Spam Classification

Ioannis Sevrisarianos

***Abstract*—The present project investigated the problem of classifying an email as spam or not via machine learning. First, pre-processing and exploratory data analysis were performed on the training dataset. Multiple classification algorithms were investigated and parameter turning was performed in order to find the optimal model for the task. A support vector classifier out-performed other options and was used to train a model, which achieved over 98% accuracy on training data. A further implementation allows classification of unseen data.**

***Index Terms*—spam, filter, classification, emails, machine learning**

I. Introduction

The rapid growth of electronic communication, has resulted in spam emails becoming increasingly prevalent and troublesome. Spam emails, also known as unsolicited bulk emails, inundate inboxes, waste time, and potentially pose security risks. To tackle this problem, the development of effective spam filtering systems is of utmost importance.

Spam filtering is the task of automatically identifying and separating unwanted spam emails from legitimate ones. Traditional rule-based approaches have limitations in adapting to new spamming techniques and suffer from high false positive rates. [1] As a result, machine learning techniques have emerged as powerful tools for building robust and accurate spam filters. [2].

Machine learning algorithms offer the ability to learn patterns and characteristics from large amounts of labeled data, enabling automated classification of emails as spam or non-spam based on their content, structure, and other relevant features. By training on diverse datasets, these algorithms can generalize and improve their performance over time.

The present project aims to evaluate different classification algorithms in regards to their accuracy. To that end, a training dataset will be pre-processed and explored, before being used to fit the various classifiers. Experiments will be conducted on the ideal values for selected parameters of each classifier, so that the ideal combination can be identified. Finally, the best performing classifier will be used in a model implementation for labeling unseen data.

By developing such a model, it is hoped to contribute to the ongoing efforts in combating spam emails and assists in the development of robust and adaptive machine learning solutions for spam detection and prevention.

II. Data

The dataset consists of 7500 email bodies and a ground truth label of 'spam', indicating unsolicited emails or 'ham', indicating useful emails.

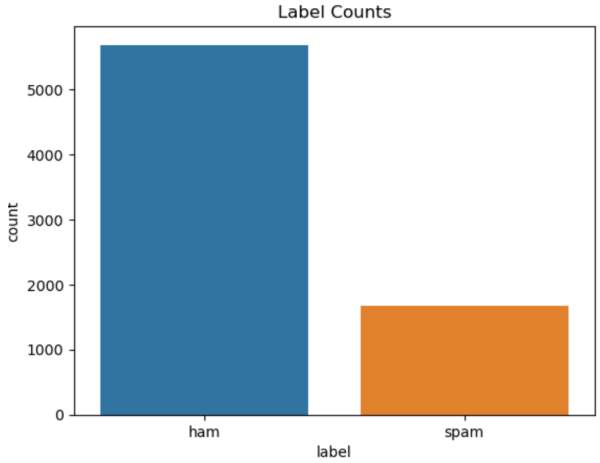

Fig. 1. Label Distribution

III. Pre-processing

Pre-processing was conducted in order to clean the data and result in a uniform text input. Specifically, all text was converted to lowercase, while punctuation and special characters were removed. At the same time, english stopwords were detected and removed using the relevant package from NLTK. Also, words with a length of three or less characters were omitted, so as to be guided to more a insightful representation of word frequency. Ubiquitous for an email words, such as 'subject' or 'mailto' were also removed. Finally, each word was substituted by using its lemma. Next, the text was tokenized and vectorized by implementing a tf-idf scheme, using a combined uni- and bi-gram range and following a euclidean distance calculation. Power normalization was then applied to the tf-idf scores to minimize the effect of vector magnitude on the predictions.

IV. Exploratory Data Analysis

The dataset was not found to contain any null values, but 136 duplicate lines, that were removed, were detected. An important observation was the significant label imbalance as seen in Figure 1. Due to that imbalance, the 'ham' labels have been undersampled for a more accurate fitting and evaluation of the models. The new label distribution can be seen in Figure 2 Furthermore, the frequency of words per label was investigated as seen in Figures 3 & 4. An interesting observation is regarding the frequency of words 'money', 'free', and 'please' in spam emails, while there can be seen an overwhelming prevalence of protocol-related words in 'ham' emails.

