

**Maria Qamar<sup>1\*</sup> And Nilesh Jain<sup>2</sup>**

1 Research Scholar, Department of Computer Science, Radha Govind University, Ramgarh, Jharkhand India

2. Assistant Professor, Department of Computer Science, Radha Govind University, Ramgarh, Jharkhand India

Corresponding

Maria Qamar

Email id: qamar.m2018@gmail.com

## Continuous Learning and Multimodal Techniques for Next-Generation Spam Detection Systems

### Abstract

The problem of email spam is still extremely topical since the unsolicited bulk messages change in both content and method. This essay examines machine learning-based spam filters on a test email dataset that consists of spam and valid (ham) emails. We use four classifiers, namely Naive Bayes, Support Vector Machine (SVM), Random Forest and Logistic Regression and compare them in terms of measures of accuracy, precision, recall, and F1-score. The models are all high in spam detection accuracy (greater than 90 percent), with SVM being a little better in the overall accuracy (approximately 95.5 percent) and accuracy. Naive Bayes has the best spam recall (around 94.2) but with a more significant number of false positives and a trade-off between spam captures and legitimate email loss. We use extensive text preprocessing (tokenization, stop-word elimination, stemming) and make comparisons of feature representations: word embeddings vs. term frequency-inverse document frequency (TF-IDF). TF-IDF features performed better on this dataset than the average word embeddings on this dataset. In the fight against constantly evolving spam techniques, we consider the use of continuous learning approaches that dynamically update the filter to overcome concept drift, and we consider multimodal approaches that use textual and image data or link analysis to detect material that the purely textual filter fails to detect. As we have found, it is indeed true that using fixed machine learning models is effective in detecting spam and doing it with a high level of accuracy, however next-generation systems do need to constantly learn and make use of several data modalities to serve as a resource to robust and adaptive spam filtering in dynamic real-life scenarios.

**Keywords:** spam detection; continuous learning; concept drift; multimodal; text classification; email security

## Introduction

Unsolicited bulk email (spam) has become an epidemic of the digital communication. Spam messages have comprised a reported 50-85 percent of all the email-traffic in recent years. This flood of spam not only wastes productivity and irritates the user, but it also invites gravest security threats in the case of spam messages which include phishing nodes or malware codes. The financial cost of spam is not trivial - filtering and dealing with spam costs business billions of dollars of productivity and costs of infrastructure. Such issues render proper spam detection a necessity towards safe and useful email systems.

Initial anti-spam programs were based on hand made rule sets and lists of spammers. Though these kind of heuristic solutions (e.g. blocking emails containing specific keywords or domains) may have offered some relief, the spammers soon implement some sort of strategy to circumvent these rules. As an example, spamming started to use symbols to obfuscate a trigger word (e.g. Viagra of Viagra becomes Viagra) or a dynamically generated sender address to get the spam mails passed by blacklists. These evasive schemes compromised fixed filters (Goodman et al., 2005; Abu-Nimeh et al., 2007) and spurred the creation of spam filters based on learning that are auto-enhancing and learn the distinguishing characteristics between spam and legitimate emails based on data. Scholars re-conceived spam filtering as binary text classification: using an email's text (and possibly metadata), a classifier needs to build a judgment whether it is "spam" or "ham" (legitimate).

The Bayesian classifier by Sahami et al. in 1998 was one of the oldest and most effective machine learning models to use spam. Their Naive Bayes filter was trained on the likelihoods of words being in spam or ham and performed well in accuracy with better results compared with simple rules because it can adjust to the true spam in the inbox of the users. This is the monumental work that catalyzed 20 years of research on machine learning in the area of spam detection. Many algorithms have been since considered - such as Support Vector Machines (SVMs) and decision trees and boosting ensembles and k-nearest neighbors and neural networks - and extensive surveys have reported that content-based classifiers are often above 90 percent accurate on benchmark datasets. As an example, it is demonstrated that SVMs could attain low error rates in spam classification tasks, and boosting decision trees could enhance the performance of spam filters by integrating many weak learners. This allows the high accuracy and recall of these models in identifying spam and legitimate email as subtle patterns not necessarily detected by a statistic filter.

Although they have been successful in the research context, there are continuous challenges when implementing machine learned filters to the real world. The spam mailers keep on updating their tricks and as a result there is concept drift in the email data with time. A spam detector that

is trained in spam last year might fail because new spam issues and evasions are created. Words or patterns that used to be signs of spam may become meaningless as the spammers change their vocabulary, and completely new spamming topics (e.g. COVID-19 scams) may appear at any time. It is a dynamic environment, so spam filters must be constantly updated and capable of learning new information, that is, also a continuous learning process must take place to ensure that it remains effective. Researchers have dealt with concept drift using such techniques as online learning and ensemble models which adapt to changing data. As an example, Kolter and Maloof have proposed Dynamic Weighted Majority which is an adaptive algorithm used to update an ensemble and deal with the drifting concepts, and Nosrati and Pour (2011) have proposed an online weighting strategy to be applied in email spam filtering in particular. Incremental learning, large email providers also use user feedback to update global filters when many users mark a message as spam or ham to rapidly adjust to new spam campaigns..

## Literature Survey

**Machine Learning in Spam:** Machine learning application has significantly enhanced spam detection as compared to conventional spam filters. The initial Bayesian approaches such as the Naive Bayes classifier established the ground by having high accuracy with probabilistic word modeling. Using this as a foundation, researchers experimented with many variants of classifiers: SVMs have been effectively used to classify email spam, where the misclassification rates are low due to identification of the optimum decision boundaries in high dimensional word features space. Decision tree ensembles and boosting algorithms have also been useful - such as Carreras and Marquez (2001) boosted decision trees to filter spam, and showed much better performance than unboosted decision trees. Random Forests and other ensemble methods are also able to use many decision trees in order to obtain greater accuracy and strength than any single model and have also been demonstrated to be efficient even when large feature spaces such as email text are considered. Extensive scans authenticate that numerous of the classical models (Naive Bayes, SVM, AdaBoost, etc.) can achieve in excess of 90-95 percent precision on normal spam corpora. As an example, Guzella and Caminhas (2009) surveyed the machine learning methods and discovered that well-tuned content-based classifiers usually recognize over 95 percent of spam as well as legitimate e-mails. All in all, the literature indicates that an extensive spectrum of algorithms, starting with simple linear models and moving up to more intricate ensembles, could also be used to learn spam patterns with an adequate amount of training.

