Spam Emails Detection: a comparison between Machine Learning and Deep Learning techniques

Andrea Cadoni, University of Bologna
andrea.cadoni2@studio.unibo.it
Number: 0001024145

September 22, 2023

Abstract—In recent years, there has been a growing interest in using Machine Learning techniques to tackle the pervasive issue of Spam Emails. One particularly effective approach is leveraging statistical features, which provide valuable insights into the characteristics of spam and non-spam emails. This project aims to develop a Machine Learning model for email classification based on classification algorithms, by training the model on a comprehensive dataset of labeled email samples to enable automated spam detection. Moreover, a comparison will be made between Machine Learning and Deep Learning techniques. The final goal of this paper is to present a comparative analysis of different approaches, understanding their relative effectiveness in identifying Spam Emails.

1 Introduction

E-mails have been one of the first means of communication in the new Internet age. Such a powerful tool has required so many adaptations over the years, when technologies have constantly changed, and at the same time the 'alternative' use for purposes that are not entirely licit has amplified.

The rise of the digital age has witnessed an alarming surge in the misuse of e-mails as a medium for phishing attempts, fraudulent schemes, and other forms of cybercrime. These illicit activities have raised serious security and privacy concerns for both individuals and organizations. Countless users have fallen prey to cunningly crafted e-mails, leading to devastating consequences such as financial loss, identity theft, and compromised personal information.

Aside from security-related issues, another significant challenge posed by Email communication is the overwhelming quantity of *Spam Emails*. These unsolicited messages fill our inboxes daily, impeding efficient communication and causing frustration for users. Spam e-mails, often promoting dubious products, services, or fraudulent schemes, not only consume valuable time but may also expose recipients to potentially harmful content, such as malware and phishing links.

Addressing the spam problem is crucial for enhancing the overall quality of Email communication and safeguarding users from potential risks. As Emails remain a vital channel for business communication, marketing, and personal correspondence, mitigating the impact of spam is vital to preserving the trust and reliability of this ubiquitous communication medium.

In this paper, we will go through a section titled *Related Work* to clarify the existing research conducted by other authors in this particular field. Subsequently, we will illustrate our *Proposed method* for constructing models by providing comprehensive explanations on the specific techniques and parameters used. Following this, the section *Results* will present the metrics applied to evaluate the model, providing detailed information on the ratios and graphs generated. Lastly, within the *Conclusions* chapter, we will present our optimal solution and rationale behind it, along with providing some valuable insights on potential enhancements that could be made to the model.

2 Related work

Doing some research, we were able to find some papers related to the topic of *Spam Emails Classification*. We will now go through some of these papers.

2.1 Machine Learning based Spam E-Mail Detection - Priti Sharma, Uma Bhardwaj

In this paper the authors try to classify Emails as [Spam, Ham (Non-spam)]; in order to do so, they used two famous machine learning algorithms, Naïve Bayes and J48 (Decision Tree). In combination with these two algorithms, the authors implemented an *Hybrid Bagged Approach*, where overall system's result is the average of the result of the two classification algorithms; J48 algorithm and Naïve Bayes are used for the multi class learning and for the classification. The Dataset considered has not been specified, but it is represented by Raw Email Data, requiring the creation of a filtering system. This system is summarized in the figure below.

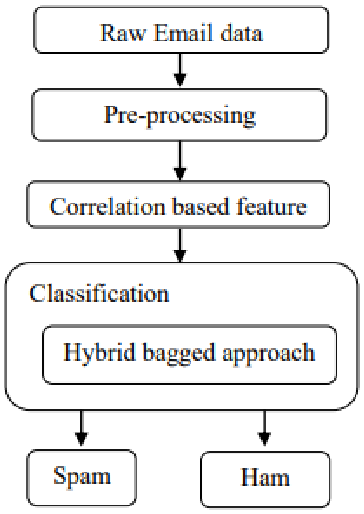

Fig. 1. Basic process for email filtering

Apart from the application of the two algorithms, is curious the creation of a *Correlation based feature selection (CFS)*: initially text data with feature set is considered as bag of words. The term frequency method is considered to show the number of words per document; the frequency of all words is calculated and words with frequency below a threshold value are eliminated. This method indicates the usefulness of the words and also reduces the search space. The obtained feature set is further reduced using correlation based feature selection method.

