Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.11, No.3, 2020
www.iiste.org

Spam Detection Using Machine Learning

Olubodunde Stephen Agboola
Department of Electrical Engineering, Louisiana State University
Patrick F. Taylor Hall, Baton Rouge, LA 70808

Abstract
Emails are essential in present century communication however spam emails have contributed negatively to the success of such communication. Studies have been conducted to classify messages in an effort to distinguish between ham and spam email by building an efficient and sensitive classification model with high accuracy and low false positive rate. Regular rule-based classifiers have been overwhelmed and less effective by the geometric growth in spam messages, hence the need to develop a more reliable and robust model. Classification methods employed includes SVM (support vector machine), Bayesian, Naïve Bayes, Bayesian with Adaboost, Naïve Bayes with Adaboost. However, for this project, the Bayesian was employed using Python programming language to develop a classification model.
Keywords: machine learning (ML), machine learning classifier, Naïve Bayes, SVM, Adaboost, spam classification, ham.
DOI: 10.7176/CEIS/11-3-04
Publication date: May 31st 2020

1. Introduction

In recent years, there has been an increase in unsolicited commercial / bulk e-mail also called spam, which has become a big trouble over the internet. Spam message wastes storage space, time, as well as bandwidth. This problem has been on the increase for years. In recent statistics, 45% of all emails are spam which about 15.6 billion email per day and that cost internet users over $300 million per year. An automatic e-mail filtering appears to be the most effective method to counter spam at the moment. With improving trickery from spammers, the classic solution of blocking a certain email address or filtering messages with certain subject lines have not been effective. Use of random sender addresses and/or append random characters to the beginning or the end of the message subject line by the spammers have effectively negated the classic approach. Knowledge engineering and machine learning are the two general approaches used in e-mail filtering. The use of a set of rules specified to which emails are categorized as spam or ham is referred to as knowledge engineering. This set of rules should be created either by the user of the filter, or by some other authority (e.g. the software company). Figure 1 shows a typical email spam deployment process. This method is effective to a specific set of messages defined as spam; hence it must be constantly updated and maintained to keep up with ever increasing type of spam messages. This is a waste of time and it is not convenient for most users. Machine learning has often been viewed in the same vein as rocket science when reported.

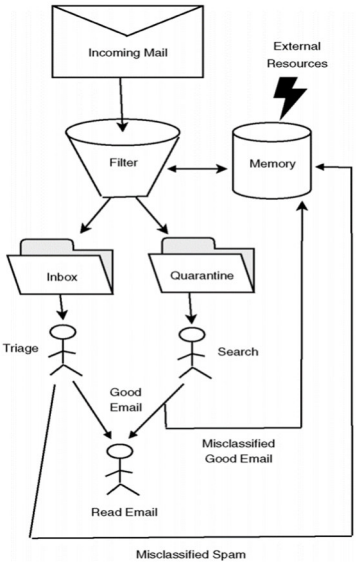


Fig 1 Typical spam filter

However, in reality, it is quite simpler than it poses. Machine learning refers to the ability of computers to learn to do something, without the need to explicitly program them for the task. Machine learning approach is

more efficient than knowledge engineering approach, as it does not require specifying any rules. Instead, a set of training samples, these samples is a set of pre-classified e-mail messages, is used to train an algorithm which is then set to classify in coming emails. Machine learning system is trained rather than explicitly programmed, data is inputted as well as the answers expected from the data and out comes the rules. Once the rules have been obtained, the system is said to have been trained. A way to measure whether the algorithm is doing a good job is to have a feedback to show how close the prediction is to the actual value. The adjustment step is called learning. First, the algorithm is made to look at a certain set of data, in order to train it for the task. Then, we give the algorithm data it has never seen before and perform the task on this data. A specific algorithm is then used to learn the classification rules from these e-mail messages. Machine learning approach has been widely studied and there are lots of algorithms can be used in e-mail filtering. They include Naïve Bayes, K-nearest neighbor, Neural Networks, support vector machines and the artificial immune system. Deep learning is a section of subfield of machine learning, it stands for the idea of successive layers of data representation which are learned habitually from introduction to training data set. A deep network can be thought of as information-distillation operation, where information goes through successive filters and comes out increasingly purified. Modern deep learning often has tens or even hundreds of successive layers of representation. This paper considered Naïve Bayes neural network system, implemented using python programming language on Jupyter notebook.

