loading to controlled environment.

7. Evaluation Metrics

For binary classification (positive class defined as *spam*):

- **Accuracy:** (TP + TN) / (All)
- **Precision (Spam):** TP / (TP + FP)
- **Recall (Spam / Sensitivity):** TP / (TP + FN)
- **F1-Score:** 2 * (Precision * Recall) / (Precision + Recall)
- **Confusion Matrix:**
- TP: Correctly predicted spam
- TN: Correctly predicted ham
- FP: Ham misclassified as spam
- FN: Spam misclassified as ham

7.1 Reported Results (Fill from your outputs)

Metric	Logistic Regression (TF-IDF)	Multinomial NB (Counts)
Train Accuracy	**(train_acc_logreg)**	**(train_acc_nb)**
Test Accuracy	**(test_acc_logreg)**	**(test_acc_nb)**
Precision (Spam)	**(prec_logreg)**	**(prec_nb)**
Recall (Spam)	**(rec_logreg)**	**(rec_nb)**
F1 (Spam)	**(f1_logreg)**	**(f1_nb)**
Inference Time per 1k msgs (ms)*	**(time_logreg)**	**(time_nb)**

*Optional: measure using `time` module.

7.2 Confusion Matrices (Test Set)

	Predicted Spam	Predicted Ham
Actual Spam	TP = **(tp_logreg)**	FN = **(fn_logreg)**
Actual Ham	FP = **(fp_logreg)**	TN = **(tn_logreg)**

	Predicted Spam	Predicted Ham
Actual Spam	TP = **(tp_nb)**	FN = **(fn_nb)**
Actual Ham	FP = **(fp_nb)**	TN = **(tn_nb)**

Ensure the positive class definition matches across both tables.

8. Comparative Analysis

Aspect	Logistic Regression + TF-IDF	Multinomial Naïve Bayes + Counts
Strengths	Handles correlated features; can yield higher F1 with sufficient data	Extremely fast; strong baseline on sparse text
Weaknesses	Slower training; may overfit without regularization tuning	Independence assumption; may underweight rare but important terms
Interpretability	Coefficients indicate word influence (sign & magnitude)	Log probabilities for each feature per class
Robustness to Rare Words	TF-IDF down-weights common & highlights rare discriminators	Rare words have minimal impact unless counts accumulate
Memory Footprint	Similar sparse structure; slightly larger if vocabulary identical	Slightly smaller model (just log probabilities)
When Preferred	Larger datasets, nuanced contextual differentiation	Baseline, limited compute, rapid prototyping

Observation (Example Placeholder): Logistic Regression achieved higher precision, reducing false positives (ham flagged as spam), while Naïve Bayes delivered comparable recall with faster inference.

9. Error Analysis (Qualitative)

- **False Positives:** Likely ham messages containing promotional keywords ("offer", "free") but legitimate context.
- **False Negatives:** Spam with obfuscated words, informal spelling, or very short messages.

Next Step: Collect misclassified samples, inspect top coefficient weights (Logistic) and conditional log probabilities (NB) to refine preprocessing (e.g., stemming, handling numerics, normalizing URLs).

10. Deployment Considerations

Component	Recommendation
Model Packaging	Bundle `vectorizer.pkl` + `model.pkl` with version metadata JSON
Inference API	REST endpoint `/predict` accepting JSON `{"message": "..."}`
Threshold Adjustment	For Logistic Regression, adjust decision threshold >0.5 if prioritizing precision
Monitoring	Track drift: daily spam ratio, precision@k, top new tokens
Security	Sanitize input, restrict pickle loading to signed artifacts

11. Improvements & Future Work

1. **Data Augmentation / Expansion:** Incorporate additional labeled corpora to diversify spam patterns.
2. **Normalization:** Add lowercasing consistency, URL/email placeholder tokens, number normalization.
3. **Advanced Models:** Try linear SVM, Complement Naïve Bayes, or transformer embeddings (e.g., DistilBERT) with fine-tuning.
4. **Feature Engineering:** Include character n-grams (captures obfuscation: "fr.ee", "0ffer").
5. **Threshold Optimization:** Optimize probability threshold using ROC/PR curves for desired precision-recall balance.
6. **Handling Class Imbalance:** If imbalance is significant, use class weighting (Logistic) or sampling strategies.
7. **Explainability:** Provide top indicative tokens for each class in model card.
8. **Pipeline Consolidation:** Wrap preprocessing + model in a single `Pipeline` object for safer serialization.

12. Ethical & Practical Considerations

- **False Positives Impact:** Legitimate emails marked as spam may cause missed opportunities; precision must be monitored.
- **Adaptation to Evolving Spam:** Periodic retraining required as adversaries change tactics.
- **User Privacy:** Avoid storing full message text post-prediction unless necessary; anonymize logs.

13. Conclusion

Both evaluated models provide effective baselines for spam detection. Multinomial Naïve Bayes offers speed and simplicity, making it an excellent starting point. Logistic Regression with TF-IDF may achieve higher discriminative performance, especially when tuned and provided with more data. A unified, reproducible pipeline with consistent labeling and enhanced feature engineering is recommended for the next iteration.