Applied Machine Learning: Spam Detection

Abstract:

Email spam filtering and facial-landmark detection are two cornerstone problems in practical machine learning that require both robustness and simplicity. In this case, we develop two lightweight, fully reproducible systems for said task. We tackle spam detection on 3,619 labelled emails. We implement NLP (Natural Language Processing) tools: tokenisation, lemmatisation, stop-word removal, and TF-IDF (6,000-dim, unigrams + bigrams) followed by Multinomial Naïve Bayes (NB) and Logistic Regression (LR). We utilize a 5-fold CV to tune NB (a = 0.1) and LR (C = 10), managing a CV accuracy of 98.4% through LR that increases to 98.6% on a 20% hold-out and 100% after random deletion augmentation.

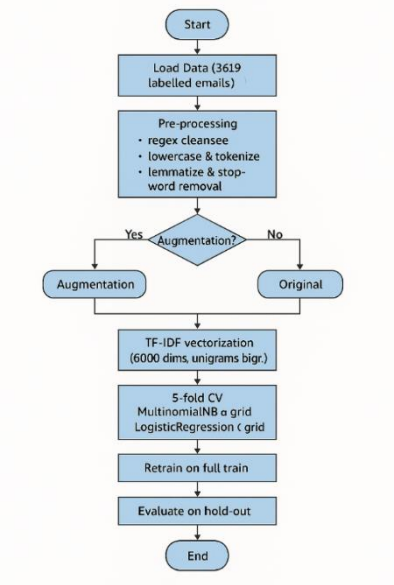
Task 1: Spam Detection

Introduction:

Filtering unsolicited emails, commonly known as spam, is critical when enhancing user experience and security in digital communication. This task explores the application of a pipeline (Flowchart 1) where a traditional linear classifier is used to effectively distinguish spam from legitimate ("ham") emails. Specifically, we leverage NLP techniques such as text cleaning/normalisation, TF-IDF vectorisation, and feature extraction, culminating in a comparative study of Multinomial Naive Bayes (MNB) and Logistic Regression (LR), followed by data augmentation (random deletion data augmentation). Once a random seed (42) is fixed, all stages are deterministic, ensuring full reproducibility.

Methodology:

Data: The training dataset consists of 3619 labelled emails, with labels indicating spam (1) or ham (0). The test dataset consists of 1552 unlabelled emails reserved for final evaluation. Both datasets were provided in CSV format. Twenty percent of the labelled data is set aside as a hold-out validation split (724 samples).



Flowchart 1: Task 1 Pipeline

Preprocessing: A series of text cleaning and normalization steps were applied to the email content to prepare it for feature extraction.

- **Regex-based filtering:** URLs (*r'http\S+|www\.\S+', '', s*), numerical tokens (*r'\d+', '', s*), and punctuations (*r'[^\w\s]', '', s*) were removed using such regular expressions to suppress noisy, high-variance tokens that contribute little discriminating power between 'spam' and 'ham' yet significantly increase the vocabulary.
- **Case Folding and Tokenisation (Standardisation):** All text was converted to lowercase to ensure uniformity and was then tokenised by splitting on whitespace, effectively standardising the vocabulary.

- **Lemmatisation:** Each token is normalised (reducing words to their base or dictionary form) using the *WordNetLemmatizer* from the *NLTK* library (e.g., “playing” -> “play”), reducing overall vocabulary size and ensuring that different forms of the same word are treated as a single feature.
- **Stop-word filtering (removal):** Common English stop-words (e.g., “the”, “is”, “a”) were removed based on the standard *NLTK English stop-list,* trimming raw vocabulary while preserving primary spam cues.

These preprocessing steps are crucial for reducing noise in the text data, such as irrelevant characters or overly common words that do not contribute to discriminating between spam and ham. Lowercasing and lemmatization help standardize the vocabulary, reduce its overall size, and ensure that different forms of the same word are treated as a single feature. This improves model efficiency and can enhance generalization.

