Support Vector Machines for Spam Email Classification: A Comparative Analysis on the UCI Spambase Dataset

Nick Langel, Alan Sun, Nathan Szeto

Williams College

ABSTRACT

This article investigates the application of Support Vector Machines (SVM) on the UCI Spambase dataset for spam email classification. The Spambase dataset contains features extracted from email messages, such as word frequencies, along with labels indicating whether an email is spam or legitimate. We perform a prediction accuracy comparison of SVM against other classification techniques, namely KNN, logistic regression, and random forest. We find misclassification and false positive rates of (9.97%, 5.39%) for KNN, (7.95%, 7.50%) for logistic regression, (5.08%, 2.96%) for random forest, and (6.32%, 3.70%) for SVM. Overall, SVM performs second best only to random forest in both metrics. These results highlight the effectiveness of SVM as a robust classifier for spam email detection, showcasing its potential in improving email filtering systems and mitigating the impact of unsolicited emails.

Keywords: Support Vector Machine, Classification, Machine Learning, Spam Email

1. INTRODUCTION

The proliferation of spam emails has become a pervasive problem in today's digital world, necessitating effective email filtering systems. The UCI Spambase dataset provides a valuable resource for studying the application of Support Vector Machines (SVM) in spam email classification. This dataset comprises various attributes extracted from email messages, including word frequencies, character frequencies, and other characteristics, along with labels indicating whether the email is categorized as spam or legitimate.

SVM, a powerful machine learning algorithm, offers an effective approach to distinguish between spam and legitimate emails. By constructing optimal hyperplanes in a high-dimensional feature space, SVM aims to maximize the separation between the two classes, enabling accurate classification. In this study, we explore the efficacy of SVM in accurately identifying spam emails using the UCI Spambase dataset.

To assess the performance of SVM, we follow standard procedures, including data preprocessing steps such as removing outliers, handling missing values, and normalizing features. SVM models are trained on the preprocessed data, and hyperparameters are tuned through k-fold cross validation.

Furthermore, we compare the performance of SVM with other popular classification algorithms, such as KNN, logistic regression, and random forest, to gain insights into its relative strengths and weaknesses. Through extensive experimentation and analysis, we aim to provide a comprehensive understanding of SVM's suitability for the UCI Spambase dataset.

The successful identification and filtering of spam emails have significant implications for mitigating the negative impact of unsolicited messages, such as privacy breaches, malware dissemination, and information overload. Effective classification algorithms, such as SVM, play a vital role in building robust email filtering systems that can accurately identify spam and prevent its delivery to users' inboxes.

2. METHODS

Support Vector Machines (SVMs) have revolutionized the field of statistical learning and have become a cornerstone in modern machine learning techniques. Originally introduced by Vapnik and his colleagues in the 1990s, SVMs offer an elegant solution to classification and regression problems, providing both solid theoretical foundations and impressive practical performance.[1] Detailed below are some of the key concepts that underpin the method.

2.1 Overview of SVM

The core principle behind SVMs is the identification of an optimal hyperplane that maximally separates data points belonging to different classes. By maximizing the margin between the hyperplane and the closest data points, SVMs strive for enhanced generalization capabilities and robustness to noise. Initially developed for binary classification, SVMs have been extended to handle multi-class scenarios through the introduction of efficient strategies such as the One-vs-One and One-vs-All approaches. The basic form of SVM can be expressed as:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \tag{1}$$

subject to:

$$\begin{aligned} y_i(\mathbf{w}\cdot\mathbf{x}_i + b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \quad \forall i \end{aligned}$$

Here, $\mathbf{w}$ represents the weight vector perpendicular to the hyperplane, b is the bias term, $C > 0$ is the regularization parameter, and ξ_i are slack variables that allow for the existence of misclassified or margin-violating data points.

The decision function can be defined as:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}\cdot\mathbf{x} + b) \tag{2}$$

The SVM algorithm seeks to find the optimal values of $\mathbf{w}$ and b by solving the constrained optimization problem using techniques such as the Sequential Minimal Optimization (SMO) algorithm.

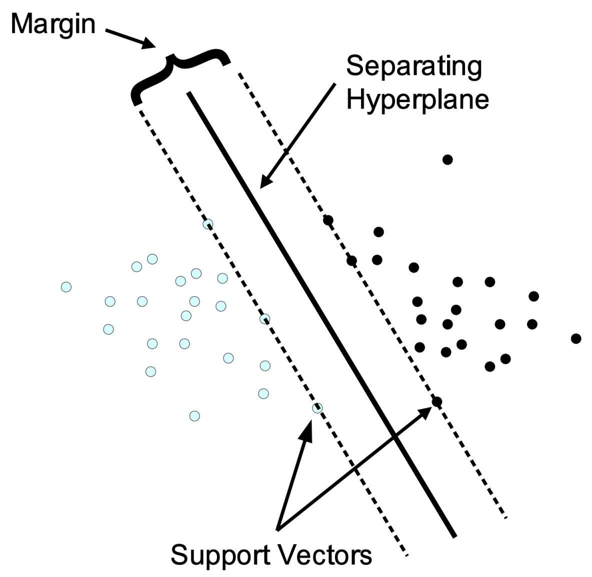

Figure 1: SVM Classification via maximum margin.

