Spam Detector (SMS/Email) — End-to-End ML Project Report

Generated on 2025-08-12 11:06

Executive Summary

I built a supervised text classifier to detect spam using TF-IDF features and a Multinomial Naive Bayes model. After cleaning and stratified splitting, I tuned the decision threshold on the validation set to t=0.13. On the held-out test set, I achieved accuracy 97.9%, ROC-AUC 0.982, and PR-AUC 0.949 with only 2 false positives and 14 false negatives (spam recall ≈ 0.85).

1) Problem & Data

Goal: Classify a short message as SPAM (1) or HAM (0).

Dataset: Kaggle SMS spam dataset (cleaned to spam_clean.csv).

Rows (cleaned): 5,158 | Spam: 642 | Ham: 4,516 | Spam %: 12.45%

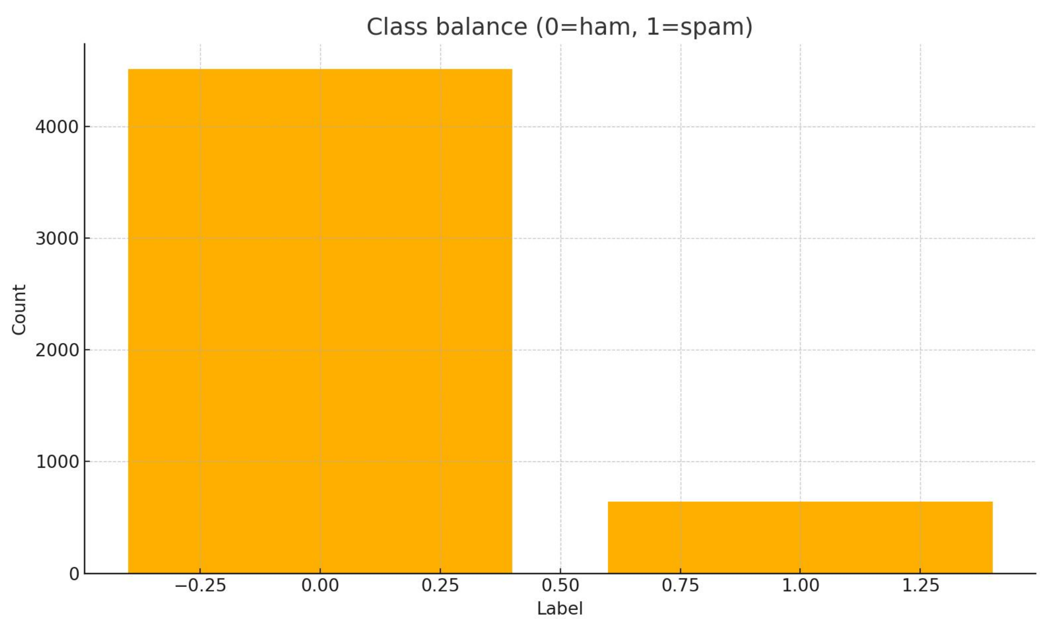

Figure 1. Class balance (imbalanced: ~12.5% spam).