The classification results can be divided as follow:

- **Naïve Bayes**: Precision (85.26), Accuracy (83.5).
- **J48 (Decision Tree)**: Precision (93.68), Accuracy (91.5).
- **Hybrid Bagged Approach**: Precision (89.47), Accuracy (87.5).

2.2 Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection - Faris, Hossam and Aljarah, Ibrahim and Alqatawna, Ja'far

The authors in this paper had the goal to classify Emails as [Spam, Not-Spam] using a FeedForward Neural Network. In order to do so, they considered the *SpamAssassin* Dataset. The peculiarity of this work is that it has been carried out through a new nature-inspired metaheuristic algorithm, *Krill Herd (KH)*; some of the main advantages of the algorithm is that it doesn't need derivative information because it performs a stochastic search instead of a gradient search. Moreover, compared to other heauristic optimization algorithms KH needs few parameters to be tuned; particularly, it only needs a time interval parameters to be adjusted.

The classification results present a huge gain in accuracy: a classification with a standard MLP-Backpropagation resulted in an accuracy score of **70.38**, while with MLP-KH the score has been **91.08**.

2.3 Text and image based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm - Aman and Gupta

The authors in this paper had the goal to classify Emails as [Spam, Not-Spam] using some Machine Learning algorithms. In order to do so, they considered the *Enron Corpus* Dataset. We decided to consider this paper since it carries out the classification task through algorithms which won't be used in our paper, namely: KNN algorithm, Naïve Bayes algorithm and reverse DBSCAN algorithm. The peculiarity of this work is that it used a *Black and White listing* approach: all those web pages and domains that are notorious for sending spam Emails are not trusted and therefore they go directly on the black list (without any further processing). Concurrently, a white list is maintained where users can mark those websites they want mails from whether they send "spam" or not. Thus no processing is done when a white listed domain matches.

The classification results can be divided as follow:

- **KNN**: Precision (95.38), Accuracy (83.92).
- **Reverse DBSCAN**: Precision (73.79), Accuracy (74.33).
- **Naïve Bayes**: Precision (82.95), Accuracy (86.83).

3 Proposed method

The first step was to find a good Dataset which was able to satisfy our needs; for this project we decided to use the *Spambase Dataset* from the *UCI Machine Learning Repository*. This Dataset has been donated on 30/6/1999 and its main purpose is to classify Emails as Spam or Non-Spam, so we are dealing with a *binary classification problem*. There were loads of different datasets regarding this task, but this was the best one and its peculiarity concerns the presence of statistics about Emails, rather than the Emails' header and body themselves.

The Dataset counts 57 attributes/features, 1 attribute label, and 4601 instances. The features are defined as follow: [1]

- **48 continuous real [0,100] attributes of type word_freq_WORD**. It represents the percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.
- **6 continuous real [0,100] attributes of type char_freq_CHAR]**. It represents the percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail.
- **1 continuous real [1,...] attribute of type capital_run_length_average**. It represents the average length of uninterrupted sequences of capital letters.
- **1 continuous integer [1,...] attribute of type capital_run_length_longest**. It represents the length of longest uninterrupted sequence of capital letters.
- **1 continuous integer [1,...] attribute of type capital_run_length_total**. It represents the total number of capital letters in the e-mail.
- **1 nominal 0,1 class attribute of type spam**. It denotes whether the e-mail was considered spam (1) or not (0).

It is important to note that this Dataset has missing values.