2.2 SVM Kernels

A key breakthrough in SVM is the utilization of the kernel trick, which allows for nonlinear separation in feature spaces of higher dimensions. This technique eliminates the need to explicitly transform the data into higher-dimensional spaces, overcoming computational challenges while preserving accuracy. Various kernel functions, such as the linear, polynomial, and radial basis function (RBF) kernels, provide flexibility in capturing complex relationships and handling nonlinearity in the data.

The two key hyperparameters in SVM are the cost parameter (C) and the kernel-specific parameters. The cost parameter controls the trade-off between maximizing the margin and minimizing the training errors. A smaller C value allows for a larger margin but may result in more misclassifications, while a larger C value emphasizes accurate classification at the expense of a narrower margin.

The Radial Basis Function (RBF) kernel is one of the most commonly used kernels in SVMs. It is especially useful when dealing with non-linearly separable data.

The RBF kernel is defined as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right) \tag{3}$$

Here, $\mathbf{x}_i$ and $\mathbf{x}_j$ are the input feature vectors, and $\gamma > 0$ is a hyperparameter that controls the smoothness and flexibility of the decision boundaries. A smaller value of γ results in smoother decision boundaries, while a larger value allows for more complex and flexible boundaries.

The RBF kernel uses the notion of the Euclidean distance between the input feature vectors. The kernel computes the similarity between two points based on their distance in the feature space. Points that are closer together will have higher similarity values, while points that are farther apart will have lower similarity values.

In SVM, the RBF kernel enables the mapping of the input feature vectors into a higher-dimensional feature space, where the data may become linearly separable. The SVM algorithm then optimizes the hyperplane in this transformed space to maximize the margin between different classes. Kernel-specific parameters, such as the degree in polynomial kernel or the width in radial basis function (RBF) kernel, also influence the SVM model's performance. These parameters determine the shape and flexibility of the decision boundaries.

2.3 Advantages/Disadvantages of SVM

SVMs are very effective in datasets with a clear margin of separation, and because this margin is maximized, results in good generalization performance on new data. In addition, they are known to be effective in high-dimensional spaces, and especially in cases where the number of dimensions exceeds the number of samples. Furthermore, because SVMs can use different kernel functions, they are versatile and can capture many different relationships within data. Kernel functions also allow for SVMs to handle non-linear relationships in data, which is very useful for a many real-world applications.

SVMs are not suitable for very large datasets due to a computationally expensive training step. Moreover, SVM is quite sensitive to parameter tuning, and results can vary significantly depending on the selection of parameters. SVM is also inherently a purely predictive model and thus lacks probabilistic interpretability.

When considering the use of SVM to deal with classification problems, it is important to keep the above advantages and disadvantages in mind to decide whether or not they are appropriate.

2.4 SVM Hyperparameter Tuning

To find the optimal values for the aforementioned hyperparameters, various techniques can be employed:

1. **Manual Grid Search**: In this approach, the hyperparameters are manually specified, and a grid of possible values is defined for each hyperparameter. The SVM model is then trained and evaluated using different combinations of hyperparameter values. The performance metrics (e.g., accuracy, F1-score) are compared across different combinations to select the best set of hyperparameters. This method can be time-consuming and requires domain knowledge to select appropriate ranges for hyperparameters.

2. **Randomized Search**: Randomized search is an alternative to grid search that offers more flexibility and efficiency. Instead of defining a specific grid of hyperparameter values, randomized search randomly samples hyperparameter values from predefined distributions. The model is trained and evaluated using these random combinations of hyperparameters. This approach is particularly useful when the search space is large and exploring all possible combinations is not feasible.

3. **Model-Based Optimization**: Model-based optimization techniques aim to find the best hyperparameters by constructing a probabilistic model of the objective function (e.g., accuracy). These techniques sequentially propose new hyperparameter configurations to evaluate based on the model's predictions. Bayesian Optimization and Gaussian Process-based methods are common model-based optimization approaches used for hyperparameter tuning.