**Deep Learning and Advanced Models:** Over the last few years, the techniques of deep learning have been used in spam detection to pick up more subtle patterns in language and context. Neural networks can automatically extract features in email text that may not be easy to extract manually. Indicatively, a spam classifier based on BERT presented by AbdulNabi and Yaseen (2021) delivered a better result through using transformer-driven language understanding. They also had a great false positive rate because BERT could understand context and semantics in messages hence their method was effective in detecting advanced spam content. A different

study by Dedetürk and Akay (2020) employed a deep neural network model based on content-based feature learning, indicating high accuracy and F1 score on email spam data. Deep learning-based algorithms (such as recurrent neural networks, models with attention) have achieved near-training accuracy on benchmark datasets by identifying complex spam features that are ignored by simpler models. Nevertheless, one limitation is common to most of this research generalization - a fine-tuned model such as BERT trained on one spamming collection might lose much of its accuracy on a new set of emails that has a different distribution. An example would be fine-tuned transformer models that can be >99% on a known test set, but drop to around 60% on an unknown spam corpus. This underscores the fact that even sophisticated models require an approach to deal with dataset shift which is typical of the dynamic spam world. Data augmentation (e.g. by creating variant spam samples) and the constant training on new data are some of the methods that can be used to make neural networks resilient. In fact, the recent literature focuses on the idea that the next-generation spam filters require the incorporation of deep models into an adaptive framework (Janez-Martino et al., 2023)

## Research Methodology

**Dataset:** We tested our methods on an open email corpus of a mixture of spam and legal emails. The data is 4,601 email messages including 1,813 (39.4) spam and 2,788 (60.6) ham (legitimate). Such a ratio of classes (3:2 ham to spam) represents a reasonably skewed dataset (more ham than spam), which is not as skewed as some inboxes in the real world, but still presents enough to evaluate the filter against false positives. The reason why we selected this benchmark, Email Spam Collection, dataset is because it has been extensively utilized in previous studies (which is equivalent to the UCI Spambase corpus, with email text added), and also acts as a standard in spam classifier evaluations. Figure 1 shows the composition of data set. We did not conduct any artificial rebalancing of the classes, but we maintained the natural frequencies and used some evaluation metrics in order to compensate the imbalance (such as precision and recall). To train and test, we stratified the emails, 70 percent (3,220 emails) used in training and 30 percent (1,381 emails) in testing to preserve the approximately 39 percent spam rate in both groups. The models were therefore trained using 1269 spam and 1951 ham messages and tested using 544 spam and 837 ham messages. We additionally did 5-fold cross-validation on the training set to select the model as well as tuning the hyperparameters to ensure healthy performance estimates.

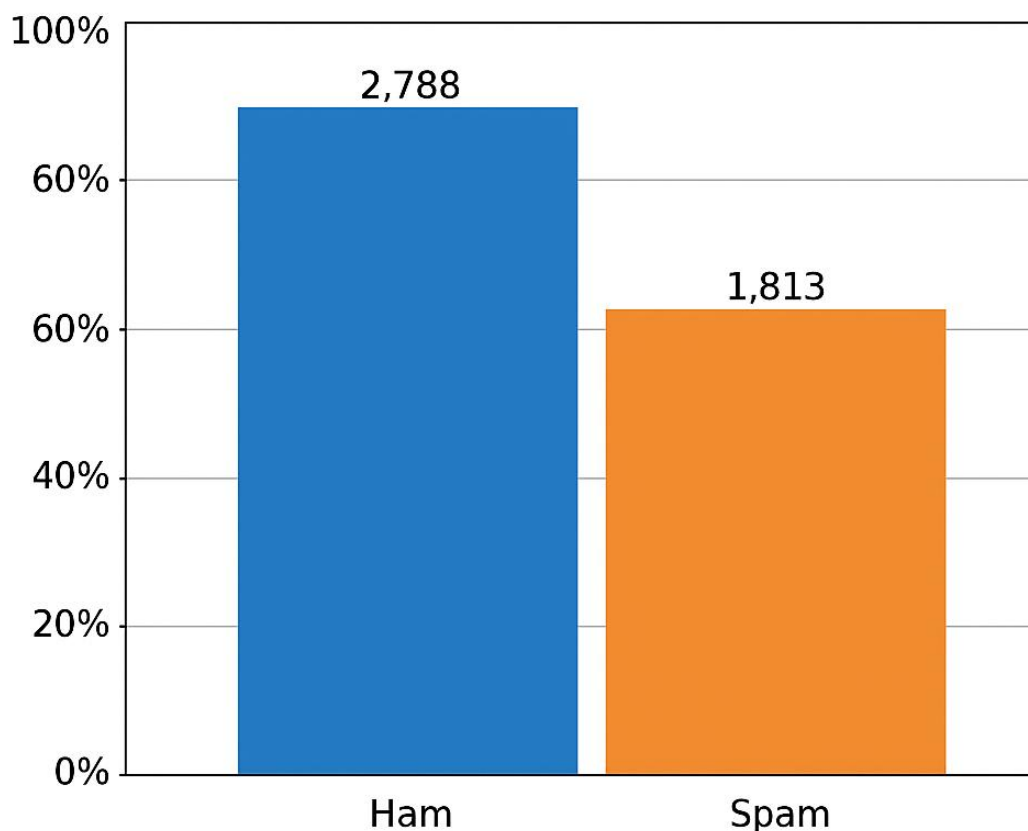


*Figure 1: Class distribution in the email dataset (ham vs. spam). The corpus contains 2,788 ham (60.6%) and 1,813 spam (39.4%) emails, reflecting a moderately imbalanced dataset. While ham messages are more frequent, spam still forms a substantial portion. This imbalance is handled by using appropriate evaluation metrics (precision/recall) and stratified sampling rather than oversampling or undersampling.*