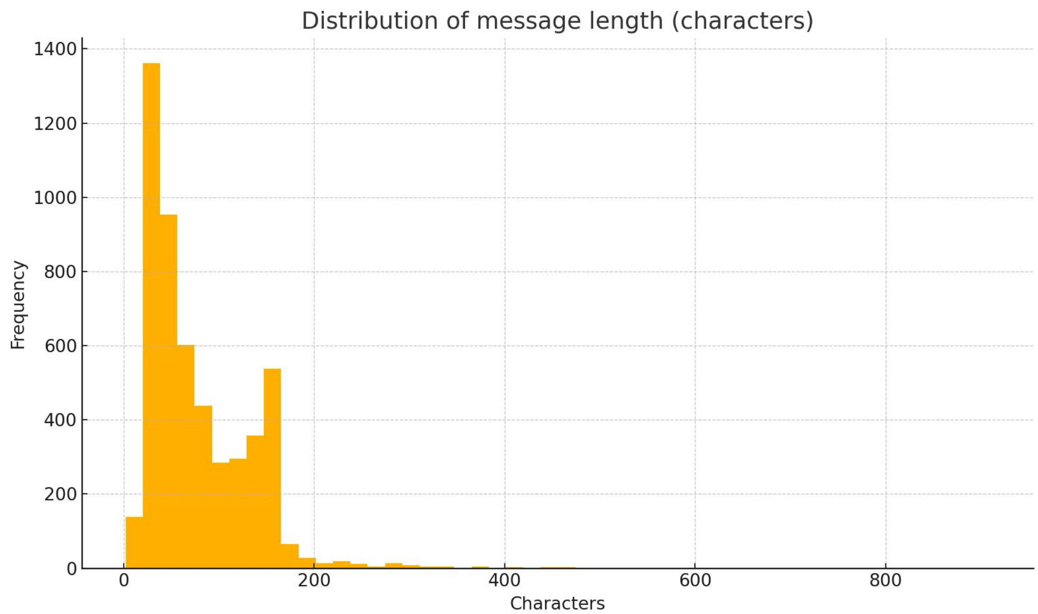

Figure 2. Message length distribution (characters).

2) Ingestion & Cleaning — Why and How

• Robustly read CSV (handled encodings).

• Standardized columns to text + label (1=spam, 0=ham).

• Dropped unnamed columns and 414 exact duplicate messages to reduce leakage/noise.

• Basic EDA to understand imbalance and length distribution.

3) Splitting Strategy — Avoiding Leakage

I stratified three-way split (70% train, 15% validation, 15% test) with fixed random seed.

Split	Rows	Spam %
Train	3610	12.44%
Val	774	12.53%
Test	774	12.40%

All splits ≈ overall spam rate (12.45%), confirming stratification worked.

4) Features & Model — Why These Choices

• TF-IDF with n-grams (1–2) and min_df=2 to balance signal and noise; bigrams capture phrases like “claim now”.

• No stop-word removal; in short texts, common words can carry signal via bigrams.

• Multinomial Naive Bayes for sparse text and strong baseline performance.

Validation vocabulary size: 8699 features.

5) Validation & Threshold Tuning

Default threshold (0.50) was too conservative (missed many spam). Tuned threshold on validation to maximize F1.

Setting	t	Spam Precision	Spam Recall	Spam F1	Confusion (rows=actual)
Default	0.50	1.000	0.680	0.810	[[677, 0], [31, 66]]
Best F1	0.13	0.989	0.887	0.935	[[676, 1], [11, 86]]

6) Final Test Performance (Locked t=0.13)

Class	Precision	Recall	F1	Support
Ham (0)	0.9797	0.9971	0.9883	678
Spam (1)	0.9762	0.8542	0.9111	96

Confusion: [[676, 2], [14, 82]] | ROC-AUC: 0.9819 | PR-AUC: 0.9489

7) Error Analysis: What We Miss and Why

Counts → correct: 758, FN: 14, FP: 2.

Patterns in FNs: adult/premium-rate promotions using slang, shortcode pricing (e.g., “150p”), or odd encodings; word-only features can miss obfuscated tokens.

Mitigations: threshold sweep; add character n-grams; consider **Logistic Regression with class weights.**

8) Deployment: What & Why

Artifacts saved: spam_nb_tfidf.joblib (full sklearn pipeline) and threshold.json (decision t).

Demo(live for 1 week only as I have not subcription of it): Gradio app returning (label, probability). Initial return-type bug fixed by returning a tuple to match two outputs.

9) Reproducibility & Good Practices

• Stratified splits with fixed random seed.

• No leakage: fit vectorizer/model on train; tune on validation; evaluate once on test.

• Removed duplicate messages to reduce leakage; saved versioned artifacts.

10) Next Steps

1) Add word + character n-gram TF-IDF; re-tune threshold.

2) Try Logistic Regression (class_weight='balanced'); compare with NB.

3) Add URL/phone masking and minimal normalization.

4) Package as FastAPI microservice; add logging & drift checks.