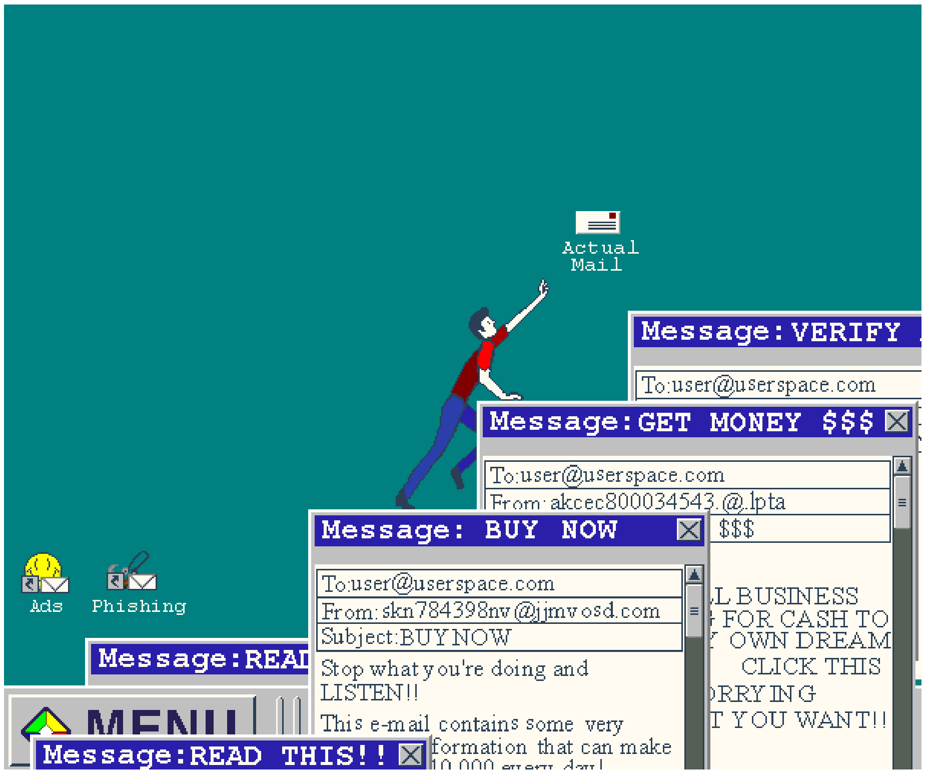
SpamBase

Use of E-Mail Message Content and Machine Learning to Determine Spam Messages

Jack Crowley

December, 2020



Abstract

This project is an analysis of the Spambase Dataset from the University of California: Irvine Machine Learning Dataset Repository. The goal of this project is to develop a Neural Network that can determine whether an E-mail message is spam by learning from the Spambase Dataset. This project explains the dataset, the preparations to use the dataset, the Neural Network used for Machine Learning, and Potential Feature Removal and Simplification.

The Spambase Dataset was donated to the UCI Archives in 1999. This dataset contains different e-mail messages as samples, defining each one as spam or not using parameters such as capital letter strings, keywords, and punctuation frequency. This dataset is prepared using Standardization with each input feature's Standard Deviation and Mean.

The neural networks for this assignment had to remain small. The neural networks typically used for this project involved at least one hidden layer, making three layers total. The node pattern was typically $6, 2, 1$, but even with a simple neural network, overfitting remained a problem with the dataset.

The results of the tests were that the dataset was prone to generating noise and overfitting. Linear regression tests returned unusual changes and fitting, despite the dataset's purpose as a classification dataset instead of a regression dataset.

When removing the input features, the accuracy decreased with each batch of 5 feature removals. However, there was a distinct curve upwards when removing 30 and 35 input features.

Contents

1 Introduction

Spam is the act of sending a repeated message throughout a network, and remains one of the most troubling and annoying aspects of online communication. Spam has become a hindrance in the online world due to the ease of producing different spam messages. E-mail spam can hinder and annoy account users, and some Spam messages are also dangerous due to malicious attachments and hyperlinks.

Spam can be for a number of purposes. Advertisements are a common use for spam. Spam can provide advertisers with the ordinance to send messages across the internet to different users, and possibly with filters and potential customer lists to personalize advertisements. Spam however, can also be used for Phishing attacks. An attacker can use spam to expose a victim to a Macro-embedded document or a Malicious Hyperlink that will allow the attacker to infect a system. Attackers use Trojan Downloads to infiltrate systems using Phishing attacks. Macro documents or Hyperlinks lead to ”drive-by-download” attacks where the Malware enters the system through a malicious fast download.

To defend against Spam, an E-mail provider could check for certain information about the e-mail's content before the user reads it to determine if the message is Spam. Certain keywords, such as ”money,” ”buy,” or dialogue involving ”you,” ”your,” or ”now” could involve purchasing some kind of product. However, Spam may not be advertisements. Spam could just as easily be an annoying message from an unknown individual in capital letters and exclamation points. Those two factors could also be taken into account.