2 Structure of a usual Spam filter

The information contained in the email message is divided into the header; this field includes general information on the message, such as the subject, sender and recipient, the body; the actual contents of the message. Prior to classification of the message into spam or ham by a classifier, the message must be first filtered. This involves, extraction of the words in the message body (tokenization), reducing them to their root forms, eliminate some words that are usually used in many messages (stop words) and the presentation the set of words in a specific format to the algorithm. Fig 1 shows this process.

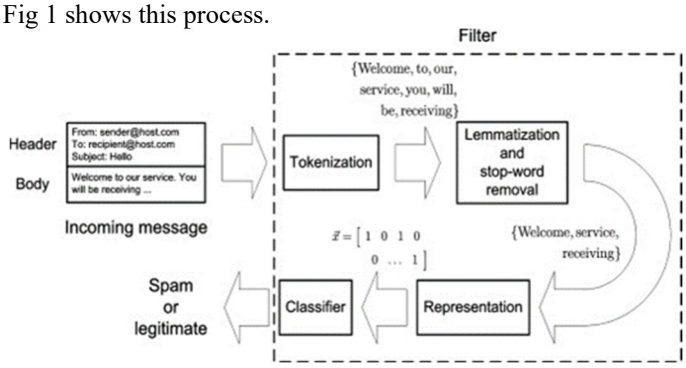


Fig 2 Illustration of steps involved in a spam filter.

A classifier is a function *f* that maps input feature vectors $x \in X$ to output class labels $y \in \{1, \ldots, C\}$, where X is the feature space. We will typically assume $\chi = R^D$ or $\chi = \{0, 1\}^D$, i.e., that the feature vector is a vector of D real numbers or D binary bits, but in general, we may mix discrete and continuous features. The goal is to learn *f* from a labeled training set of N input-output pairs, i.e. supervised learning.

A simple illustration is;

Input: X = email messages

Header: From:

Date:

Subject:

Body:

Output: $y \in \{spam, not\ spam\}$

Hence the objective is to obtain a predictor *f* that maps an input x to output y.

In deep learning the system does not need the data to be pre-processed; it learns the feature directly from the data set. It is also able to handle higher amount of data compare to machine learning producing a better performance. For this project two set of data was used, one for machine learning and the other for deep learning.

3.1 Naïve Bayes Classifier

This is a method of classification based on Bayes theorem. The conditional probability of an event is a probability obtained with the additional information that some other event has already occurred. P(B|A) is used to denote the conditional probability of event B occurring, given that event A has already occurred. The following formula was provided to obtain P(B|A):

$$P(B|A) = \frac{P(A \text{ and } B)}{P(A)} \quad (1)$$

A prior probability is an initial probability value originally obtained before any additional information is obtained. A posterior probability is a probability value that has been revised by using additional information that is later obtained.

Bayes' Theorem

The probability of event A, given that event B has subsequently occurred, is

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \quad (2)$$

Where A and B are

P(A|B) is a conditional probability: the likelihood of event A occurring given B is true. P(B|A) is also a conditional probability, the likelihood of event B occurring given A is true. The Naïve Bayes classifier calculates the probabilities for every factor. Then it selects the outcome with highest probability.