4. **Nested Cross-Validation**: Nested Cross-Validation is employed when both hyperparameter tuning and model evaluation are required. The dataset is divided into an outer loop and an inner loop. The outer loop performs K-Fold Cross-Validation to evaluate the model's performance using different hyperparameter configurations. The inner loop performs K-Fold Cross-Validation to tune the hyperparameters by comparing different combinations. This approach provides an unbiased estimate of the model's performance and helps prevent overfitting during hyperparameter tuning.

5. **K-Fold Cross-Validation**: Cross-validation is a resampling technique used to estimate the performance of a model on unseen data. K-Fold Cross-Validation divides the training data into K equally sized folds. The SVM model is trained K times, each time using K-1 folds for training and one fold for validation. Performance metrics are averaged across the K iterations, providing a more robust estimate of the model's performance. K-Fold Cross-Validation is commonly used in combination with grid search or randomized search to assess the performance of different hyperparameter combinations.

Through careful selection and tuning of the hyperparameters, the SVM model can achieve better performance and generalization capabilities on unseen data. The choice of tuning method depends on the specific problem, available computational resources, and the desired trade-off between exploration and exploitation.

The development of efficient training algorithms, most notably the Sequential Minimal Optimization (SMO) algorithm, has significantly improved the scalability of SVMs.[2] By exploiting the sparsity of the solution, these algorithms reduce the computational burden while maintaining the model's accuracy. Consequently, SVMs can now be applied to large-scale datasets, making them valuable tools in diverse real-world applications.

2.5 SVM Software

The `e1071` R package provides a robust implementation of Support Vector Machines (SVMs) for classification and regression tasks.[3] We use this library over others for its relative computational efficiency (See Figure 1, taken from Karatzog, 2006).[4]

	`ksvm()` **(kernlab)**	`e1071()` **(e1071)**	`svmlight()` **(klaR)**	`svmpath()` **(svmpath)**
musk	1.40	1.30	4.65	13.80
Vowel	1.3	0.3	21.46	NA
DNA	22.40	23.30	116.30	NA
BreastCancer	0.47	0.36	1.32	11.55
BostonHousing	0.72	0.41	92.30	NA

Table 1: The training times for the SVM implementations on different datasets in seconds. Timings where done on an AMD Athlon 1400 Mhz computer running Linux.

The `e1071` package provides the `svm()` function to train SVM models. The SVM algorithm aims to find an optimal hyperplane that separates data points of different classes with the largest margin. The mathematical formulation behind SVMs involves solving a quadratic optimization problem.

The `svm()` function in `e1071` allows you to specify various parameters to customize the SVM model. Here are some key parameters:

- `formula`: Specifies the formula that defines the relationship between the target variable and the predictor variables.
- `data`: Specifies the training data used to fit the SVM model.
- `kernel`: Specifies the kernel function to be used. The e1071 package supports various kernel functions, such as linear, polynomial, radial basis function (RBF), and sigmoid.
- `cost`: Specifies the cost parameter C that controls the trade-off between misclassification and maximizing the margin. A higher value of C allows for more training errors but narrower margins, while a lower value of C results in fewer training errors but wider margins.

Once the SVM model is trained using the `svm()` function, you can use the `predict()` function to make predictions on new, unseen data. The `predict()` function takes the trained model and the new data as input and returns the predicted class labels or regression values.

It's important to note that SVMs can be used not only for classification but also for regression tasks. For regression, the `e1071` package provides the `svm()` function with the `type = "eps-regression"` argument, allowing you to perform epsilon-support vector regression.

In addition to the basic functionalities, the `e1071` package offers various options for advanced SVM techniques. These include feature scaling, cross-validation, parameter tuning, and support for multi-class classification using techniques such as one-vs-one and one-vs-all.

3. DATA

We use the UCI Spambase dataset which contains 4601 emails with 58 attributes. All attributes are continuous with the exception of one nominal attribute denoting whether or not a particular email is spam. There are 1813 instances of spam in the dataset, accounting for 39.4% of the data. 48 of the attributes are word frequencies, indicating the percentage of words in a given email matching a select word—examples include *mail*, *business*, and *money*. 6 attributes are frequencies of the characters *;*, *(*, *[*, *!*, *$*, and *#*. The remaining 3 attributes measure the length of sequences of consecutive capital letters—average length of uninterrupted