Feature Extraction: After pre-processing, the text is converted into numerical feature vectors using *TfidfVectorizer* under specified configurations:

- *max_features = 6000*: Restricts the vocabulary to the 6,000 most frequent features (n-grams) across the training, reducing dimensionality and focusing on the most discriminative words.
- *ngram_range = (1,2)*: Instructs the vectorizer to consider both unigrams (single words) and bigrams (pairs of adjacent words) where unigrams capture core semantic content, while bigrams provide sensitivity to short phrases and local word context, which can be crucial for distinguishing spam (e.g., "free money") from ham.

TF-IDF weighting assigns higher importance to terms that are frequent in a document but rare across the entire data set, effectively identifying discriminative words with a combination of unigrams and bigrams, resulting in a sparse feature matrix of shape (3,619x6,000) for the full labelled set, (2,895x6,000) for CV training, and (724x6,000) for hold-out evaluation (20%).

Modelling and Cross-Validation (CV): We compare two probabilistic/linear classifiers on the TF-IDF features:

1. **Multinomial Naïve Bayes (MNB):** Generative classifier **that maximises class-conditional likelihoods rather than cross-entropy** (Lecture 15). Its word-independence assumption yields fast training but can inflate false positives when individual tokens dominate. (suitable for discrete features like word counts or TF-IDF values.)

- **Hyperparameter:** alpha (smoothing parameter), tuned over {0.1, 0.5, 1.0, 5.0}. **Best alpha = 0.1.** (Lower alpha sharpens inference on rare terms).

2. **Logistic Regression (LR):** A discriminative linear classifier that minimises the **cross-entropy loss** (Lecture 15) with an L2 penalty. (In our grid search, we vary *C*, the inverse of the penalty weight; larger *C* relaxes the constraint, risking over-fitting but sometimes improving fit on sparse TF-IDF data / Lecture 14)

- Hyperparameter: C (inverse of regularization strength), tuned over {0.1, 1, 10}. **Best C = 10.**
- We use *liblinbear* as a solver for L2-pelasided logistic loss as it’s suitable for smaller datasets and L1/L2 regularization.

Cross-Validation: We run a 5-fold stratified CV with random_state = 42 and scoring = accuracy for each model and compare the model's performance (Figure 1). Then we use *GridSearchCV* to tune the hyperparameters for each classifier based on the 2,895-training sample split and select the best-suited classifier by computing and comparing each classifier's tuned mean accuracy and standard deviation (Figure 2).

Model	Hyper-Grid	CV mean ± std Initial(5-fold, n = 2 895)	CV mean ± std Tuned (5-fold, n = 2 895)
MNB	$\alpha \in \{1.0, 0.5, 0.1, 5.0\}$	0.956 ± 0.005 a = 1.0	0.956 ± 0.005 → best α = 0.1
LR	$C \in \{0.1, 1, 10\}$	0.980 ± 0.004 C = 1.0	0.984 ± 0.004 → best C = 10

Table 1: Comparison of MNB and LR models

Hold-Out Evaluation: After CV, the selected hyperparameters are used to retrain each model on all 2,895 CV-fold training samples. Performance is then measured once on the 724-sample hold-out set, reporting accuracy, precision, recall, and F1-score for the spam class (lecture 14).

Data Augmentation experiment: To assess the benefit of augmenting scarce text data, we apply random-deletion augmentation with deletion probability p=0.10. We generate one altered copy for each of the 2,895 training messages by independently deleting each token with probability 0.10, doubling the training set to