Naive Bayes assumption:

$$P(a_1, a_2 \ldots a_n|v_j) = \prod_i P(a_i|v_j) \quad (3)$$

$$\textbf{Naive Bayes classifier: } v_{NB} = \underset{v_j \in V}{\operatorname{argmax}} P(v_j) \prod_i P(a_i|v_j) \quad (4)$$

For the purpose of this research, rewriting the Naïve Bayes equation gives:

What we want is*: P (spam | words)*

$$P(spam \mid words) = \frac{P(words \mid spam)P(spam)}{P(words)} \quad (5)$$

$$P(words) = P(words \mid spam) \times P(spam) + P(words \mid good) \times P(good)$$

Assume independence: probability of each word independent of others

$$P(words|spam) = P(word1|spam) \times P(word2|spam) \times \ldots \times P(wordn|spam)$$

3.2 AdaBoost

The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners. Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules. Adaptive boosting works to re-weight the data rather random sampling. This method develops a concept of building ensembles for performance improvement of the classifiers. AdaBoost (Adaptive boosting) classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. Adaboost should meet two conditions:

1. The classifier should be trained interactively on various weighed training examples.
2. In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

The goal of boosting is to improve the accuracy of any given learning algorithm.

AdaBoost is popular for a number of reasons:

• It has no tricky parameters to set.
• It is computationally tractable and not too hard to implement.
• It generalizes very well and avoids overfitting.
• It satisfies some nice theoretical guarantees, as we will see later in this lecture and next time.

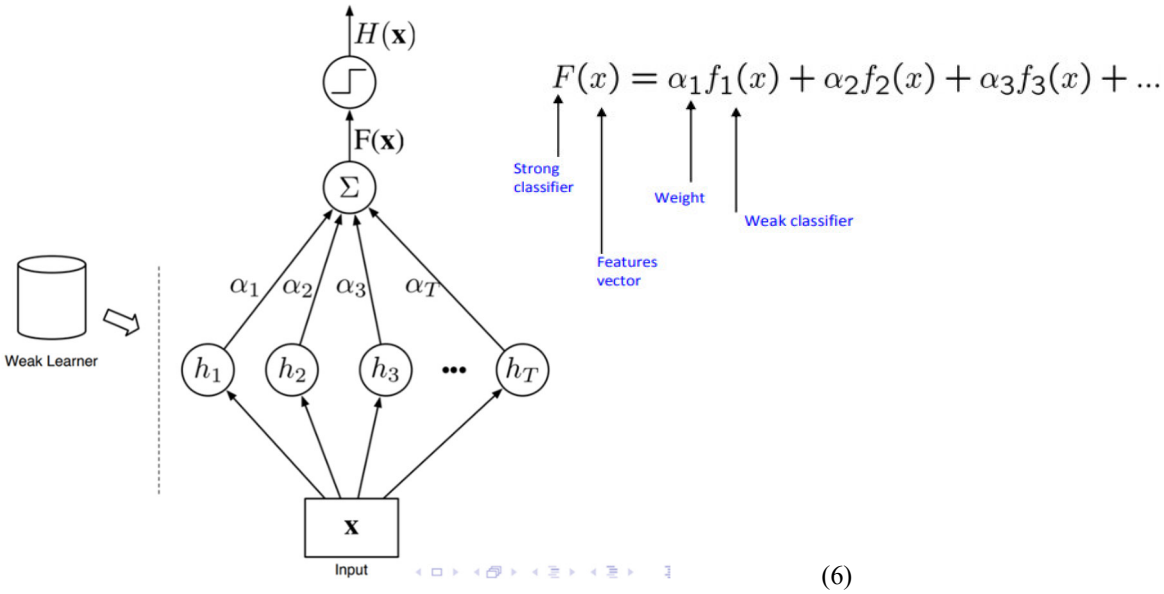
Weak Learner: A simple rule-of-the-thumb classifier that doesn't necessarily work very well. A class of concepts is learnable (or strongly learnable) if there exists a polynomial-time algorithm that achieves low error with high confidence for all concepts in the class. A weaker model of learnability, called weak learnability, drops the requirement that the learner be able to achieve arbitrarily high accuracy; a weak learning algorithm need only output an hypothesis that performs slightly better (by an inverse polynomial) than random guessing. Let D be a distribution over examples, and h be a classifier Error of h with respect to D is:

ErrorD (h(x)) ≡ Pr (X,Y) ~D(h(X) ≠ Y)

h is called a weak learner if *ErrorD (h(x))< 0.5*

With Adaboost, instead of sampling uses training set re-weighting each training sample uses a weight to determine the probability of being selected for a training set.