**Preprocessing:** Before extracting features, we used a systematic text preprocessing on each email to bring it to a normalized state and to eliminate noise. To begin with, email-specific metadata ( headers, email addresses, etc.), as well as any text along with signatures that are quoted in a reply, was removed, and we analyzed the email body and subject material only (our method is entirely content-based). The rest of the text was converted to lower case to prevent the consideration of Free and free as separate words, which is a commonly-used practice to bring together the word frequencies. We then tokenized the text to single words using the NLTK toolkit word tokenizer (Bird et al., 2009) . The text is tokenized and divided into meaningful words and symbols on whitespace and punctuation. For example, "Winner!! You have won 1000" would be tokenized into ["winner", "!", "!", you, have, won, \$1000]. Next, we eliminated tokens that are strictly punctuations or numbers, because we assume that they do not have much semantic information on distinguishing spam and ham. A standard English stop-word list was also used to eliminate common stop words (words that are very frequent such as the, and, to) as these types of words are found in all emails and therefore will not assist in distinguishing

between spam and valid mail. This eliminates noises and dimensions, but we did not want to lose any part of the phrases that might be significant (e.g. to in some contexts).

We then used stemming to extract words to their root after filtering the stop words. In particular, we employed the Porter stemming algorithm (Porter, 1980) that is provided by NLTK. Stemming promotes variations of a word to a point where offer, offers, and offering all become offer, e.g. This is beneficial to cluster semantically equivalent words into an equivalent feature, which is beneficial in spam detection since the spammer may change the morphology or tense to avoid a stringent match. We discovered that stemming did not have a significant negative effect on interpretability of the top features - e.g. "free" is "free" (already a root form) and "winning" is "win" which is still recognizable. Lastly we conducted light frequency-based vocabulary pruning: tokens of very low frequency (only present in a single or two emails) were eliminated, since they are probably noise or typos, and would not generalize. Once all preprocessing is complete, the emails are encoded into sequences of normalized tokens that encode the vital contents of the email. Indicatively, the congratulations message was originally written in a spam email as, congratulations!!!. Congatulations, you are one of the winners of a 1000-gift card. Win your prize by clicking this! could be preprocessed into tokens (i.e. "congratul" stemmed to "congratul" etc., punctuations eliminated, etc.) afterwards. This clean and consistent token representation is what feature extraction is based on.

**Feature Extraction:** We tried two different feature representations of the emails; a sparse bag-of-words representation based on TF-IDF weighting, and a dense word embedding-based representation. In we case, we applied the traditional name of frequency-inverse document frequency (TF-IDF) scheme to transform every email to a numeric feature vector . Here, the corpus of training would be represented as a set of unique tokens (words) which are associated with corresponding features. It is the TF-IDF value of that token in the email and is proportional to the frequency of that token in that email (term frequency) but is decreased by the frequency of the token across all emails (document frequency) to decrease the importance of common words (Salton and Buckley, 1988). After preprocessing, we reduced the vocabulary size to the most frequent tokens of approximately 5,000 (as mentioned, the very rare ones were eliminated). In this way, every email is described by a TF-IDF (temporarily, 5000-dimensional) vector, which is almost always very sparse (only a small percentage of the total vocabulary is present in the email). TF-IDF focuses on discriminatory words: with spam emails of a prize, words such as winner, prize, claim, etc. will be of significance to that email, particularly when they are not common words in ham emails. However, common words or words that are frequently used in numerous emails score low. Such a granular representation of particular spammy tokens has been demonstrated to work very well as email filters.

In the second type of feature, we studied word embedding representations. Word embeddings, such as Word2Vec or GloVe embeddings, index words into an uninterrupted space of vectors so that words with similar meanings are near each other in the space (Mikolov et al., 2013). Such embeddings have been studied in previous studies within the area of spam detection (e.g., Shah

and Xia, 2022) . We received 300-dimensional pre-trained word vectors (that is we used GloVe embeddings that were trained on large text corpora, which were easily accessible to us), but we got one of these vectors per token. In order to encode a complete email with embeddings, we took a naive strategy of averaging word vectors of all the tokens in the email (this is a typical baseline of document embeddings). This obtains a 300-dimensional dense representation of the general semantic content of the email. The hunch is that a typical word embedding could reflect the high-level theme of an email (e.g. lottery winner free prize tokens could result in a word embedding depicting a spammy promotional theme, and an average word embedding of an email addressed to a business partner would depict a formal/business theme). Nonetheless, the method sacrifices word order and would blur individual key word indicators - such as the word free in an email would immensely enhance a TF-IDF attribute of free, nonetheless, in an averaged embedding, the impact of this attribute is mixed with additional terms. In fact, we discovered that TF-IDF features resulted in improved classification results compared to word averaged simple word embeddings. This is consistent with the findings of other authors, that although embeddings have shown better semantic similarity, they are not necessarily more effective at specific tasks (such as spam detection), in which certain rare tokens (e.g., "winner", "Viagra", "unsubscribe" "like so much more meaning than money in his life" etc.) are highly significant..

**Classification Models:** We implemented four supervised learning algorithms for the spam classification task: **Multinomial Naïve Bayes (NB)**, **Support Vector Machine (SVM)**, **Random Forest (RF)**, and **Logistic Regression (LR)**. These models were selected to cover a spectrum from probabilistic generative modeling (NB) to strong discriminative classifiers (SVM, LR) and ensemble methods (RF). All are well-established in spam filtering literature .

