Bayesian Spam classifier

Prasanth Gudiwada

Abstract:

With the surge of spam content on smartphones, there is a necessity to build a spam filter for text messages as accurately as possible. Traditional email spam filters cannot be used for text messages as most of the emails contain full sentences, common-source, and a subject line whereas, text messages tend to contain incomplete sentences and are misspelled. This project seeks to build a spam classifier using Bayes theorem with other variations to predict the nature of an incoming text message.

1.Introduction:

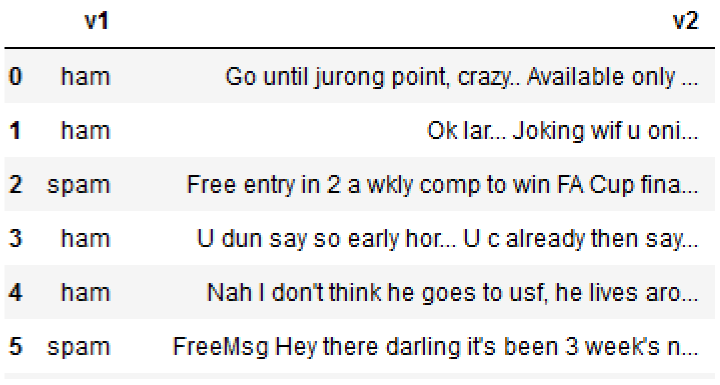
Spam content is increasing more and more in text messages with recent developments in technology. A recent survey from Techjury found that text messages have 209% higher response rates than emails, and 23 billion texts are sent globally every day. It is important to detect spam messages not only for personal convenience but also for security. Being able to remove texts with potential viruses or malicious intent of any kind is necessary on both individual user levels and larger scales. Email service providers use a spam filter which automatically filters out highly potential spam emails but, such advanced filters are not available in text messaging.

Many machine learning techniques such as KNN (Clustering), Random forest, SVM, Gradient boosting, Naïve Bayes, etc. can be used to classify the text messages. All the methods except Naïve Bayes convert the text messages into word vectors and use the words as features. For example, the KNN algorithm considers the Euclidian distance between two texts as a parameter, and the Random forest generates multiple decision trees based on the word frequencies to classify the text message. On the other hand, Naïve Bayes uses the word frequency to calculate the individual probability of words to estimate the probability of a text being spam, which makes it powerful for the small amounts of data.

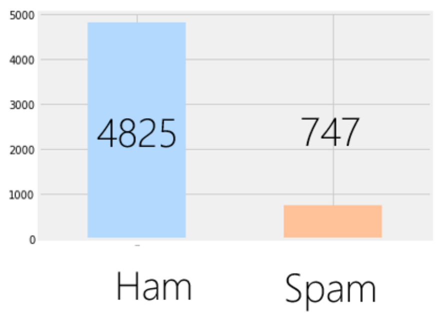
This project analyzes a sample data set of text messages which are prelabeled and implements Bayes theorem in determining the spam nature of any text message. The organization of this project will be as follows, Section 2 describes the dataset and cleaning process; Section 3 shows the descriptive statistics of the data; Section 4 contains the description of the methodology using Bayes theorem. Results of the project and future extensions are described in Section 5 and Section 6 respectively.

2.0.Data:

The dataset for this project is obtained from the UCI machine learning library, which consists of 5574 text messages.



	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...

