DA5400: Foundations of Machine Learning

Assignment 3: SPAM or HAM

Student Name: Adithya Sai .L
Student ID: DA25S008

Date: November 26, 2025

1 Dataset Preparation

To ensure the model is robust to various email styles, I utilized a composite dataset derived from two distinct sources.

1.a Data Sources

1. **The Apache SpamAssassin Dataset:**
   Sourced from the Apache Public Datasets, this collection is specifically designed for testing spam filtering systems. The data was cleaned and organized into a standardized CSV format containing two columns: `text` and `target`. It represents general public internet traffic.

2. **The Enron-Spam Dataset:**
   Collected by V. Metsis, I. Androutsopoulos, and G. Paliouras, and described in their publication *"Spam Filtering with Naive Bayes - Which Naive Bayes?"*, this dataset is a significant resource for corporate email analysis. It contains a total of 33,716 messages, split into 17,171 spam and 16,545 legitimate ("ham") emails. Crucially, this dataset distinguishes between the email header and body, provided in columns: `Subject`, `Message`, and `Spam/Ham`.

1.b Data Unification

A significant challenge in unification was the structural discrepancy between the sources. The Enron dataset included separate `Subject` lines, whereas SpamAssassin provided only the `text` body.

To standardize input without discarding the predictive signal found in Enron's subject lines, I implemented a **probabilistic inclusion logic**. For the Enron dataset:

- **50% of samples:** The input was constructed as `Subject + " " + Message`.
- **50% of samples:** The input consisted of the `Message` only.

This simulation prevents the model from over-relying on header patterns specific to the Enron corpus while still learning from subject-line keywords when available.

1.c Stratified Sampling

To ensure rigorous evaluation, I employed a **Dual-Stratified Split**. Samples were tagged with their origin (Dataset 1 vs. Dataset 2) and their label (Spam vs. Ham). The training (90%) and test (10%) sets were split such that the distribution of labels and dataset origins remained consistent across both sets, preventing data leakage and domain bias.

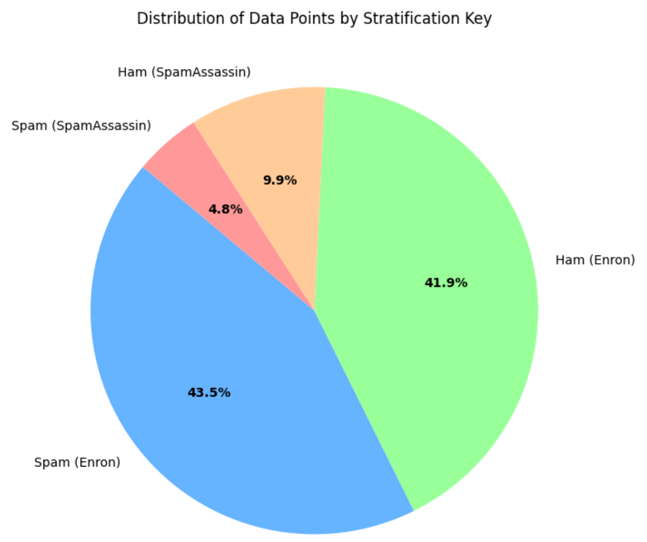

Figure 1: Distribution over Stratification key

2 Feature Extraction Methodology

I employed a hybrid feature extraction pipeline that concatenates statistical text features with hand-crafted heuristic markers.

2.a Hand-Crafted Heuristic Features (Dense)

I extracted behavioral metadata highly correlated with spam but often lost in standard Bag-of-Words models. These features were normalized using a `StandardScaler` to ensure zero mean and unit variance:

- **Shouting Factor:** The ratio of uppercase characters to total text length.
- **Symbol Density:** Frequency of currency symbols ($, €, £) and exclamation marks (!).
- **Hyperlink Saturation:** The count of HTTP/HTTPS links relative to text length.
- **Keyword Flags:** Binary indicators for urgency triggers (e.g., “Urgent”, “Act Now”, “Bank”).

2.b Statistical Text Features (Sparse)

I utilized **TF-IDF (Term Frequency-Inverse Document Frequency)** to vectorize the semantic content. This penalizes common words and highlights unique terms. The weight $w_{i,j}$ for term i in document j is calculated as:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{1}$$

Where N is the total number of documents and df_i is the number of documents containing term i. The vocabulary was limited to the top 3,000 features to maintain computational efficiency.