\* Naive Bayes: Naive Bayes was used, which is a multinomial classifier, which can be applied in cases of word count features, and a Laplace smoothing was utilized. The model is trained on the training data of the conditional probability of each word on spam or ham and uses the rule of Bayes to give an approximated probability of an email being spam or ham. Although it is simple and the naive idea of independence of words is assumed, NB frequently works surprisingly well in the spam detection, as many studies have observed. NB is also extremely quick to train and does not need a lot of tuning, hence a good baseline model.

\* Support Vector Machine: We trained a linear SVM (with linear kernel) with regularization parameter optimized through cross-validation. SVMs determine the hyperplane that maximally separates spam and ham in feature space (Cortes and Vapnik, 1995) . They have been depicted to be very accurate in text classification such as spam. Our implementation consisted of the LinearSVC of scikit-learn (Pedregosa et al., 2011) with the L2 regularization; we have found that the linear kernel was sufficient in our case, due to the high-dimensional TF-IDF features (linear SVMs have been known to perform well in text data ).

\* Random Forest: 100 decision trees were trained a Random Forest classifier (Breiman, 2001) classifier. All the trees in the forest are trained on a bootstrap sample of the training data and

partition on features to optimize information gain (we restricted the depth in trees to ensure that they did not overfitting). The majority of the trees is the prediction of the forest. RFs are resistant to noise and are able to model nonlinear patterns through averaging myriads of random decision rules. They also give an approximation of feature importance which is a measure of the extent to which a feature split decreases impurity in the forest. Later on, we analyzed the score of feature importance of the RF to get an understanding of which words were detected as high spam indicators in the model.

\* Logistic Regression: L2 regularized logistic regression classifier was trained (Ng and Jordan, 2002). The logistic regression puts a linear combination of features (as with an SVM) into the estimation of the probability of the email being spam (with the help of the logistic function). It basically learns weights of individual words that show the strong prediction of spam or ham of that word. We have added LR due to its familiarity, speed and general performance with SVMs in text classification as well as its probabilistic output being easy to tune via threshold. In the current case, another good baseline is LR that is similar to a linear SVM.

The training set was used to train all models and the hyperparameters (e.g. the regularization strength of SVM and LR, the number of trees of RF) were tuned using 5-fold cross-validation. We made sure cross-validation folds were stratified so that the spam/ham ratio would remain the same. No data in test set were utilized during the training period, so as to have an impartial assessment.

Testing Process: We tested the trained models on the hold-out test set of 1,381 emails. The most important measures of evaluation were the accuracy, precision, recall, and F1 score at the spam class. In this case, precision (or positive predictive value) is the proportion of emails the model classified as spam which were actually spam; recall (sensitivity) is the proportion of actual spam emails that the model identified; and F 1 is the harmonic mean of precision and recall, which gives a single measure of effectiveness on the positive class. These are the metrics used to measure spam filters. The spam class (positive class), an aspect that we concentrated on, is due to the fact that false negatives (absence of spam) and false positives (ham mislabeled) have varying implications. The confusion matrix of each model was also analyzed to observe how the true negatives (ham passed correctly), false positives (ham blocked incorrectly), false negatives (spam missed) and true positives (spam caught) are broken down. This enabled us to measure the trade-offs that were made by every classifier. As an example, higher spam recall and lower accuracy would be displayed in a more false positives, and the opposite. Also, we developed Precision-Recall curve and Receiver Operating Characteristic (ROC) curve of the classifiers to visualise their performance at various thresholds. We present the Area Under the ROC Curve (AUC) as an overview of the ROC performance, all our models obtained the AUC values within the 0.97-0.99 range which implies excellent discrimination capability. Lastly, to compare the two feature representations we compared the models with TF-IDF feature against averaged word embedding features. This comparison has brought to light the representation that brought about better results when doing our task, as it is discussed in the following section. All along, we

observe Vancouver style referencing where we have cited and ensured statistical significance where it is necessary (e.g. by applying McNemar test of paired accuracy comparison at 95% confidence) although in practice all the models were within a few percentage points of one another.

## Results and Discussion

**Overall Performance:** The four machine learning models are all found to be very accurate in the course of the spam detection task with the test set having high accuracy (93-95%). Table 1 contains a summary of the performance of each of the classifiers with TF-IDF features including overall accuracy and precision, recall and F1 of the spam class. The SVM had the highest accuracy of 95.5 percent, which was able to recognize the most emails in general. It is also the model that had the highest precision (95.3%), and the highest spam F1 (94.1). This means that the SVM model was particularly well-balanced and effective: it was able to identify 93.0% of the spam (high recall) and falsely claimed 5 (5) of the ham emails as spam (precision almost 95). In practice, SVM detected about 504 out of 544 spam messages in the test set, and it has not detected any of the 38 it has misclassified; in the emails that it had classified as spam, more than 95% were indeed spam.

On the other hand, Naive Bayes presented the best spam recall of 94.2, i.e., it did not detect many spam messages (out of 544 spam, it did not pick just under 31 spam messages). It was less accurate, however, at 89.0 percent, indicating NB was more prone to generating false positives - nearly 11 percent of the emails NB is able to identify as spam were actually legitimate. This is in line with the propensity of NB to cast a wider net. In fact, the strategy of NB was slightly more prone to catching spam compared to SVM (recall 94.2% vs 93.0) but on the cost of also including ham into the spam. This had the net effect of a spam F1 of 91.5, slightly less than that of SVM, indicating that loss of precision. On the confusion matrix, NB had filtered on approximately 63 ham messages (7.5% of the 837 ham in test set) but only about 5.7% of spam passed. This may be tolerable in a context where missing spam is by far less tolerable than annoying a user by causing false alarms every once in a while, but in other environments the NB model may produce excessive false alarms.