3.1 Machine Learning Algorithms

Classification is one of the most common activities in the field of Machine Learning. For this task, the main objective is to assign a data/object to one of the predetermined categories or classes. This process involves training a model on a set of known training data so that it can learn to distinguish different classes based on the characteristics or attributes of the data. Once trained, the model uses validation data in order to tune the hyperparameters. In the last phase, the model is applied to the test data in order to understand how the model perform on new data; subsequently, we calculate different metrics to verify the overall performance of the model.

Operationally, we first extracted the features and the labels, converting the features in float values in order to not have problems with the usage of some *scikit-learn* algorithms. Then we randomized the data with the seed (1821) and split the data in training (60%), validation (10%) and testing (30%). After this preparation, the final step is the real classification.

Here we applied several well-known Machine Learning algorithms, like Support Vector Machines, RandomForest classifier, Decision Tree classifier, AdaBoost and Logistic Regression.

3.2 Deep Learning Algorithms

Deep Learning is a discipline, similar to Machine Learning, that permits to learn from data and generalize the knowledge on other new data; differently from Machine Learning, we are able to skip the feature extraction phase (in case of high dimensionality of data) thanks to *Neural Networks*. An important implementation of neural networks is the *Multi-Layer Perceptron (MLP)*. MLP consists of one or more layers hidden between the input layer and the output layer, where each neuron in a layer is connected to all neurons in the preceding and following layer. MLPs are known for their ability to learn complex patterns and relationships in data.

Operationally, we have split the data in training (80%) and validation (20%). When it came to define the architecture of the Multi Layer Perceptron, we decided to define 5 neural layers, specifically:

- Input Layer: this is the initial layer that accepts the input data, which has 57 neurons corresponding to the number of features or attributes present in the data.
- Dense Layer(s): this is a fully connected neural layer with 128 neurons and a ReLU (Rectified Linear Unit) activation function. There are also 2 additional layers with 64 and 32 neurons respectively.

1. https://archive.ics.uci.edu/dataset/94/spambase

- Output Layer: this is the output layer with a single neuron, used for binary classification. It has a *sigmoid activation function*, which produces a probability as the output for classification.

Eventually, we have defined the epochs (30) and we wrote the code to compile the model. The gradient descent approach chosen is *Adam*, while we used *Binary Cross Entropy* as loss function.

3.2.1 MLP with Dropout added

Dropout helps prevent overfitting, which is a condition in which the neural network fits the training data too well, but does not generalise well on new data. At the beginning of training, the weights of the connections between units are randomly initialised; during each iteration, some units are deactivated, which forces the network to learn more robust and independent representations of the data. We split as for standard MLP the dataset into training (80%) and validation (20%), and then we added two layers of dropout with a value of 0.3, which indicates the fraction of units (neurons) in the layer that will be randomly switched off during training in each step (epoch).

Every other step is equal to the MLP without Dropout added.

4 Results

4.1 Machine Learning

To sum up, regarding the sphere of Machine Learning we have used a variety of algorithms that led to different accuracy and elapsed time values. The results can be seen below:

Algorithm used	Accuracy	Elapsed time (s)
SVM (kernel RBF)	0.902	0.837
DecisionTree	0.895	0.156
RandomForest (1 feature)	0.849	0.106
RandomForest (8+ features)	0.922	0.115
AdaBoost	0.934	0.810
Logistic Regression	0.907	0.179

TABLE 1
Accuracies

As we can see from the table, even though the elapsed time is bigger, the best accuracy value is achieved through the usage of **AdaBoost**. Therefore we will compute the confusion matrix on this algorithm. NB: confusion matrix is a metric used in multiclass classification problems to understand how errors are distributed; its application in our binary classification problem is not so useful, but we will proceed anyway just to have a visual representation of the error distribution.