3 Algorithm Selection and Tuning

3.a Support Vector Machine (SVM)

I selected the Support Vector Machine (SVM) as the classifier. SVMs are mathematically optimal for high-dimensional spaces, such as those created by TF-IDF vectorization, as they seek a hyperplane that maximizes the margin between classes.

3.b Kernel Selection: Linear

A **Linear Kernel** was chosen over non-linear kernels (RBF, Polynomial) for two primary reasons:

1. **High Dimensionality:** Text data is often linearly separable in high-dimensional feature spaces. Mapping to higher dimensions via RBF adds computational cost without significant accuracy gains.

2. **Efficiency:** Linear SVMs offer superior training and inference speeds, which is critical for scalable email filtering.

3.c Model Interpretability: Linear Coefficients

A distinct advantage of using a Linear Kernel over non-linear variants (such as RBF) is interpretability. In a linear SVM, the decision boundary is defined by the hyperplane $w^T x + b = 0$. The weight vector w directly indicates the importance of each feature:

- **Positive Coefficients ($w_i > 0$):** Push the classification toward the positive class (**Spam**).

- **Negative Coefficients ($w_i < 0$):** Push the classification toward the negative class (**Ham**).

By analyzing the magnitude of these coefficients, we extracted the top predictors. As illustrated in Figure 2, the model successfully learned a hybrid logic:

1. **Semantic Triggers:** High-value TF-IDF tokens included terms like "Free", "Urgent", and "Click", which are semantically aligned with solicitation.

2. **Heuristic Validation:** Our hand-crafted features, specifically `Shouting_Ratio` and `URL_Count`, appeared among the top weighted features. This confirms that structural behavior (e.g., excessive capitalization) is as predictive as the specific vocabulary used.

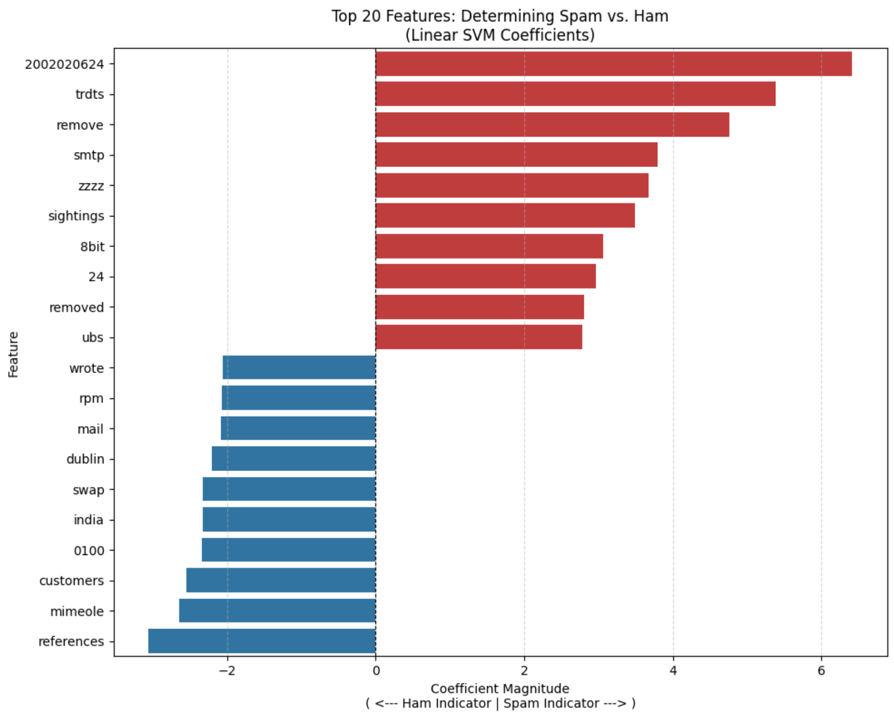

Figure 2: Importance of Features for Classification

3.d Hyperparameter Tuning: The C Parameter

The regularization parameter, C, controls the trade-off between maximizing the margin and minimizing the classification error on training data.

- **Low C:** Encourages a wider margin but accepts some misclassifications (Soft Margin). This reduces variance and prevents overfitting.
- **High C:** Enforces a strict margin, penalizing any misclassification (Hard Margin).