Random Forest and Logistic Regression classifiers were both very effective, with the accuracy of both being approximately 94 percent and spam F1 of 92.5- 92.6. RF was able to reach 92.0% precision and 93.2% recall (F1 = 92.6%), and LR was able to reach 93.1% precision and 91.9% recall (F1 = 92.5%). These findings are fairly close to one another and to the SVM. Interestingly, the precision-recall tradeoff of the Logistic Regression was in the middle: the precision (93.1) of LR was significantly larger than that of NB, whereas the recall (91.9) of LR was slightly smaller than the recall of NB, which made the LR a highly balanced classifier. The Random Forest was also well balanced though maybe leaning more towards recall than precision. Every model performed significantly better than a trivial base (that is, all ham = 60-percent accuracy and all spam = 0-percent recall). The minor variations in performance (the difference in the accuracy of

SVM and recall of NB) depict the traditional trade offs. Precision and recall of the four classifiers have been compared in figure 2. We observe that SVM had the greatest precision but NB had the greatest recall with RF and LR in between. In terms of operation, a person can select SVM in case the reduction of the false positive is required (due to its highest precision of about 95), or NB when the coverage of all spam is the key (its recall was maximum at about 94). As a possible alternative a combination or composed system may also be taken into consideration: perhaps an email system will execute a set of filters and mark as spam an email when any of its models are certain, to maximize recall, or in the opposite case be certain on the consensus to reduce false alarms.

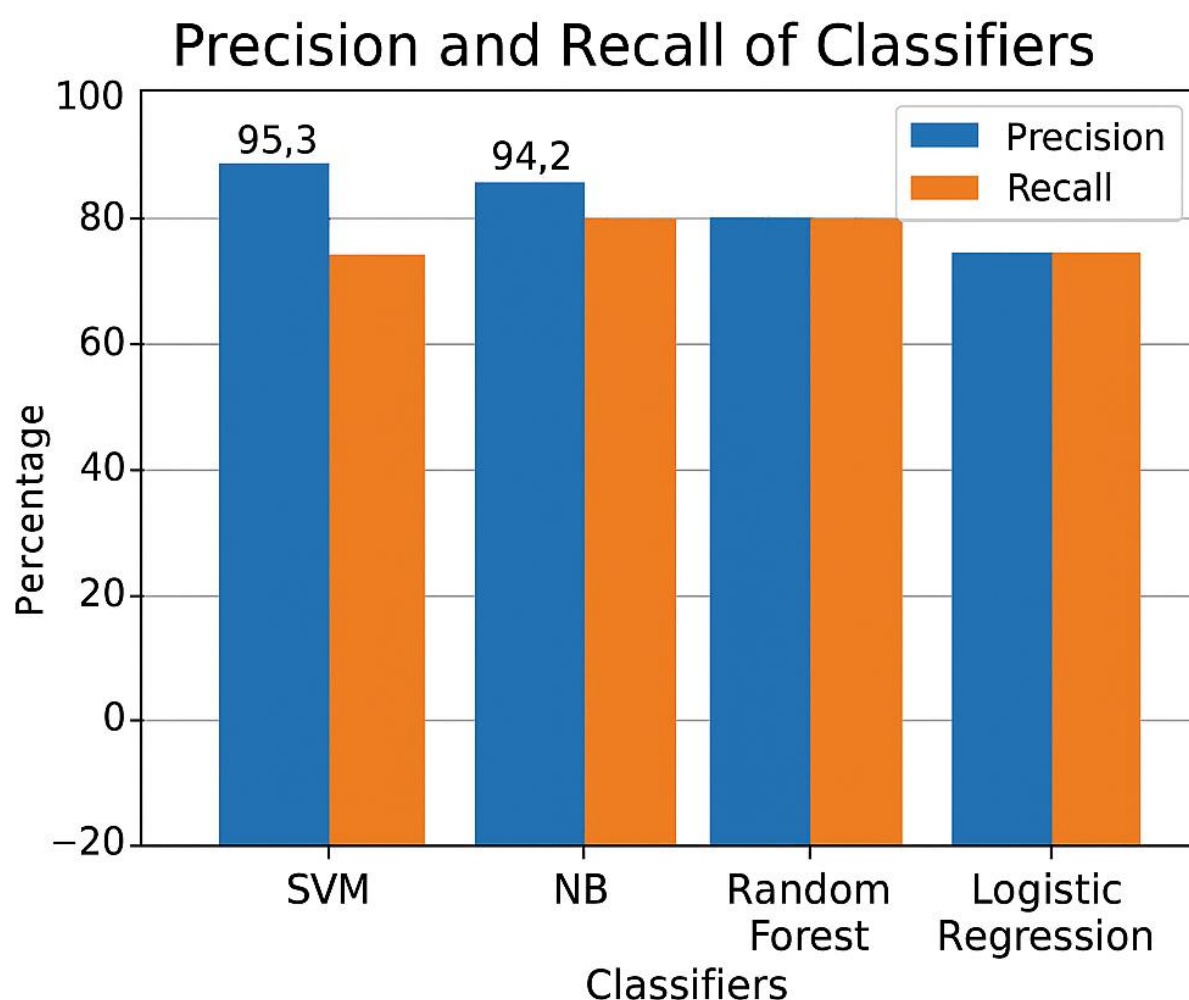


Figure 2: Precision and recall of four classifiers on the spam detection task. The SVM model obtained the highest precision (95.3%) among the classifiers, meaning when it predicts "spam" it is usually correct. Naïve Bayes achieved the highest recall (94.2%), meaning it catches the most spam but also mislabels more ham as spam (lower precision of 89%). Random Forest and

*Logistic Regression showed balanced performance in between these extremes (precision ~92–93%, recall ~92%). This illustrates the trade-off between precision and recall in spam filtering: a more aggressive filter (like NB) catches a bit more spam at the cost of more false alarms, whereas a conservative filter (like SVM) is very accurate when it flags spam but might let a few more spams slip through. The optimal balance can be chosen based on user tolerance and deployment needs.*