The content of Spam can change, and Spam can be sent for many purposes. A machine learning approach could defend a user E-mail system by dynamically changing its analysis on spam. A Good Spam System should change its vocabulary and learn different structures to keep up with the times and stop Spam as Spam changes.

The goal of this project is to use the Spambase Dataset to develop a Neural Network for a Machine Learning Solution to Spam Detection. A machine learning product could learn Spam from a set of training values involving the data, and then learn new information as Spam changes over time.

2 Spambase Dataset Introduction

This project makes use of the ”Spambase” Dataset, which was donated to the University of California, Irvine Academy, in 1999. This Dataset, though outdated, is intended for machine learning using Classification Techniques.

”http://archive.ics.uci.edu/ml/datasets/Spambase/”

This dataset uses 57 different input features to determine if a message is ”spam.” These input features involve different keywords, capital letter string details, and

2.1 Name Percentages

The first section of input fields is the percentage of times a certain phrase is found in the spam message.

Each column counts the words using the following equation:

$$\frac{(FrequencyofWord)}{(WordsTotal)} * 100$$

Key Phrases			
make	address	all	3D
our	over	remove	internet
mail	receive	will	people
report	addresses	free	business
business	email	you	credit
your	font	000	hp
hpl	650	lab	labs
telnet	857	data	415
85	technology	1999	parts
pm	direct	cs	meeting
original	project	re	edu
table	conference	blank	blank

2.2 Character Frequency

The second section is the frequency of certain characters.

These characters are all punctuation marks, including the following:

- ";"
- "("
- "["
- "!"
- "#"
- "$"

2.3 Capital Letters

The final section concerns the use of capital letters and running lengths of capital letters.

- Average Capital Run Length
- Length of Longest Capital Letter Run
- Total Number Capital Letter Runs

3 Python3 Notebook Link

All the Work of this notebook is kept in this Colab Notebook here.
https://colab.research.google.com/drive/1cAGgho8TbSN1SglfM2o3Mqe5icu2bnA7?usp=sharing

4 Standardization of Data

Before the Neural Network can learn from the Dataset, the Dataset must be normalized to prepare it for use. To normalize, this system uses the Standardization method for Normalization.

The Standardization Practics involves taking the Standard Deviation and the Mean of each entry's input feature. For every column of input features, each entry has the input feature subtracted against the input feature's standard deviaton, and then divided by the mean.

5 Neural Network Design

The Machine Learning Aspect of this Project uses a Neural Network with Three Layers.

The structure of this Neural Network is $6, 2, 1$. The equation for improvement is "Binary Crossentropy" and the Optimizer Function is "Adam." These were found to be useful functions during a test with different fitting parameters.

The epochs for this system are 256. The Neural Network tends to overfit easily, so the Epochs are kept to something manageable, yet enough to provide a description for the testing. This system also uses a Batch Size of 32.

The size of the Validation Dataset is 25 percent of the original dataset. The Validation set makes up 1150 of the 4601 total entries.

6 Test 1: Classification and Regression

In Phase 2, one goal was to determine the results between classification and regression to solve the problem.

The Regression Dataset uses the same number of neurons and layers, but the Optimizer is "rmsprop" and the success metric and loss is calculated with Mean Squared Error.

From these results, the Regression Dataset ended up being very tame and noiseless compared to the Sigmoid Dataset. The Regression Dataset did not have as much noise, and showed no real signs of overfitting, even when the epochs reached the end of the 256 segment.

7 Test 1a: Model Checkpointing

Another test was the use of Model Checkpointing and Early Stopping to improve the dataset. This test involves using the same dataset, but with the Model

Figure 1: Machine Learning with Binary Sigmoid Classification

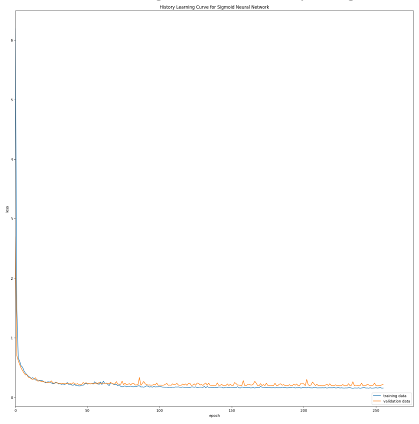
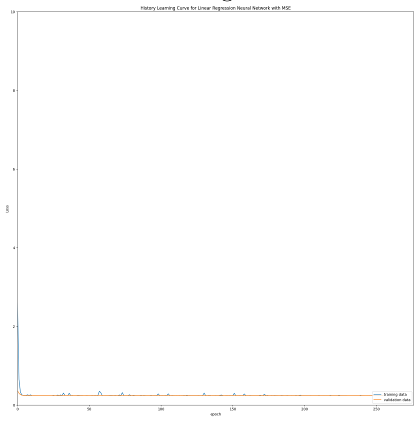


Figure 2: Machine Learning with Linear Regression



Checkpointing and Early Stopping features to prevent overfitting.