– Previous weak learner has only 50% accuracy over new distribution
– Can be used to learn weak classifiers
– Final classification based on weighted vote of weak classifiers



(6)

The selected weight for each new weak classifier is always positive. The smaller the classification error, the bigger the weight and the more this particular weak classifier will impact the final strong classifier.

$\varepsilon(h_a) < \varepsilon(h_b) \Longrightarrow \alpha_a > \alpha_b$

$$\alpha_t = \frac{1}{2} \times \ln \frac{1-\varepsilon}{\varepsilon} \quad (7)$$

Weight of classifier is straight forward; it is based on the error rate Ɛ. Ɛ is just the number of misclassifications over the training set divided by the training set size. After weak classifier is trained, we update the weight of each training example with following

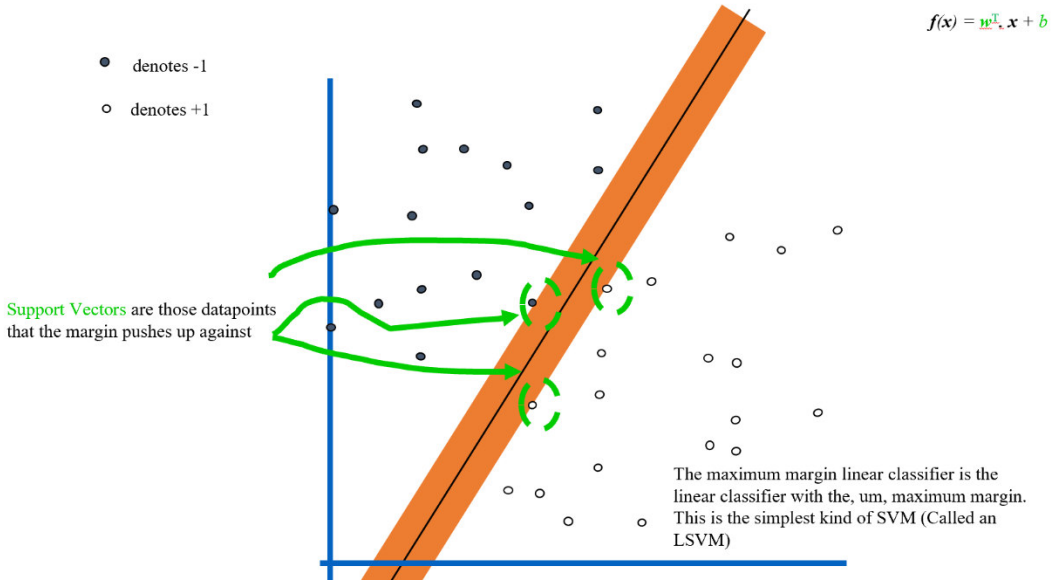
$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_t h_t(x_i))}{z_t} \quad (8)$$

We normalize the weights by dividing each of them by the sum of all the weights, Zt.

3.3 Support Vector Machine

Support vector model (SVM) seeks to draw a hyperplane to classify points in a finite dimensional space. SVM is a supervised learning model, i.e. it requires labeled input and output data or pattern be fed into the system, used for classification and regression analysis. For this paper linear SVM classifier is used, however it can efficiently accomplish a non-linear classification by employing the kernel trick. Given two classes,
$y_i \in \{-1, 1\}$ and there are n training samples, (x_1, y_1), ……, (x_n, y_n). if we run with the assumption that these two classes are linearly separable, then an optimal weight vector w can be obtained. A linear classifier is explained using the diagram below:

$$f(x) = w^T . x + b \quad (9)$$



From the diagram above the support vector classifier (SVC), drew an optimal hyperplane to separate the two points and introduce a maximum margin.

$$w^T . x + b < 0 \text{ predicts } +1$$
$$w^T . x + b = 0 \text{ On the hyperplane}$$
$$w^T . x + b > 0 \text{ predicts } -1$$

Mathematically the margin is maximized at 2/|W|.