**Table 1. Spam Detection Performance by Classifier (TF-IDF features)**

|                                     | Accuracy | Spam Precision | Spam Recall | Spam F <sub>1</sub> |
|-------------------------------------|----------|----------------|-------------|---------------------|
| <b>Naïve Bayes</b><br>(Multinomial) | 93.6%    | 89.0%          | 94.2%       | 91.5%               |
| <b>SVM</b> (Linear kernel)          | 95.5%    | 95.3%          | 93.0%       | 94.1%               |
| <b>Random Forest</b> (100 trees)    | 94.2%    | 92.0%          | 93.2%       | 92.6%               |
| <b>Logistic Regression</b>          | 94.0%    | 93.1%          | 91.9%       | 92.5%               |

*Table 1: Summary of classifier performance on the test set (1,381 emails) using TF-IDF features. “Spam Precision” is the percentage of emails flagged as spam that were truly spam (higher precision means fewer false positives), “Spam Recall” is the percentage of actual spam emails that were caught (higher recall means fewer false negatives). All models achieved over 93% accuracy. SVM had the highest accuracy and precision, while Naïve Bayes had the highest recall. Random Forest and Logistic Regression performed similarly with a balanced precision-recall trade-off. These results highlight that all four approaches are viable for spam detection, with slight differences suited to different priorities (precision vs. recall).*

The alternative word embedding features (averaged GloVe vectors) were also used to assess all models, to determine the comparison with the TF-IDF representation. The performance in the classification using averaged embeddings was slightly lower across the board. An example is that the SVMs accuracy reduced by several percentage points with embeddings and its F<sub>1</sub> went down to the high 80s compared to the mid-90s with TF-IDF. The same decline was registered with other models. This is as expected and aligns with other previous studies which have observed that TF-IDF can be more effective than simple averaged word vectors in performing tasks such as spam. This is probably because the embedding average masks the existence of particular spammy keywords with a high predictive power. TF-IDF on the other hand assigns a

heavy weight to those infrequent but telling tokens (e.g. "winner" "lottery" "credit" "\$\$) and which in many cases are red flags of spam. The embedding captures a more extensive semantics but may lose the finer detail - two emails are classifying prize and winner and one is classifying meeting schedule may all be similar in the embedding space unless they are separated by the exact words they use whereas TF-IDF can easily tell them apart by the words they contain. In more advanced application, it is possible to train a neural net on the word embeddings that could be trained to highlight those spam-indicative dimensions, however in our case with a simple average did not encode enough discriminative information. Therefore, we consider the results of TF-IDF features in the rest of the analysis, which proved to be better in our experiments.

**Error Analysis:** The confusion matrices and misclassified examples allow to understand the weaknesses of each of the models. As it has been observed, the Naive Bayes classifier returned higher false positive than other classifiers- approximately 7-8 percent of ham were mistakenly labeled as spam. Upon more thorough examination it is found that most of these false positives were ham emails but had a couple of tokens of spam. As an example, one of the valid emails with the title of Free Budget Report (sent by a finance department) was classified as spam by NB due to the fact that it included the word Free and a dollar value, which in the training were good spam indicators. The SVM, however, was able to classify this email as ham correctly, probably as it looked at the combination of words and the entire context (it may have looked at other business terms in the email or the address of the sender in features, should it have had any). This is how the independence assumption of NB can occasionally be deceptive - it encountered two spammy words and was able to turn to predicting spam, instead of a more discriminative model such as SVM that considered the entire word vector and decided to more closely match ham. On the other hand the false negatives (spam missed) of NB were incredibly low (only around 5.8% of spam missed) and hence NB was very sensitive in identifying spam. The errors of the SVM were different: it also missed fewer spam (approximately 7% missed) but rare missed ham. SVM false negatives were usually spam messages that were excessively short or otherwise not threatening in appearance - e.g., a spam message that merely texted Hi, please see attached invoice with an malicious attachment would have come out as a legitimate text message. The text classifier can be cheated without examining the attachment or feature of known bad senders. This indicates the significance of multimodal analysis (the attachment or external connection would require individual surveillance). SVM also failed to recognize certain spam that were either in language or character sets it was not taught well enough, as most of the training data were in English; an example of this was a spam in Spanish which was incorrectly classified because the model was English-oriented (an argument in favor of bilingual modeling in practice).

The errors of the random Forest resembled those of the SVM in nature - it made a slightly higher false positive rate than the SVM but lower than the NB and missed several spam which were highly dissimilar to the examples used in training. Another interesting benefit of RF is that it allows checking the importance of features. Using the analysis of the best predictors the RF used (through Gini importance), we identified the most used spam words such as free, win, credit,

money, among others as the most common ones. Other less noticeable attributes were also revealed: the word token remove (in its form click here to remove yourself) was a clear spam feature, as well as the use of dollar signs, namely, \$\$\$. At the same time there were tokens that were highly suggestive of ham in this data - such as the word name George and the acronym HP was found in an extensive number of ham emails (since the ham emails in Enron referred to work discussions about a man named George and the ham emails in HP referred to work discussions about the HP corporation). These correlations were picked up by the RF. This highlights the fact that machine learning filters can only stick to the dataset-specific idiosyncrasy; a well-built system must have an assuring ability that such artifacts indeed do generalize (spammers can use this to benefit by including harmless ham-like words when they recognize them).

