Springboard

Karolina Bzdusek

Data Science Course: Capstone Project 1

SMS Spam Detection

1. Exploratory Data Analysis

1.1 Introduction

The aim of this project is to determine whether SMS is spam or not. The dataset is composed by 5,574 English, real and non-enconded messages, tagged according being legitimate (ham) or spam.

The table below lists the provided dataset in different file formats, the amount of samples in each class and the total number of samples.

Application	File format	# Spam	# Ham	Total	Link
General	Plain text	747	4,827	5,574	Link 1
Weka	ARFF	747	4,827	5,574	Link 2

In the table below we can see statistics regarding our dataset. Where spam can have two values (0,1) -> (ham, spam), length is length of characters of each message, num_words is number of words of each message.

	spam	length	num_words
count	5572.000000	5572.000000	5572.000000
mean	0.134063	76.312455	15.374372
std	0.340751	57.079199	11.144851
min	0.000000	1.000000	0.000000
25%	0.000000	33.000000	7.000000
50%	0.000000	58.000000	12.000000
75%	0.000000	116.000000	22.000000
max	1.000000	888.000000	171.000000

As it is a NLP task, the dimensionality is high (each unique word is a feature), therefore some cleaning of the text is needed (e.g. punctuation is possibly not needed, or if the message contains two and more exclamations marks in a row, it makes difference whether it's four or ten; lower and upper case of the word can on the other hand reveal some insight whether it's a spam or not; etc). This is one way how to reduce dimensionality before transforming words to vectors. Another would be to determine features that are the most helpful to recognize spam from ham (e.g. using Linear Regression LASSO regularization), in other words dimension reduction.

1.2 Data Wrangling

As we are dealing with NLP, it is necessary to transform text to the numbers. Firstly, we get rid of the punctuation:

```
import string
translator = str.maketrans('', '', string.punctuation)
df.text = df.text.apply(lambda x: x.translate(translator))
```

Before creating bag of words, we used causal tokenizer TweetTokenizer . Basically, this part is about how to split sentence to the "words" (tokens) or more precise, split to a set of characters from which we can create bag of words. After creating bag of words we

then finally can reach to our desired dataframe - words are features in dataframe. Each NaN value we changed to zero value.

1.3 Data Story Telling

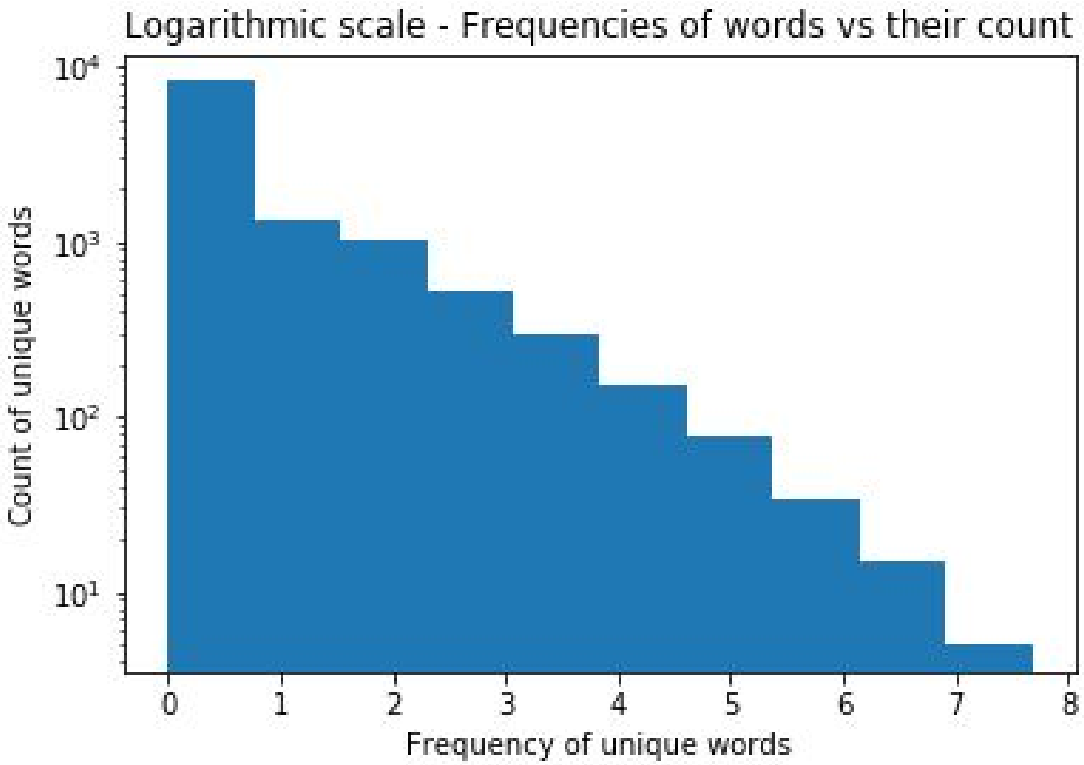
We will look closely on all words (referring as 'words'), then 'spam words' ('spam') and then non-spam words ('ham').