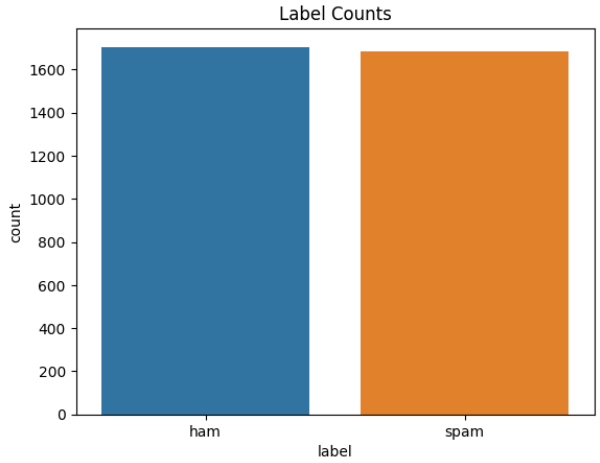

Fig. 2. New label distribution

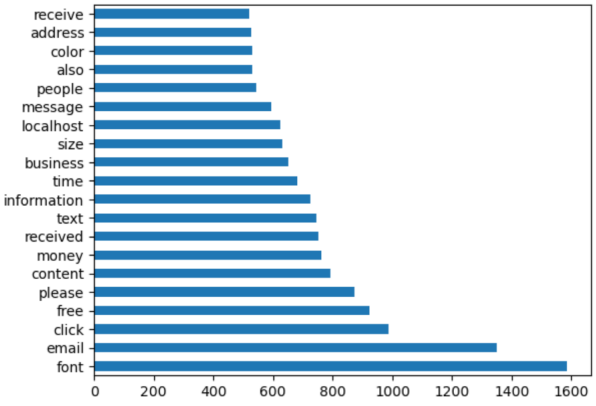

Fig. 3. Spam Words

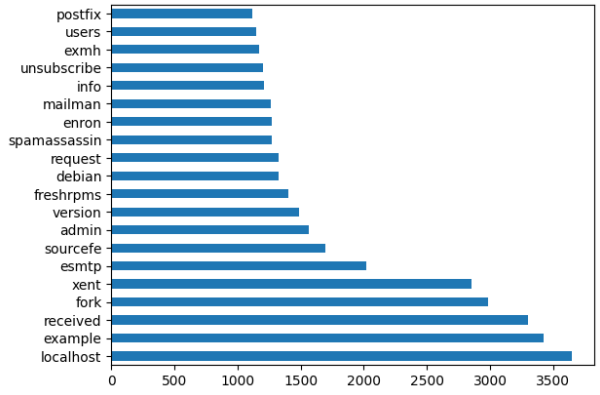

Fig. 4. Ham Words

V. Algorithm Evaluation

Five different classifiers were selected in order to approach this problem. Past literature [3] indicates that Naive Bayes is a strong algorithm for a task of this nature. The algorithm is effective with high-dimensional data, such as text data, while being fast and efficient. At the same time it is robust to irrelevant attributes, ignoring dependencies between them. As such, it is ideal for focusing on features lending to spam detection and ignore noise. Since the class labels have been balanced prior to training, it is assumed to be safe to use a Gaussian Naive Bayes classifier.

Secondly, a k-Nearest Neighbors algorithm was considered [4]. The algorithm is a strong candidate for classifying unknown data, despite is high time complexity. Moreover, it allows for intuitive understanding of the classification process and can help in identifying important features or patterns in spam emails. [5]

Given the state-of-the-art performance of Support Vector Classifiers in binary tasks [6], it was one more of the choices. The SVC is known for its resiliency to high dimensions and its ability to handle textual data, even though it is a harder model to interpret. At the same time, it provides flexibility with kernel choice, which will be beneficial in the parameter tuning stage. [7]

Random Forests [8] was also selected. Even though it might be slow at training, it is a good option for high dimensional data and often provides favourable classification performance. [9]

Finally, Stochastic Gradient Descent optimization [10] for Logistic Regression [11] and Perceptron [12] algorithms was attempted.

In order to evaluate the algorithms for the specific problem, two steps were followed.

First, different values for the various parameters of each algorithm were explored using a grid-search approach. Table 1 shows the parameters selected for each classifier. The grid search was performed using 5-fold cross-validation, while the accuracy metric was used for both refitting and evaluating. Even though other metrics, such as recall or precision are particularly useful when evaluating a spam detection classifier, the performance on unseen data will be measured in accuracy. Therefore, we will focus on this metric alone. The search yielded a best parameter combination for each classifier. These combinations can be seen in Table 2.

Secondly, based on the above results, a model was fitted for each classifier using those parameter values. The models were once more evaluated using a 5-fold cross-validation and the accuracy metric. The per-fold performance of the best found parameters for each algorithm can be seen in Figure 5, while the mean accuracy is demonstrated in Figure 6. Apart from Random Forest, all four other algorithms appear to exhibit fairly similar performance on our dataset. Out of those Support Vector Classifier appears to have the highest accuracy. At the same time, the time for performing the parameter tuning for each model can be seen in Table 3.

TABLE I
PARAMETERS SELECTED

Classifier	Parameters	Values
GaussianNB	alpha	[0.1, 0.5, 1.0]
	fit_prior	[True, False]
kNN	n_neighbors	[3,5,9]
	weights	[uniform, distance]
	p	[1,2]
Support Vector	C	[0.1, 1, 10]
	kernel	[linear, rbf, poly]
Random Forest	n_estimators	[50, 100, 200]
	criterion	[gini, entropy]
	max_depth	[5, 7, 10]
	oob_score	[True, False]
StochasticGD	loss	[log_loss, perceptron]
	penalty	[l1, l2, elasticnet]
	alpha	[0.0001, 0.0003, 0.0009, 0.001, 0.003, 0.009]
	max_iter	[1000, 3000, 9000]