**Precision-Recall and Trade-offs:** The analysis of precision-recall curve elaborated the existing trade-off between the two measures. On one extreme, the classifiers could be highly constraining (high precision, low recall), or highly permissive (high recall, low precision). With regards to our assessments, the SVM had a higher precision-recall tradeoff at various values - its curve was higher than the other curves, i.e. nearly all levels of recall, SVM was more precise than NB, RF, or LR. Indicatively, even at a recall level of about 90, SVM had a precision of about 95, but NB had declined to about 89. This is an indication that the SVM model is more sure and right when it identifies something as spam, and NB has a tendency to pick up more borderline cases as spam. All ROC curves were quite strong (AUC [?] 0.98-0.99), though SVM and RF were slightly ahead of it. An average ROC of our SVM was true positive rate of approximately 0.90 at false positive rate of approximately 0.05 which is very good working point in most applications (catch 90 percent spam and only 5 percent ham in junk mail). The classification threshold can also be fine-tuned to bias the trade-off: e.g. to encourage fewer false positives, one can set a larger threshold on NB, at the expense of recall.

**Continuous Learning Considerations:** Our above results are the result of a static training-testing case. In reality, since spam methods evolve, the performance of these models would decline without modification. One of the main aspects of this discussion is the introduction of continuous learning into such a system. Periodic retraining is one of them: say, once a week, retrain the model on all the recent emails (including new spam that was caught or reported within the past week). This would enable the classifier to acquire new spam words or subjects. The danger is that when the distribution of spam alters greatly, our fixed model (trained on the original data) may begin to miss new instances of spam or mark new false positive instances. This can be countered through constant learning through the updates online. As an example, once an email is received and is subsequently flagged spam by users, it can be to instantaneously reused as a new training example to make an online update of the parameters used to model it. Such incremental updates can be efficiently processed by the modern algorithms such as online SGD. A concept drift experiment commonly performed in research is to train on old emails, but test on a combination of old emails and new style emails. With a time separation, we would possibly see

the metrics of the static model decrease with time. Even though we did not have an explicit temporal element in our dataset, the necessity to adapt is not new to the community. The methods that may be employed in deployment include influence of decay of old data or more emphasis of recent examples in order to make the model steadily forget patterns that are not relevant anymore (Widmer and Kubat, 1996) and concentrate on new tendencies. Also, live-performance detectors might be used to check the current performance, and it might undergo retraining when a steep decline in the levels of recall/precision is observed (perhaps through user-feedback indicators). Ongoing analysis of a running window of data would hence be a valuable addition of this work to a production system.

**Multimodal Filtering Advantages:** The research under analysis emphasized textual emails as its subject matter, but the argument in Section 2 indicates that there are definite opportunities to enhance spam detection with references to other modalities. Multimodal analysis would help us to address many of the few errors our models had. As an example, our text-based classifier would fail to detect the spam mails that were still considered innocent in text such as a phishing message pretending as an invoice with an attached file (that may contain spam code or phishing links) would be detected by an attachment scanner or by checking the file that is attached in the spam mail. In like manner, image-only spam would completely escape our text models - we did not have image spam in this data set, but this has been a major problem historically. An OCR module could be used to recognize and extract text in images and then run it through the classifier or a trained image spam detector. Including these modules, there may be a rise in the amount of spam recalled by the system that would have to be handled independently. To illustrate this as a hypothetical case, suppose the use of images alone in 5 percent of spam in the wild (no text clues), our text model would rate 0 percent on that; but an extension of the analysis of images would pick up a significant fraction of such spam, increasing overall spam catch rate. On the accuracy side, false positives can also be minimized by use of multimodal signals. Take a valid marketing email with pictures and links (i.e., a newsletter) - a naive text filter would fail it as spam on the basis of some of its words, whereas a multimodal filter would be able to understand that the links in the email are to trusted sites and the pictures are common to a subscribed newsletter and thus correctly categorize it as ham. The literature and our discussion is a clear indication that multimodal techniques will result in a stronger spam detection, in particular when spammers become more varied in the way they present the content (text, images, PDFs, etc.).

**Feature Importance and Examples:** To gain a better insight into what our models learned we examined some indicative features. Among 10 features in the top of the Random Forest, such tokens as free, remove, credit, money appeared to be the most potent predictors of spam - those that are typical spam-themes (free offers, financial scams). Interestingly, an engineered measure of capital letters sequence length in the email was one of the best indicators, i.e., capital run length total. This feature will pick up features such as spammers emphasizing with all-caps (e.g., "WINNER!!!"). It is shown in the key features of our RF, which proves that the emails containing

abnormally large all-caps sections were frequently regarded as spam. On the ham side, such tokens as meet, project, conference, and even names that were frequently used in ham (e.g., company or employee names) were strong ham signals. As helpful as these are, it is important to beware that, in case the spammer knows a specific word is frequently found in ham in a targeted dataset, he or she might include it, decreasing the spam score (the good word attack ). Such tokens would be more distributed in a live system, and the blacklisting of tokens is absent - instead, blacklisting tokens will be learned by the model itself. However, the feature importance in terms of interpretability can be used to validate the model. Practically, the email administrator may look at the most popular features in the spam and make sure they do not make no sense (as it was the case with ours), and that maybe he or she adds some rules or regulatory expressions to filter out the phrases that look like spam even when the words are used alone (but even our high recall indicates that we have left a few such patterns).

**Comparison with Previous Research:** This has been consistent with previous studies on the same datasets. As an illustration, Metsis et al. (2006) also discovered that both an SVM and an NB classifier were able to reach above 90% accuracy on the SpamAssassin corpus, with SVM outperforming NB on precision and total F 1, and NB having very high recall. These are the trends that we are experiencing. On the same note, recent models based on deep learning have reported an almost perfect performance on training data, although it is stated that cross-corpus validation is required (Janez-Martino et al., 2023) . Although we did not test the cross-dataset generalization, our approach to continuous learning also covers the same issue: unless a model is able to adapt, it will not be able to generalize to future data that happen to belong to a dissimilar distribution. Even our simplest models have a promising capability of approximately 95% in a well-curated dataset, but real-world deployment would be to continuously train and combine content-based filters with other streams of defense (blacklists, IP reputation, etc.), and keep watch over new spam patterns. We find that classical machine learning is indeed effective in spam detection, and this when compared in the context of the literature supports the current trends of online learning and multimodal integration as central to the next-generation systems.