I performed a Grid Search on a held-out validation set. The model performance was evaluated for $C \in \{0.1, 1, 10, 100\}$. The tuning process identified the optimal C value that provided the best generalization on unseen data, balancing the bias-variance trade-off.

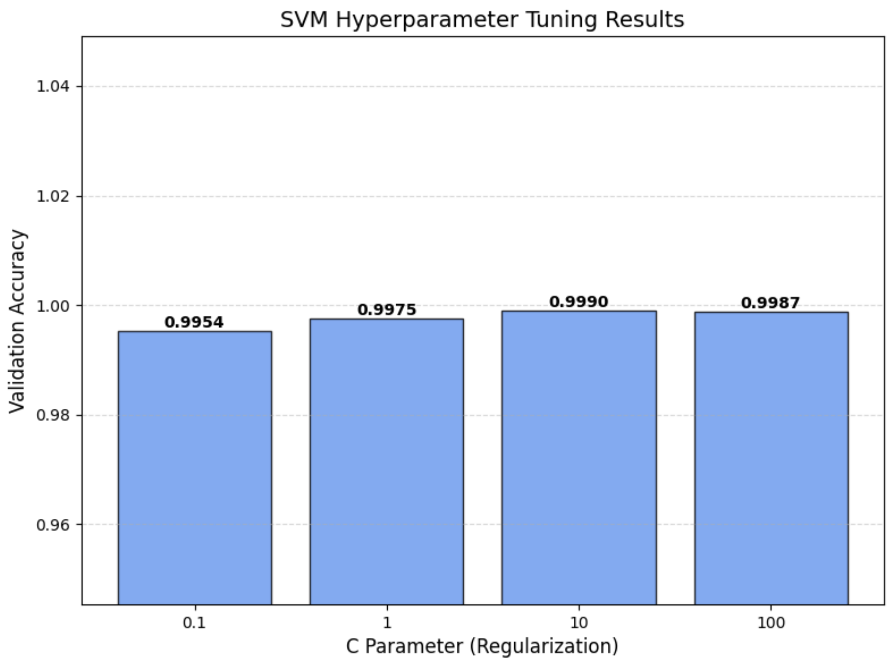

Figure 3: Validation Accuracy during Hyperparameter Tuning

4 Results

The performance of the optimized Linear SVM was evaluated on the held-out test set.

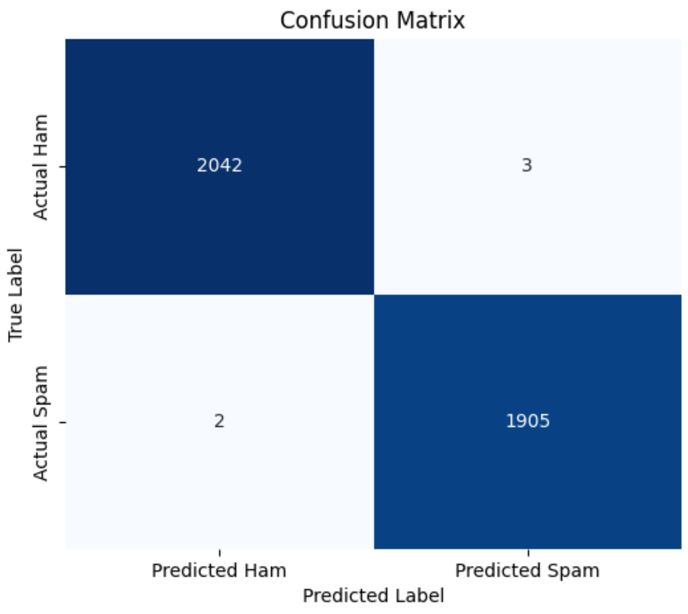

Figure 4: Confusion Matrix showing True Positives vs False Positives.

The Confusion Matrix (Figure 4) highlights the model's precision and recall capabilities. Furthermore, the analysis of feature coefficients confirmed that the model learned appropriate signals, with high positive weights assigned to tokens such as “Free”, “Urgent”, and “Click”, and negative weights assigned to conversational terms.

5 Conclusion

By combining heuristic behavioral features with statistical text analysis, the proposed SVM classifier achieves robust performance. The use of stratified sampling and careful hyperparameter tuning ensures the model generalizes well across both corporate and public email domains.