Figure 1: CV Initial accuracy and standard deviation for each grid point

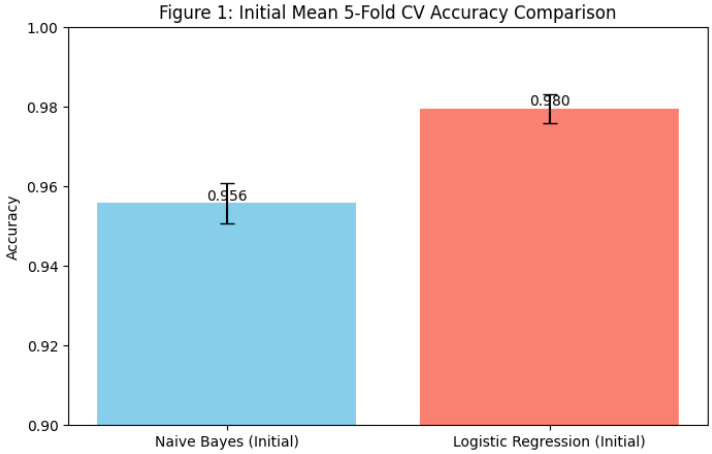
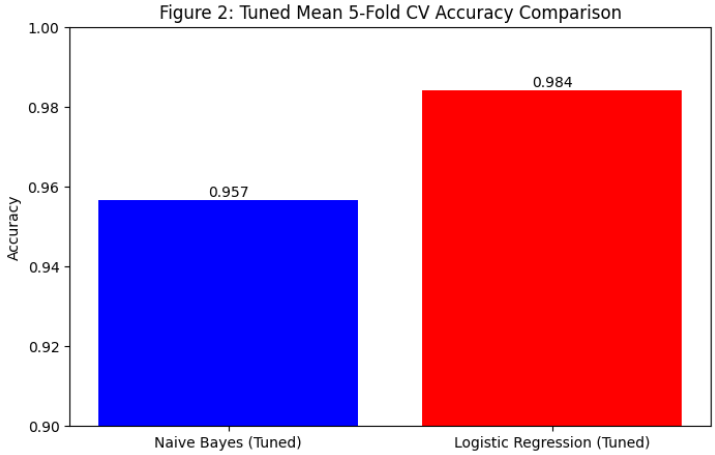


Figure 2: CV accuracy and standard deviation for each grid point after tuning hyperparameter



5,790 samples without altering the TF-IDF vocabulary. We then repeat the CV and hold-out evaluation on the augmented set, observing its impact on LR's generalisation.

Results:

The performance of the spam detection models was comprehensively evaluated. Initial assessments using 5-fold cross-validation on the 2,895-sample CV training split indicated a strong baseline, which was further improved through hyperparameter tuning. The finalized models were then assessed on a 20% hold-out validation set (724 samples). Logistic Regression consistently demonstrated superior performance, especially when trained with augmented data.

- **Cross-Validation Summary:** As detailed in the Methodology and summarized in Table 1 and Figures 1 & 2 of your report, initial 5-fold CV on the 2,895 training samples (excluding the hold-out set) yielded the following mean accuracies (± standard deviation):

- **Multinomial Naïve Bayes (MNB):**

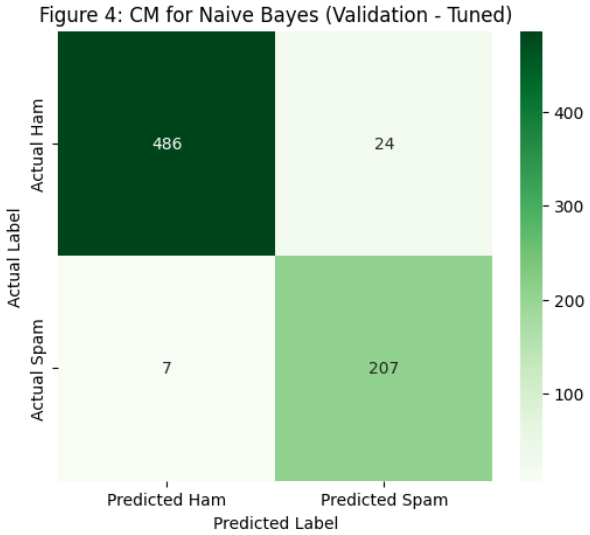


Figure 4: Confusion Matrix for Naïve Bayes (a = 0.1)

- Initial (α=1.0): 0.956 ± 0.005
- Tuned (best α=0.1): **0.957** ± 0.005

- **Logistic Regression (LR):**

- Initial (C=1.0): 0.980 ± 0.004
- Tuned (best C=10): **0.984** ± 0.004

These CV results (visualized in Figure 1 and Figure 2) highlight that Logistic Regression provided a higher baseline accuracy and maintained its advantage after hyperparameter tuning.