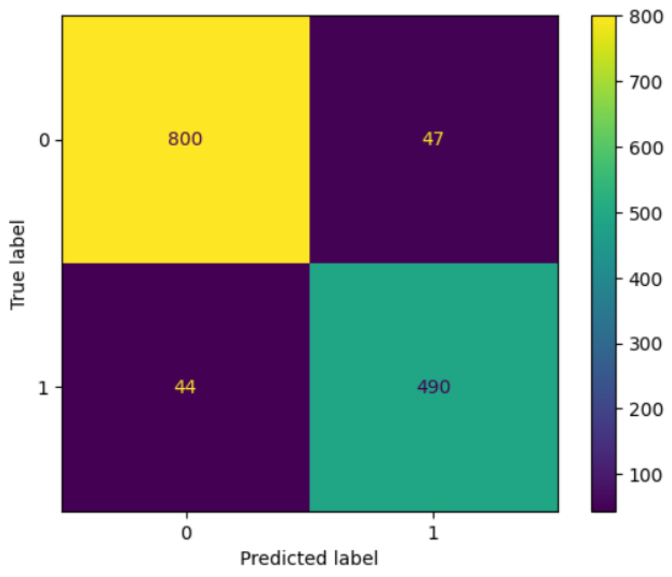

Fig. 2. Confusion Matrix

As expected, the higher values are on the main diagonal; high (off-diagonal) values would indicate concentrations of errors.

Let's make a comparison with the data about accuracy provided by the UCI Machine Learning Repository website:

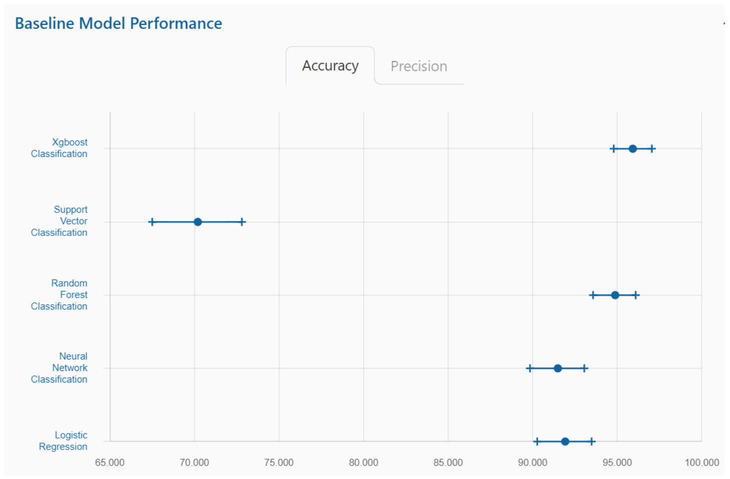

Fig. 3. Baseline Model Performance: Accuracy

As we can see in the above figure, the accuracy values of *RandomForest* and *Logistic Regression* are pretty aligned with what we have observed in our analyses. The only accuracy value that differs quite significantly from our analysis is the one for *Support Vector Machines*.

4.2 Deep Learning

We deployed the first model without any dropout added over 30 epochs, and the final accuracy value is **0.9435**, with a validation accuracy of **0.9327**; the loss value at the final epoch is **0.1551**, while the val_loss is **0.2417**. As we can see in figure 4, the trend of accuracy is rather upward steadily, with a stabilization following the epoch 20. The trend for the loss function is also extremely positive and it can be seen in figure 5.