Firstly take look on words that we have in our corpus. For illustration, here is Word Cloud of our messages:

In our corpus we have 11662 unique words, to be more precise 11662 tokens (we will be using words instead of tokens from on). Messages contain emojis, various abbreviations (such '2U'). This 11662 token where counted after our preprocessing.

Unique words statistics table shows us, that half of the words occur only once and only and highest frequency of word is 2159.

Here is visualization of frequencies of words vs. their count in logarithmic scale (as we have a huge amount of word that occur just few time and on the other hand there are few words that are very high in frequency).


Logarithmic scale - Frequencies of words vs their count
Count of unique words
10^4
10^3
10^2
10^1
0
1
2
3
4
5
6
7
8
Frequency of unique words

Now, having better understanding of our words, look on the set of words and its subsets: spam and ham words. We start with comparing their statistics:

Unique words - statistics

	Count of unique words
count	11662.0
mean	7.426513462527868
std	46.47081101779689
min	1.0
25%	1.0
50%	1.0
75%	3.0
max	2159.0

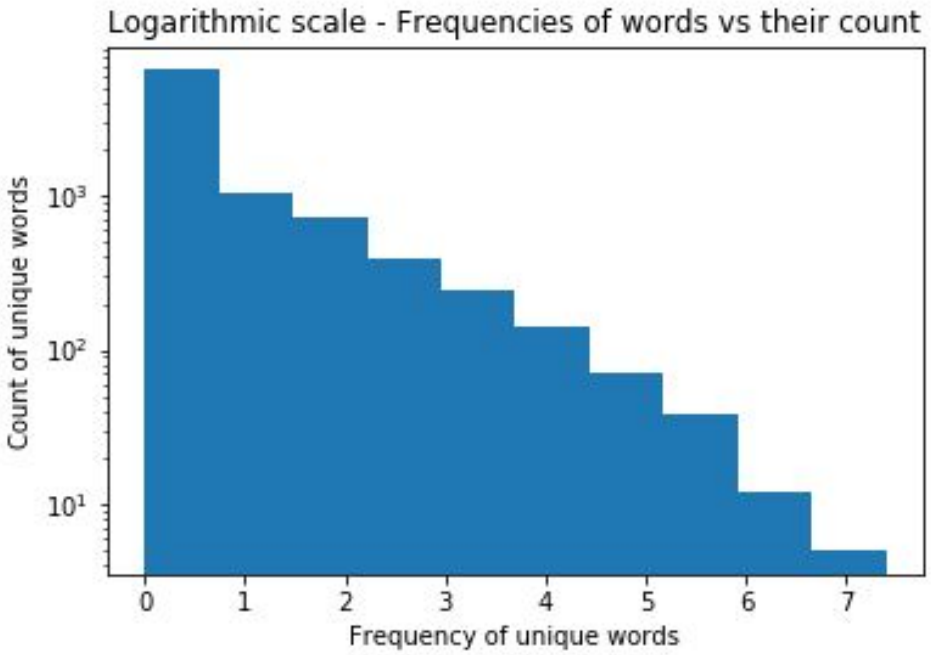
Ham statistics

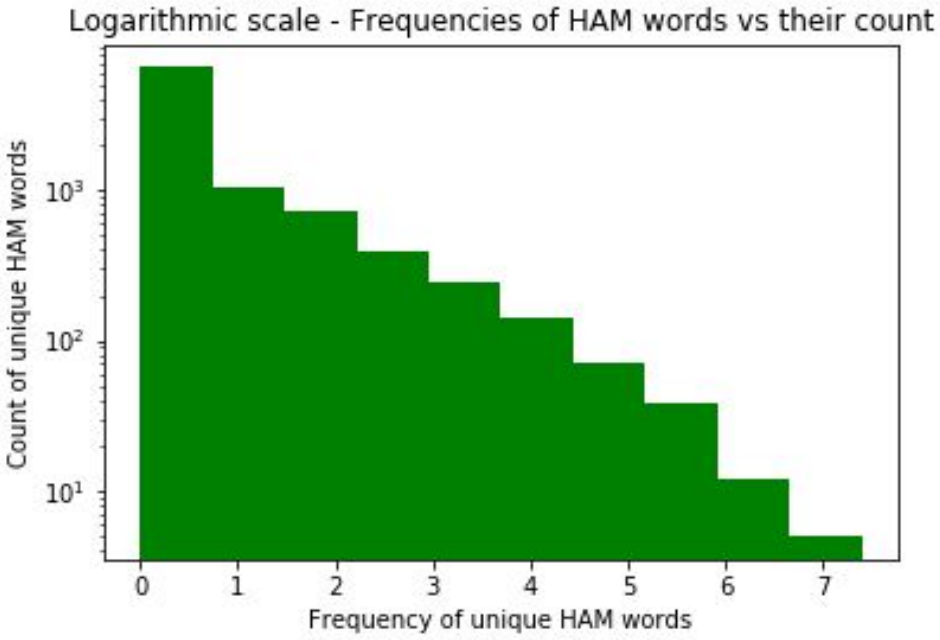
	Count of unique 'ham' words
count	9202.0
mean	7.4186046511627906
std	43.946803580331 81
min	1.0
25%	1.0
50%	1.0
75%	3.0
max	1630.0