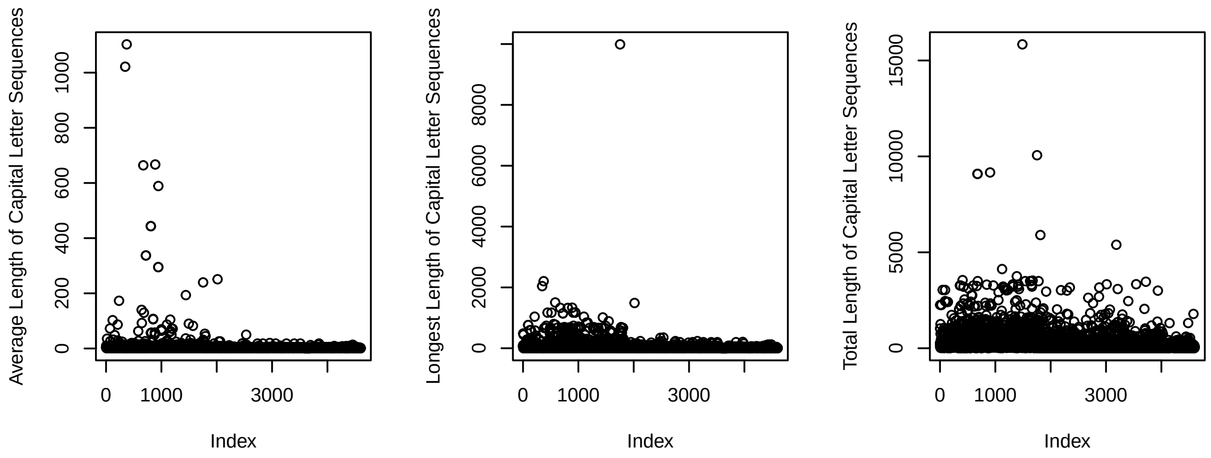

Figure 2: Capital Letter Sequence Outliers

sequences of capital letters, length of longest uninterrupted sequence of capital letters, and total number of capital letters.

On balance, the dataset is slightly imbalanced, with non-spam emails (60.6%) being more prevalent than spam emails (39.4%). There is no meaningful or problematic multicollinearity between predictors. As visualized below, there are a number of notable outliers with regard to the three capital letter sequence length variables (Figure 2). However, since there is nothing inherently erroneous about these observations upon closer inspection, we retain them in the dataset.

4. RESULTS

The data is first normalized and randomly split into training and testing sets, with 2/3 of the observations allocated to training and 1/3 allocated to testing. Each of the classification models to be described are trained and tested on the same data splits.

In addition to SVM, we examine and compare the performances of k-Nearest Neighbour Classification (KNN), logistic regression (GLM), and random forest Classification (RF) on the Spambase data. We use 10-fold cross-validation to tune the various hyperparameters associated with each of these methods. We also initially considered including both Linear Discriminant Analysis and Quadratic Discriminant Analysis, but violations in multivariate normality in the data ruled out this possibility.

We evaluate each method using misclassification error (Err) and false positive rate (FPR). FPR is a particularly important metric for spam email detection as it should be minimized in order to avoid falsely classifying legitimate emails as spam. Through our analysis, we find that SVM performs better than both KNN and logistic regression, but marginally worse than random forest (Table 2).

	Err	**FPR**
SVM	0.0632	0.0370
KNN	0.0997	0.0539
GLM	0.0795	0.0750
RF	0.0508	0.0296

Table 2: Misclassification and false positive rates of each method on predicting spam emails in the data.

5. CONCLUSION

In this study, we compared the performance of Support Vector Machines (SVM) to other popular classification methods, including k-Nearest Neighbors (KNN), logistic regression, and random forest, on the Spambase dataset. Our goal was to assess their accuracy in identifying spam emails and determine the method that achieved the lowest error and false positive rate.

Based on our comprehensive analysis and evaluation, we found that random forest outperformed SVM, KNN, and logistic regression in terms of both error rate and false positive rate. Random forest consistently demonstrated superior classification accuracy and showed better control over false positives when compared to the other methods.

While SVM showed competitive performance and exhibited its ability to capture complex decision boundaries, it was surpassed by random forest in terms of overall accuracy. This may be attributed to the ensemble nature of random forest, which combines multiple decision trees and leverages their collective predictive power. This ensemble approach allowed random forest to effectively capture the intricate patterns and relationships present in the Spambase dataset.

Nonetheless, SVM remained a viable and reliable option, displaying competitive performance in terms of classification accuracy. Furthermore, SVM's ability to handle high-dimensional feature spaces and its robustness against overfitting made it a valuable choice for spam email classification tasks.

REFERENCES

[1] Cortes, C. and Vapnik, V., "Support-vector networks," *Machine learning* **20**, 273–297 (1995).

[2] Platt, J., "Fast training of support vector machines using sequential minimal optimization," in [*Advances in Kernel Methods - Support Vector Learning*], MIT Press (January 1998).

[3] Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., and Weingessel, A., "Misc functions of the department of statistics (e1071)," *R package* **1**, 5–24 (2008).

[4] Karatzoglou, A., Meyer, D., and Hornik, K., "Support vector machines in r," *Journal of statistical software* **15**, 1–28 (2006).