4. Machine Learning

In this section, the dataset used was SMS Spam Collection of tagged messages that have been collected for SMS Spam research. It contained one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam. The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text. NumPy library; fundamental package for scientific computing with Python, was used for the analysis. Pandas was used for loading the dataset unto the program.

Table 1. Data Set

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN
5	spam	FreeMsg Hey there darling it's been 3 week's n...	NaN	NaN	NaN
6	ham	Even my brother is not like to speak with me. ...	NaN	NaN	NaN
7	ham	As per your request 'Melle Melle (Oru Minnamin...	NaN	NaN	NaN
8	spam	WINNER!! As a valued network customer you have...	NaN	NaN	NaN
9	spam	Had your mobile 11 months or more? U R entitle...	NaN	NaN	NaN

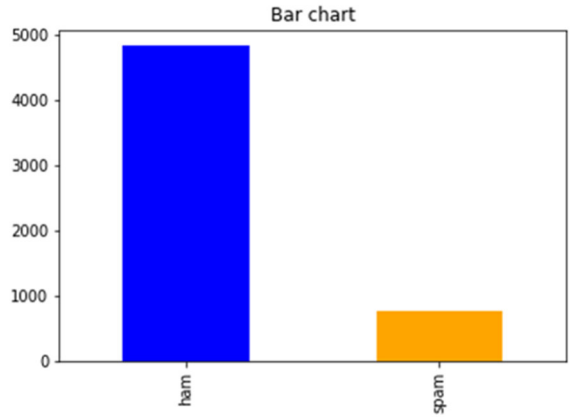


Fig 4 Bar Chart of dataset distribution

Further, the message; both spam and ham (not spam) were analyzed by extracting the words and showing its frequency.

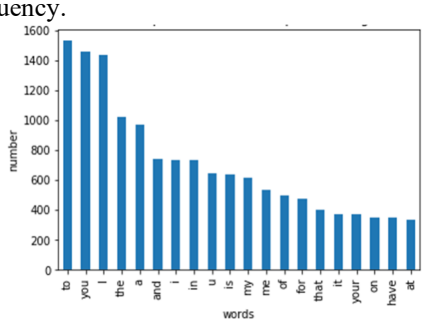


Fig 5 Bar Chart of more frequent words in ham messages

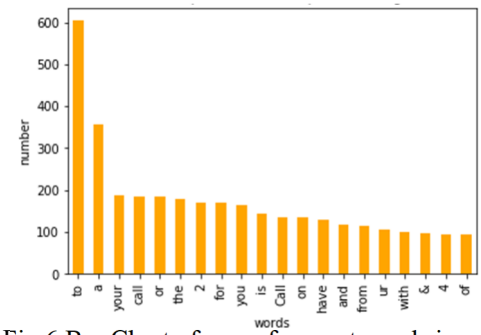


Fig 6 Bar Chart of more frequent words in spam

Text preprocessing, tokenizing and filtering of stop words are included in a high-level component that can build a dictionary of features and transform documents to feature vectors. For better analysis, the stop words were removed. The two possible outcome is either to predict spam as ham (false negative) or ham as spam (false positive). However, having a false positive is worse in this case because the ham message goes into spam and the recipient does not read it. The feature vector was converted to binary (spam = 1 and ham = 0) and split into training set and the testing set. The data set was applied using a regularization parameter α.