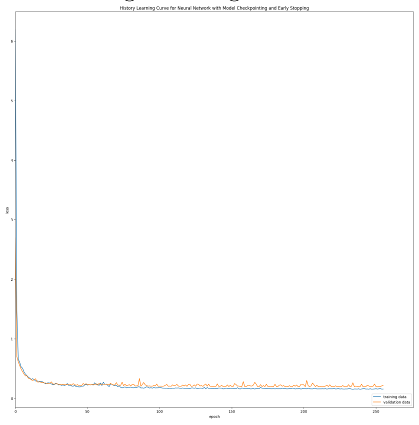
8 Test 2: Feature Removal

The Third Phase of the Project was iterative feature removal. This testing involved checking the importance of each input feature, and the simplifying the dataset to determine if the accuracy can be improved by removing input features.

8.1 Individual Input Field Testing

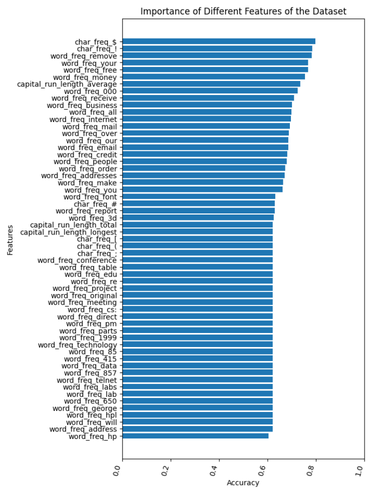
The Third Phase of the Project starts by checking each feature for its importance. This importance is checked by analyzing the accuracy of the data using

Figure 3: Machine Learning with Sigmoid and Model Checkpointing



the single input feature.

Figure 4: Individual Feature Significance

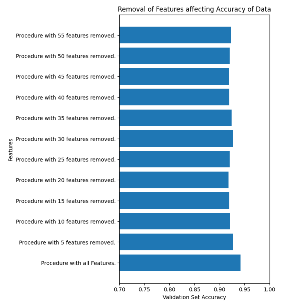


After analysis and the learning, the model returned that the most significant input feature was the frequency of the character "\$." The dollar sign makes a significant character in the list because advertisements for products typically list the price in Dollars. In second place, the character "!" makes the second most important feature. This is perhaps because of the use of exclamation points to make advertisement spam stick out and grab the user's attention.

8.2 Input Feature Removal

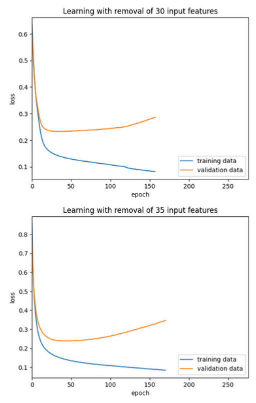
Removing Input Features caused the accuracy of the machine learning to decrease. The machine learning system also became prone to overfitting, due to the decreasing number of input features.

Figure 5: Accuracy for Number of Features Removed



There was a slight increase when the number of input features reached 30-35. This range would eliminate most of the ambiguous keyword input features, such as ”your” and ”you.” These words would have too many meanings to determine spam, but their removal doesn’t make the machine learning any more accurate.

Figure 6: Loss Graphs for 30 and 35 Input Feature Removals



9 Conclusion

The removal of the least important input features helped to improve the accuracy of the system, albeit, minimally.

The neural network, even with a smaller amount of neurons, overfits and generates noise, leading to difficulties when generating data and results. The training ended up being worse as more and more input features were removed from the data set. Fortunately, Early Stopping and Model Checkpointing prevented loss from increasing more.

The loss of the accuracy and performance was unexpected when working on this program. The initial hypothesis was that the removal of the unimportant features would make future models and predictions more accurate, due to the removal of ambiguous word frequency features. However, the loss only increased as more features were removed.

Machine Learning can still be a useful approach to detecting Spam, though this dataset is very limited in its scope. Currently, the SpamBase dataset focuses on keywords and character frequencies to determine if something is spam. New Machine Learning approaches could learn new words, possibly using some kind of new word generation system that develops new vocabulary as time progresses, though that might be difficult.

One example of an improvement from learning new words could be to figure out if a Phishing Spam E-mail has a dangerous link attachment. This could include looking for a URL-Shortening Service URL Attachment and determining if it is dangerous mail. However, this might not work entirely. New words could be added by a centralized Spam Detection Server, like a Protection service, to users, or the Machine Learning System could automatically make a choice to add a word. Care would need to be taken, as Legitimate E-mails that are mistaken as Spam could hinder and harm the user.