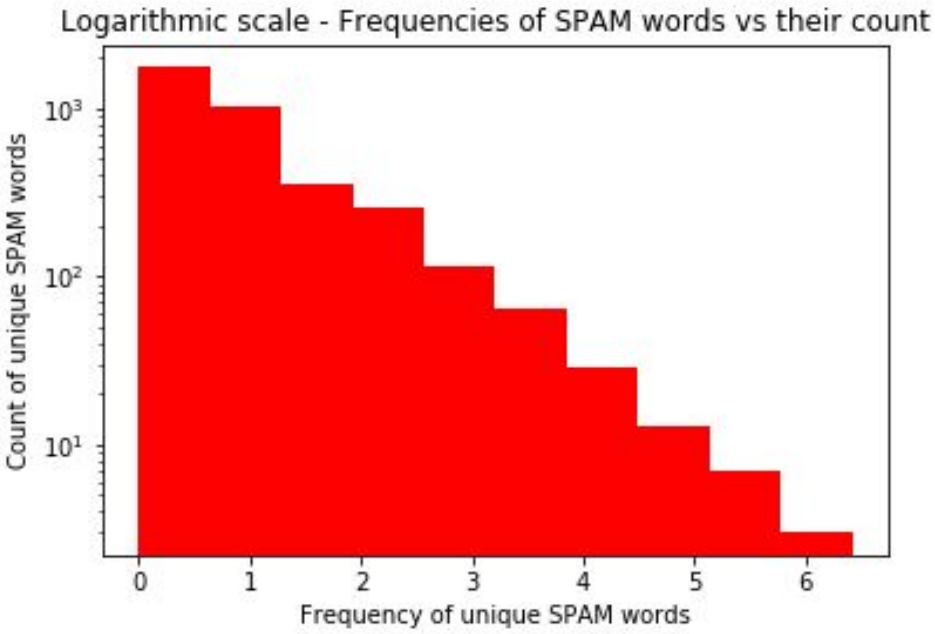
Spam statistics

	Count of unique 'spam' words
count	3624.0
mean	5.061258278145695
std	18.289151763341653
min	1.0
25%	1.0
50%	2.0
75%	3.0
max	611.0

To get better insight, let's look on histograms of frequencies of words, spam, ham versus count of unique words, spam, ham in logarithmic scale.


Logarithmic scale - Frequencies of words vs their count
Count of unique words
10^3
10^2
10^1
0
1
2
3
4
5
6
7
Frequency of unique words


Logarithmic scale - Frequencies of HAM words vs their count
Count of unique HAM words
10^3
10^2
10^1
0
1
2
3
4
5
6
7
Frequency of unique HAM words



The words that are occuring in spam but not in the ham:

['Call', 'won', 'Txt', 'free', '1', 'from', 'week', 'ur', '£', 'text', 'Nokia', 'or', '4', 'mobile', 'who', 'You', 'contact', 'FREE', 'claim', 'Your', 'service', 'prize', 'txt', 'To', 'STOP', 'only', 'reply', 'our', '16']

Thoughts:
After having final model, make similar pictures (or just lists) and compare how important these most frequent words are for spam detection. We have a feeling, they look as good candidate to be important features in predicting spam/ham. In the end we check our intuition.

1.3 Inferential Statistics

In our data set of SMS we have spam and ham messages. We want to find out, whether some features is more important than others in order to predict whether is spam or ham. For the sake of demonstration we will pick just one word ("Call").

Springboard

Firstly, we need to find out whether the CLT (Central Limit Theorem) applies:

1) We have 5572 messages, so N is large enough -- check
2) We assume that independent condition is satisfied as well

We state our null and alternative hypothesis and then we test them. We set $\boldsymbol{\alpha} = 0.5$

$\mathbf{H_0}$: $\mu_{spam\text{-}call} = \mu_{ham\text{-}call}$

$\mathbf{H_A}$: $\mu_{spam\text{-}call} \neq \mu_{ham\text{-}call}$

In this Jupyter Notebook you can find details about testing our hypothesis. We here concluded our findings.

Our 95% confidence interval is <0.1803, 0.2435> . And therefore we can't reject null hypothesis. This is quite expected result -- wrod 'call' is widely used in 'ham' communication as in the 'spam'. So word 'call' alone is not sufficient to decide whether message is a spam or ham.