5. Result

In this section all the results obtained from the experiment was compiled and observed analytically. Obtaining a result of correct classification is the main goal, however the most important basis to determine the most effective of the methods is based on the ability of the model to accurately predict a ham message (precision).

5.1Naïve Bayes

Table 1. Table showing result using multi-nominal Naïve Bayes with varying alpha

	alpha	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.00001	0.998661	0.974443	0.920635	0.895753
1	0.11001	0.997857	0.976074	0.936508	0.893939
2	0.22001	0.997857	0.977162	0.936508	0.900763
3	0.33001	0.997589	0.977162	0.936508	0.900763
4	0.44001	0.997053	0.977162	0.936508	0.900763
5	0.55001	0.996250	0.976618	0.936508	0.897338
6	0.66001	0.996518	0.976074	0.932540	0.896947
7	0.77001	0.996518	0.976074	0.924603	0.903101
8	0.88001	0.996250	0.976074	0.924603	0.903101
9	0.99001	0.995982	0.976074	0.920635	0.906250

```
alpha              15.730010
Train Accuracy      0.979641
Test Accuracy       0.969549
Test Recall         0.777778
Test Precision      1.000000
Name: 143, dtype: float64
```

	Predicted 0	Predicted 1
Actual 0	1587	0
Actual 1	62	190

It was observed that a test precision of 1.0 was obtained when the hyperparameter, alpha, was 15.73. The model produced a perfect classification of ham messages while wrongly classifying 62 spam messages.

5.2 Support Vector Classifier

Table 2. Table showing result using the SVC with varying C

	C	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	500.0	0.994910	0.982599	0.873016	1.0
1	600.0	0.995982	0.982599	0.873016	1.0
2	700.0	0.996785	0.982599	0.873016	1.0
3	800.0	0.997053	0.983143	0.876984	1.0
4	900.0	0.997589	0.983143	0.876984	1.0
5	1000.0	0.998125	0.983143	0.876984	1.0
6	1100.0	0.998928	0.983143	0.876984	1.0
7	1200.0	0.999732	0.983143	0.876984	1.0
8	1300.0	1.000000	0.983143	0.876984	1.0
9	1400.0	1.000000	0.983143	0.876984	1.0

```
C                  500.000000
Train Accuracy       0.994910
Test Accuracy        0.982599
Test Recall          0.873016
Test Precision       1.000000
Name: 0, dtype: float64
```

	Predicted 0	Predicted 1
Actual 0	1587	0
Actual 1	31	221

C is the regularization parameter. It determines the size of the margin. For large value of C, the hyperplane margin is small to ensure that all or most of the training points are classified correctly. However, for small value of C, the optimizer finds a hyperplane with a large margin at the cost of misclassifying some points. Hence adjusting the C parameter is a trade-off between obtaining low training error and low testing error. From the table above, the accurate with a small C was obtained when C was set to 500. SVC like Naïve Bayes produced a result that correctly classifies all the ham messages while wrongly classifying 31 spam messages. In other words, it has a high lower negative relative to the Naïve Bayes.

5.3 AdaBoost

The AdaBooost model was employed using SVC as the base classifier.

Table 3. Using a learning rate of 0.5 with different weak classifiers

Number of weak classifiers	Base Estimator	Learning rate	Test Accuracy
50	SVC	0.5	0.9821
100	SVC	0.5	0.9810
150	SVC	0.5	0.9766

Table 4. Using a learning rate of 0.25 with different weak classifiers

Number of weak classifiers	Base Estimator	Learning rate	Test Accuracy
50	SVC	0.25	0.9864
100	SVC	0.25	0.9858
150	SVC	0.25	0.9858

Table 5. Using a learning rate of 0.1 with different weak classifiers

Number of weak classifiers	Base Estimator	Learning rate	Test Accuracy
50	SVC	0.1	0.9858
100	SVC	0.1	0.9858
150	SVC	0.1	0.9864

Table 6. Using 150 weak classifiers and 0.3 learning rate.