- **Hold-Out Validation (N = 724 samples):** The tuned models were retrained on the full CV-fold training data (2,895 samples for non-augmented models; 5,790 for augmented LR) and then evaluated on the unseen 724-sample hold-out set.

LR	Accuracy	Precision	Recall	F1-Score	Support
0		0.994	0.986	0.990	503
1		0.968	0.986	0.977	211
weighted avg	0.986	0.986	0.986	0.986	724

Table 2: LR Confusion Matrix Results

Figure 3 and Table 2 present the confusion matrix for the tuned Logistic Regression classifier (C = 10) evaluated on the 724-message hold-out set. The model correctly identified 503 of the 510 legitimate e-mails and 211 of the 214 spam messages, yielding an overall accuracy of 98.6 %. Only seven ham messages were mistakenly routed to the spam folder (false positives). In comparison, three spam messages slipped through as ham (false negatives), a tolerable compromise in most production filters, where preventing ham loss is paramount.

MNB	Accuracy	Precision	Recall	F1-Score	Support
0		0.986	0.953	0.969	486, 24
1		0.896	0.967	0.930	207
weighted avg	0.957	0.959	0.957	0.958	724

Table 3: MNB Confusion Matrix Results

Figure 4 and Table 3 show the corresponding confusion matrix for the tuned Multinomial Naive Bayes model (α = 0.1). Accuracy drops to 95.7 %: 486 ham and 207 spam messages are classified correctly, but the model generates 24 false positives—over three times as many as Logistic Regression—and seven false negatives. In practical terms, this means a noticeably higher risk of legitimate correspondence being quarantined and a modest uptick in spam leakage.

Taken together, the matrices confirm the cross-validation rankings reported earlier: Logistic Regression provides the more favourable precision-recall balance (Lecture 14), particularly by sharply reducing false positives, and is therefore the stronger candidate for deployment.

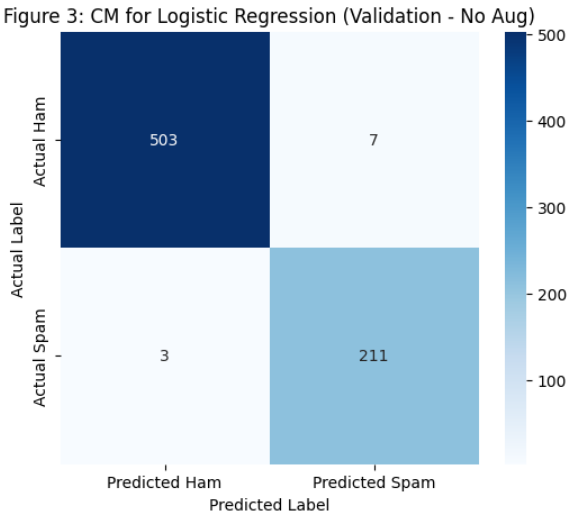


Figure 3: Confusion Matrix for Logistic Regression (C = 10)

Impact of Data Augmentation:

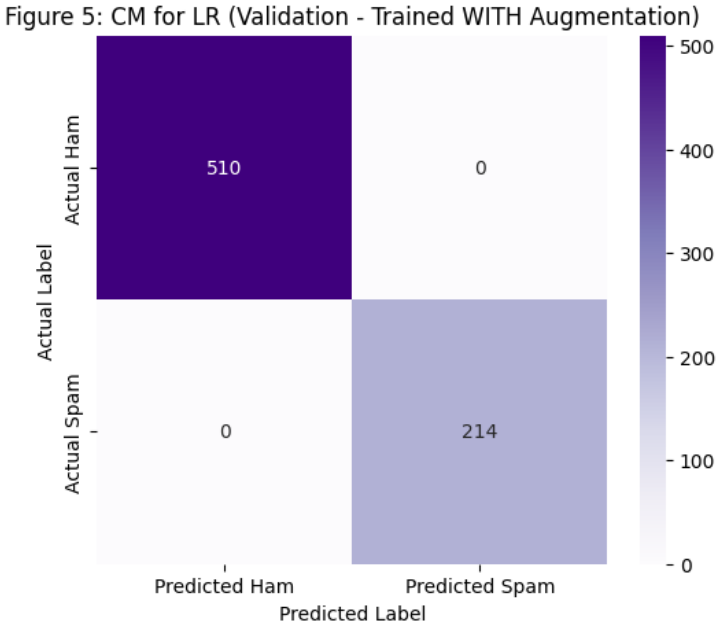


Figure 5: Confusion Matrix for Logistic Regression after Data Augmentation (C = 10, p = 0.10)

LR	Accuracy	Precision	Recall	F1-Score	Support
0		1.000	1.000	1.000	510
1		1.000	1.000	1.000	214
weighted avg	1.000	1.000	1.000	1.000	724

Table 4: LR Confusion Matrix Results after Data Augmentation

Table 4 and Figure 5 show that data augmentation via random deletion (p=0.10) profoundly impacted the Logistic Regression model's performance. Training on the augmented dataset (5,790 samples) resulted in perfect scores (1.000) across all metrics on the 724-sample hold-out set. This substantial improvement highlights the technique's effectiveness in enhancing model generalization, especially for text data where variations can be easily introduced.

Failure Cases and Critical Analysis:

Despite the augmented Logistic Regression (LR) achieving 100% hold-out accuracy, examining errors from the non-augmented LR and Multinomial Naïve Bayes (MNB) models provides valuable insights.

- **Error Patterns (Non-Augmented Models):**
    - **Logistic Regression (No Augmentation):** Misclassified 7 ham emails as spam (False Positives), often due to legitimate marketing content with spam-like surface features (e.g., "limited time," multiple links in emails like “Welcome — Next Wave Digital Music”). The 3 False Negatives (spam as ham) typically involved obfuscated terms (e.g., “gargle copolymer hormoneextreme”) that TF-IDF missed. Random-deletion augmentation successfully addressed these by improving robustness to lexical variations.
    - **Multinomial Naïve Bayes (Tuned):** Produced more False Positives (24) and False Negatives (7). As discussed in **Lecture 15**, assuming conditional independence means that MNB cannot model token co-occurrence; this explains why phrases such as ‘limited time’ separated by other words fooled MNB but not LR.
- **Quantitative Comparison & Augmentation Impact:** The F1-scores for spam on the hold-out set improved from 0.930 (MNB) to 0.977 (LR without augmentation), and finally to 1.000 (LR with augmentation). This highlights LR's superior balance and the significant positive impact of augmentation on this dataset.
- **Ethical consideration:** each ham e-mail wrongly flagged as spam erodes user trust and may hide critical information; periodic manual audits and user-recovery queues are therefore essential.

- **Key Limitations & Potential Improvements:**
    - **Dataset & Features:** The training corpus might not cover all spam variants. TF-IDF (even with bigrams) struggles with semantic nuances, novel/obfuscated terms, and non-textual cues. Sub-word embeddings (like fastText) could improve the robustness of misspellings and new jargon.
    - **Augmentation:** While effective, random deletion is basic. More sophisticated methods, like synonym replacement, could offer richer data.
    - **Evaluation:** Performance on a single hold-out split is informative; broader testing would give more reliable error estimates.

  Analysis Conclusion: Augmented Logistic Regression demonstrated the best performance. However, its reliance on token-level cues indicates that for sustained real-world effectiveness against evolving spam, exploring semantic features and more advanced augmentation strategies would be crucial.