TABLE II
PARAMETERS COMBINATION

Classifier	Combination
GaussianNB	alpha: 0.1 fit_prior: True
kNN	n_neighbors: 9 p: 2 weights: uniform
Support Vector	C: 1 kernel: linear
Random Forest	criterion: gini max_depth: 10 n_estimators: 100 oob_score: False
StochasticGD	loss: perceptron penalty: l2 alpha: 0.003 max_iter: 3000

TABLE III
PARAMETER TUNING TIME

Classifier	Time
GaussianNB	0m 1s
kNN	3m 50s
Support Vector	6m 5s
Random Forest	7m 40s
StochasticGD	0m 29s

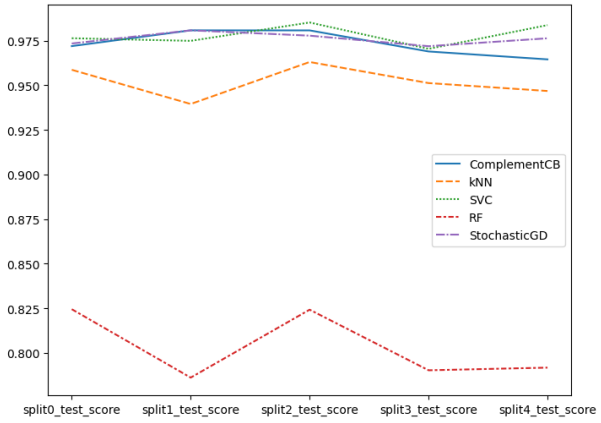

Fig. 5. Per-fold Accuracy

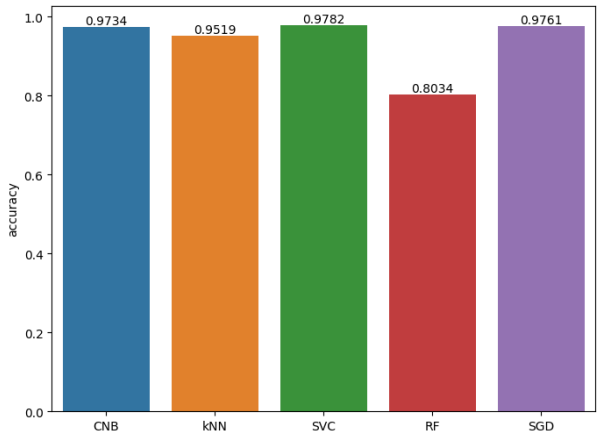

Fig. 6. Mean Accuracy

VI. IMPLEMENTATION

In a choice between the three highest performing algorithms, a choice was made in favour of the Support Vector classifier. Although Naive Bayes and Perceptron performed similarly, there are several factors that make SVC preferable. SVC is known to perform better than the other two in high-dimensional spaces, such as vectorized text data. At the same time, even though SVC requires more computational resources, it is considered a state-of-the-art algorith for tasks such as binary classification. Also, while it is a model harder to interpret, it did exhibit marginally better accuracy and provides the flexibility of handling non-linear data should it be required.

The implementation is comprised of a function that receives as arguments both a training dataset, as well as, an unseen before testing dataset. Both sets are first pre-processed in the same manner as the initial training dataset was. Then a vocabulary is created using the training set, while both sets are tokenized and then vectorized using that vocabulary. Similarly to earlier, all vectors are power normalized. A support vector classifier is then fitted using the training data and the parameters found by the above grid search operation. The fitted classifier can then make predictions by labeling

each item of the test dataset as either 'spam' or 'ham'. Those predictions are saved into a separate file as one prediction per line.

VII. Conclusions

The present project aimed to implement a model that can classify emails as either 'spam' or 'ham'. A training dataset was pre-processed and explored, showing a significant class imbalance and particular word frequency. The dataset was used to to investigate the best parameter values for five different classifiers. Support Vector Classifier outperformed the rest with an accuracy of 97.8%. The classifier was the implemented to train a model that can be used classify unseen emails.

References

[1] Chih-Hung Wu. Behavior-based spam detection using a hybrid method of rule-based techniques and neural networks. *Expert systems with Applications*, 36(3):4321–4330, 2009.

[2] Alexy Bhowmick and Shyamanta M. Hazarika. Machine learning for e-mail spam filtering: Review, techniques and trends. *CoRR*, abs/1606.01042, 2016.

[3] Ion Androutsopoulos, John Koutsias, Konstantinos V Chandrinos, George Paliouras, and Constantine D Spyropoulos. An evaluation of naive bayesian anti-spam filtering. *arXiv preprint cs/0006013*, 2000.

[4] Vishwanath Bijalwan, Vinay Kumar, Pinki Kumari, and Jordan Pascual. Knn based machine learning approach for text and document mining. *International Journal of Database Theory and Application*, 7(1):61–70, 2014.

[5] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[6] Harris Drucker, Donghui Wu, and Vladimir N Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural networks*, 10(5):1048–1054, 1999.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] Irena Koprinska, Josiah Poon, James Clark, and Jason Chan. Learning to classify e-mail. *Information Sciences*, 177(10):2167–2187, 2007. Including Special Issue on Hybrid Intelligent Systems.

[9] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[10] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[11] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[12] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.