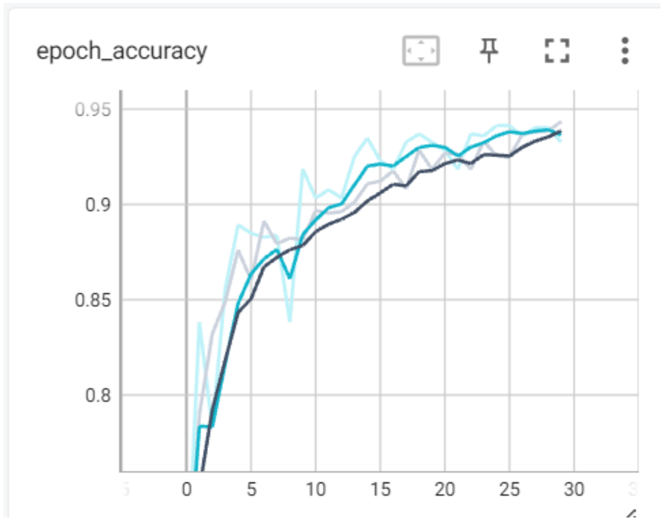

Fig. 4. Epoch Accuracy

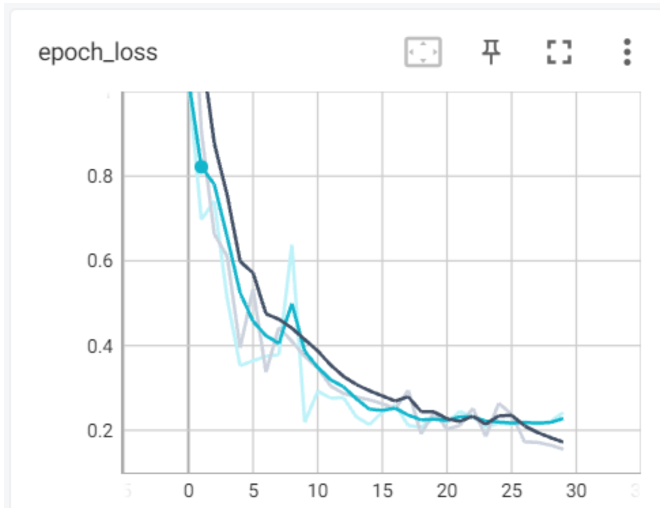

Fig. 5. Epoch Loss

We have also deployed the model adding two layers of dropout. Precisely, we inserted one dropout layer after the first dense layer (128 neurons) and another dropout layer after the second dense layer (64 neurons). Thus, we used two dropouts at a rate of 0.3, which means that during training, approximately 30% of the units of each of these layers will be randomly dropped out in each training step for regularisation purposes. In this context, the overall accuracy dropped quite significantly, with the final value set at **0.8927** (accuracy) and **0.9316** (validation accuracy); the loss value at the final epoch is **0.2778**, while the val_loss is **0.2249**. This result is in line with our expectations, considering that we have seen that our model doesn't suffer from overfitting, and then adding Dropout layers leads to a reduction in performance instead of an improvement.

5 Conclusions

This project has been based on the Spambase dataset, aimed at addressing the challenging task of email classification through statistics and not texts. We based our experiments on some of the most used Machine Learning algorithms, like SVM or RandomForest, and even on Multi Layer Perceptron, one of the most common Neural Networks. These models served as our baseline, offering valuable insights into their performance for Spam Emails Classification.

In assessing our models' performance, we utilized a range of metrics to estimate mainly accuracy and precision. Ultimately, our findings pointed to *AdaBoost* and *MLP without Dropout added* as the most promising candidates, achieving notable accuracy scores.

Future developments might encompass exploring advanced Deep Learning architectures, fine-tuning hyperparameters, or investigate into ensemble methods to further boost classification accuracy.

References

[1] Sharma, Priti, and Uma Bhardwaj. "Machine Learning based Spam E-Mail Detection." International Journal of Intelligent Engineering & Systems 11.3 (2018).

[2] Faris, Hossam, Ibrahim Aljarah, and Ja'far Alqatawna. "Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection." 2015 IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT). IEEE, 2015.

[3] A. Harisinghaney, A. Dixit, S. Gupta and A. Arora, "Text and image based spam email classification using KNN, Naïve Bayes and Reverse DBSCAN algorithm," 2014 International Conference on Reliability Optimization and Information Technology (ICROIT), Faridabad, India, 2014.