	trials	Train Accuracy	Test Accuracy	Test Recall	Test Precision
0	0.0	1.0	0.986406	0.904762	0.995633
1	1.0	1.0	0.985318	0.900794	0.991266
2	2.0	1.0	0.985862	0.904762	0.991304
3	3.0	1.0	0.985318	0.900794	0.991266
4	4.0	1.0	0.985318	0.900794	0.991266
5	5.0	1.0	0.985318	0.896825	0.995595
6	6.0	1.0	0.985318	0.896825	0.995595
7	7.0	1.0	0.984774	0.896825	0.991228
8	8.0	1.0	0.985318	0.900794	0.991266
9	9.0	1.0	0.985318	0.896825	0.995595

```
trials              0.000000
Train Accuracy      1.000000
Test Accuracy       0.986406
Test Recall         0.904762
Test Precision      0.995633
Name: 0, dtype: float64
```

```
trials              0.000000
Train Accuracy      1.000000
Test Accuracy       0.986406
Test Recall         0.904762
Test Precision      0.995633
Name: 0, dtype: float64
```

	Predicted 0	Predicted 1
Actual 0	1586	1
Actual 1	26	226

From the tables above, it is seen that getting a balance between the learning rate and base classifier affects the test accuracy and precision. The best value was obtained using 150 weak classifiers with learning rate of 0.3. It also wrongly classifies one ham message and 26 spam messages.

6. Conclusion

From the obtained results, it can be deduced that the native Bayes classification is 100% efficient as some spam messages were misclassified. SVC also produced no false positives with less false positives compared to Naïve Bayes. Finally, the use of number of combined SVC classifiers in and Adaboost model produce a more balanced result. It misclassified one spam message however with less misclassification of spam as ham. Future research in this area can be geared towards developing a deep learning model that can offer a better classification rate.

References

Awad, W. (2011). Machine Learning Methods for Spam E-Mail Classification. *International Journal of Computer Science and Information Technology*, *3*(1), 173-184. doi:10.5121/ijcsit.2011.3112

C. Ross Quinlin, *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann, 1988.

Deep Learning Tutorial for Beginners | Kaggle. (n.d.). Retrieved from https://www.kaggle.com/kanncaa1/deep-learning-tutorial-for-beginners/notebook

Guzella, T. S., & Caminhas, W. M. (2009). A review of machine learning approaches to Spam filtering. *Expert Systems with Applications*, *36*(7), 10206-10222. doi:10.1016/j.eswa.2009.02.037

Larsen, K. (2005). Generalized Naive Bayes Classifiers. *ACM SIGKDD Explorations Newsletter*, *7*(1), 76-81. doi:10.1145/1089815.1089826

Naive Bayes & SVM Spam Filtering | Kaggle. (n.d.). Retrieved from https://www.kaggle.com/pablovargas/naive-bayes-svm-spam-filtering/notebook

T. Joachims, “Text categorization with support vector machine: Learning with many relevant features,” in

European Conf. Machine Learning, 1998.

Trivedi, S. K. (2016). A study of machine learning classifiers for spam detection. *2016 4th International Symposium on Computational and Business Intelligence (ISCBI)*. doi:10.1109/iscbi.2016.7743279

Visani, C., Jadeja, N., & Modi, M. (2017). A study on different machine learning techniques for spam review detection. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*. doi:10.1109/icecds.2017.8389522

[Wang and Manning2012] Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *ACL*.

[Weinberger et al.2009] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *ICML*.

[Weston et al.2011] Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*.

[Weston et al.2014] Jason Weston, Sumit Chopra, and Keith Adams. 2014. #tagspace: Semantic embeddings from hashtags. In *EMNLP*.

[Xiao and Cho2016] Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

[Zhang and LeCun2015] Xiang Zhang and Yann LeCun. 2015. Text understanding fromscratch. *arXiv preprint arXiv:1502.01710*.

[Zhang et al.2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*