

Email Spam Classifier
(AI Interview Task Submission)

Submitted by:

Name: Md. Emdadul Hasan Shishir
Role: AI Developer (Interview Candidate)
Email: emdadul2251@gmail.com
Phone: 01814633663







Submitted to:

ScaleUp Ads Agency
Interview Assessment Panel

Project Details:

Project Title: Email Spam Classification using Machine Learning
Dataset Source: [Kaggle – Email Spam Classification Dataset](#)
Programming Language: Python
Development Environment: Google Colab
Date of Submission: 03.10.2025

Included Deliverables:

-  Full Report (this document)
-  Codebase (.ipynb notebook)
-  Video Demonstration (3 minutes)
-  Comparative Analysis
-  Saved Model (.joblib file)
-  README (How to Run Instructions)

Note: This project demonstrates an end-to-end spam email detection pipeline using NLP and ML. The best-performing model from this experiment is Random Forest (TF-IDF) with 98.41% accuracy.

Final Report — Email Spam Classifier

1. Executive Summary

This project implements an end-to-end machine learning pipeline to classify emails as spam or ham using only the email text. Multiple classical NLP feature pipelines (CountVectorizer, TF-IDF) and four supervised algorithms were evaluated. Based on the results recorded in the notebook, Random Forest (TF-IDF) achieved the highest accuracy and the best F1-score and is selected as the final model for this submission.

2. Objective

Build and evaluate a binary classifier that labels email messages as **spam** or **ham**, and provide the deliverables: dataset/source link, codebase (notebook), demo video, comparative analysis, full report (this document), and a README.

3. Dataset

- **Source:** Kaggle — [Kaggle – Email Spam Classification Dataset](#) (downloaded to Colab via Kaggle API)
- **Samples:** 83,448 emails
- **Columns used:** label, message
- **Label distribution (notebook):**
 - Spam: 43,910
 - Ham: 39,538

Details, inspection outputs, and sample messages are available in the notebook's EDA cells.

4. Preprocessing

- Normalized column names to label and message.
- Converted numeric label values (1/0) to spam/ham.
- Cleaned messages with `clean_text()`:
 - Lowercased text
 - Removed URLs and email addresses
 - Removed non-letter characters and collapsed whitespace
- Optionally remove stopwords (not forced in final runs).
- Train/test split: **70% train / 30% test**, stratified by label.

5. Feature Engineering

Two text vectorizations were used:

1. **CountVectorizer** — unigram + bigram (ngram_range=(1,2)), min_df=2
2. **TfidfVectorizer** — unigram + bigram (ngram_range=(1,2)), min_df=2

TF-IDF was the preferred representation for final model training due to better emphasis on informative, rare terms.

6. Models Trained

- **Multinomial Naive Bayes** (CountVectorizer)
- **Multinomial Naive Bayes** (TF-IDF)
- **Logistic Regression** (TF-IDF)
- **Random Forest** (TF-IDF)

All models were trained on the same train set and evaluated on the same test set for a fair comparison.

7. Comparative analysis of algorithms

Model	Feature Extraction	Accuracy	Precision (spam)	Recall (spam)	F1 (spam)	Train time
Naive Bayes (CountVectorizer)	CountVectorizer	97.62%	0.9898	0.9647	0.9771	36.77 s
Naive Bayes (TF-IDF)	TF-IDF	97.98%	0.9887	0.9728	0.9807	42.36 s
Logistic Regression (TF-IDF)	TF-IDF	98.27%	0.9805	0.9867	0.9836	61.17 s
Random Forest (TF-IDF)	TF-IDF	98.41%	0.9795	0.9905	0.9849	1973.25 s

Note: The Random Forest model achieved the **highest accuracy and F1-score** in this experimental run. Training time for Random Forest is substantially higher than other models.

8. Observations and Analysis

- **TF-IDF vs CountVectorizer:** TF-IDF consistently performed better by upweighting discriminative terms (e.g., “free”, “winner”, “clicker” tokens).
- **Naive Bayes:** Very fast and strong baseline with excellent precision; useful where compute/latency is critical.
- **Logistic Regression:** Good all-around performance with fast training/inference; good candidate for production with lower compute cost than Random Forest.

- **Random Forest:** Achieved the highest accuracy and F1 on the test split in this run, but requires substantially longer training time (about 30 minutes on the runtime used). This suggests Random Forest captures additional patterns but with high computational cost.
- **Tradeoff:** There is a tradeoff between maximal accuracy (Random Forest) and computational efficiency & simplicity (Logistic Regression or Naive Bayes).

9. Final Model Selection

Selected final model: Random Forest (TF-IDF) — chosen because it produced the highest accuracy and best F1 score in the actual experiment outputs recorded in the notebook.

Deployment note: If operational constraints (latency or compute) matter, Logistic Regression (TF-IDF) is a close second and the recommended alternative because it keeps near-top accuracy with far lower compute/time requirements.

10. Confusion Matrices & Error Analysis

(See the notebook for the plotted confusion matrices and classification reports for each model.)

High-level error analysis:

- Most false positives/negatives involve borderline promotional messages or short messages lacking clear spam tokens.
- Interpretable coefficients (Logistic Regression) and feature importance ranking (Random Forest) confirm domain intuition (spam tokens dominate spam class).

10. Confusion Matrices & Error Analysis

(See the notebook for the plotted confusion matrices and classification reports for each model.)

High-level error analysis:

- Most false positives/negatives involve borderline promotional messages or short messages lacking clear spam tokens.
- Interpretable coefficients (Logistic Regression) and feature importance ranking (Random Forest) confirm domain intuition (spam tokens dominate spam class).

11. Deliverables (included in submission)

1. **Dataset Source link:** Kaggle — [purusinghvi/email-spam-classification-dataset](https://kaggle.com/purusinghvi/email-spam-classification-dataset) (include the exact URL in README).
2. **Codebase:** Md_Emdadul_Hasan_Shishir_Email_Spam_Classifier.ipynb — full Colab notebook with all cells.
3. **Video demo:** video_demo.mp4 — 2–3 minute demo showing notebook highlights and live inference.
4. **Comparative Analysis:** Section 7 & 8 of this report (and Markdown in notebook after Cell 13).

5. **Full Report:** this document (PDF).

6. **Short README:** included below.

13. README

Email Spam Classifier — README

1. Open the notebook: Md_Emdadul_Hasan_Shishir_Email_Spam_Classifier.ipynb
2. Upload your kaggle.json (Kaggle API token) to Colab (if needed) and run the download cell.
3. Run cells sequentially (1 → 17).
4. The best performing model from this run is Random Forest (TF-IDF). Saved pipeline: `/content/rf_tfidf_pipeline.joblib`
5. For inference, load saved model with `joblib.load('/content/rf_tfidf_pipeline.joblib')` and call `predict([clean_text(your_message)])`.

4. Future Work

- Reduce Random Forest training time (distributed training or fewer estimators) or use XGBoost/LightGBM tuned for sparse data.
- Evaluate transformer-based models (BERT) for even better contextual detection (requires more compute).
- Implement threshold tuning on Logistic Regression probabilities to optimize precision vs recall based on business requirements (avoid false positives on inbox filtering).
- Deploy as a REST API (FastAPI / Flask) for production inference.