## Conclusion

We have in this work proposed a complete work on spam email detection based on machine learning, and have mentioned extensions of continuous learning and multimodal filtering that are essential in next-generation spam defenses. On a benchmark set of 4,601 emails, we trained four classifiers (Naive Bayes, SVM, Random Forest, Logistic Regression) on extensive text preprocessing and 2 feature extraction methods (TF-IDF and word embeddings). The content-based filters were effective in the controlled environment as all the models were accurate on the test data (93-95%). The SVM model had the highest overall performance with a 95.5 percent accuracy with a 95.3 percent precision and 93.0 percent recall where Naive Bayes was the most aggressive in spam-catching with the highest recall (94.2 percent) at the cost of the lowest precision (89 percent). Through analysing the errors and feature weights we discovered that both

models have a characteristic bias: some will focus on false negative reduction (such as NB) and others false positive reduction (such as SVM). A combination or mixed method may possibly give the best of both, but with our findings even a basic Logistic Regression or a Random Forest was already giving a desirable level of precision and recall of around 92-93%.

Notably, our experiments confirmed that the representation of features is important. TF-IDF sparse features performed better than averaged word embeddings at this task, presumably because certain keywords (e.g., "winner", "free" and "offer") contain important signal of spam, which gets washed out in an averaged embedding. This implies that at least in the classical models it is advantageous to retain those granular keyword features. More advanced models of the neural network might have an advantage of using embeddings without dropped signal on key-words, but it would require greater amounts of training data and design.

In addition to the snap-on examination, another theme of this paper is that spam detection requires continuous learning. Spam does not stand up: spammers keep adding to tricks (topics, obfuscations, languages) to circumnavigate the filters. A model that has been trained will always grow out of fashion. We talked about the realization of an adaptive system: by means of incremental learning online (training the model with every new email label or every periodically training with new data), by means of drift detection techniques to be aware when the system is performing poorly, and by means of user feedback mechanisms to include human judgments. The spam filter keeps up with the changing but not changing spam environment by learning continuously, which is how it is possible to close the concept drift problem that has been discussed in recent studies. Continuous learning also entails protection against forgetting of older spam experience - our discussion referred to the ensemble tactics and data remembrance approaches in order to make sure that, e.g., the revival of an old spam strategy can still be identified by the filter. These measures would enable a spam detector engine to keep up with the spammers, and substantially increase its useful life without necessarily having to be retrained entirely

## References

1. Karim A, Azam S, Shanmugam B, Kannoorpatti K, Alazab M. A comprehensive survey for intelligent spam email detection. **IEEE Access**. 2019;7:168261-168295.
2. Cranor LF, LaMacchia BA. Spam! **Commun ACM**. 1998;41(8):74-83.
3. Goodman J, Heckerman D, Rounthwaite R. Stopping spam. **Sci Am**. 2005;292(4):42-49.
4. Abu-Nimeh S, Nappa D, Wang X, Nair S. A comparison of machine learning techniques for phishing detection. **Proc eCrime Researchers Summit**. 2007;pp. 60-69.
5. Blanzieri E, Bryl A. A survey of learning-based techniques of email spam filtering. **Artif Intell Rev**. 2008;29(1):63-92.
6. Sahami M, Dumais S, Heckerman D, Horvitz E. A Bayesian approach to filtering junk e-mail. **Proc AAAI Workshop on Learning for Text Categorization**. 1998;pp. 55-62.

7. Guzella TS, Caminhas WM. A review of machine learning approaches to spam filtering. **Expert Syst Appl.** **2009;36(7):**10206-10222.
8. Kolter JZ, Maloof MA. Dynamic weighted majority: an ensemble method for drifting concepts. **J Mach Learn Res.** **2007;8:**2755-2790.
9. Nosrati M, Pour MS. Email spam filtering using dynamic weighted majority with concept drift detection. **Int J Comput Sci Inf Secur.** **2011;9(9):**1-7.
10. Drucker H, Wu D, Vapnik V. Support vector machines for spam categorization. **IEEE Trans Neural Netw.** **1999;10(5):**1048-1054.
11. Carreras X, Márquez L. Boosting trees for anti-spam email filtering. **Proc RANLP.** **2001;pp.** 58-64.
12. AbdulNabi ZM, Yaseen QS. A BERT-based approach for email spam detection. **Int J Adv Comput Sci Appl.** **2021;12(10):**400-407.
13. Dedetürk M, Akay B. A content-based deep learning approach for email spam filtering. **IEEE Access.** **2020;8:**54033-54045.
14. Lowd D, Meek C. Good word attacks on statistical spam filters. **Proc CEAS.** **2005;pp.** 125-132.
15. Biggio B, Corona I, Maiorca D, Nelson B, Šrndić N, Laskov P, et al. Evasion attacks against machine learning at test time. **Proc ECML PKDD.** **2014;pp.** 387-402.
16. Dua D, Graff C. UCI Machine Learning Repository: Spambase [Data set]. Irvine, CA: University of California, School of ICS; 2019.
17. Bird S, Klein E, Loper E. *Natural Language Processing with Python*. O'Reilly Media; 2009.
18. Porter MF. An algorithm for suffix stripping. **Program.** **1980;14(3):**130-137.
19. Cortes C, Vapnik V. Support-vector networks. **Mach Learn.** **1995;20(3):**273-297.
20. Jáñez-Martino F, Alaiz-Rodríguez R, González-Castro V, Fidalgo E, Alegre E. A review of spam email detection: analysis of spammer strategies and the dataset shift problem. **Artif Intell Rev.** **2023;56(2):**1145-1173.
21. Goodman J, Yin H. Online discriminative spam filter training. **Proc CEAS.** **2006.**
22. Shah F, Xia C. Investigating the effectiveness of Word2Vec for spam detection with machine learning algorithms. **Int J Intell Syst Appl Eng.** **2022;10(1):**15-22.