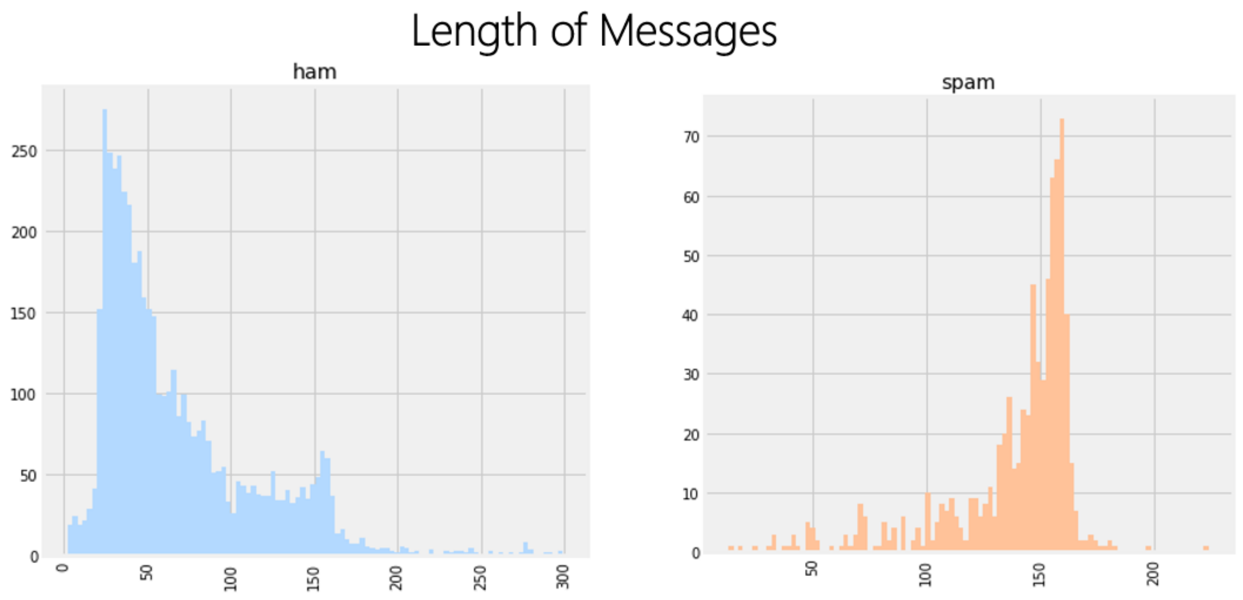


2.1.Cleaning:

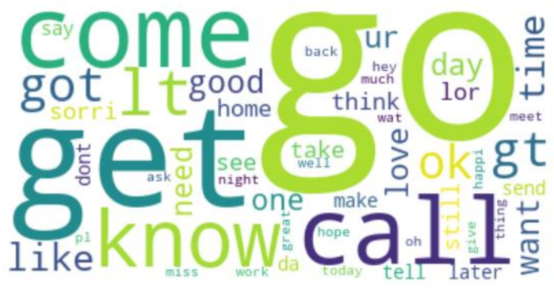
One of the first steps in working with text data is to pre-process it. The dataset contains raw text messages which is a mixture of words, special characters, emojis, and numbers. It is an essential step for the analysis to remove the noise in data. The following procedure is used to clean the text messages data,

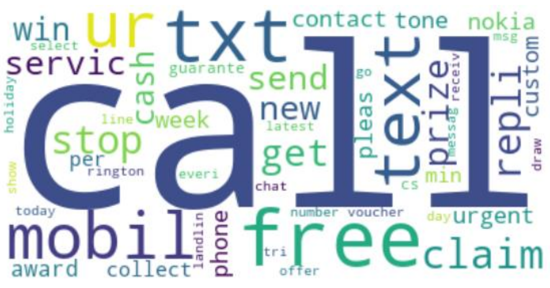
- **Case**: Alphabets in the messages are converted into small case, because the case of word does not alter the meaning, intent of the text.
- **Tokenization**: All the text messages are split into a vector of words, and removed punctuations, numbers.
- **Small words**: Words with less than two characters are removed.
- **Stop words:** Stop words are generally the most common words in a language such as, 'is', 'are', 'an', 'the' , 'as' etc. These words do not have any particular meaning, and their frequency would not aid in estimating the probability of a word being spam.
- **Stemming:** Stemming is an algorithm which detects the core meaning of a word and replaces all the similar words with a single word. For example, words 'go', 'going' , 'goes' have the same meaning, and stemming converts all the words into 'go'.

3.Exploratory data analysis:



From the histogram of the length of text messages, it is evident that the length for ham messages follows a unimodal right skewed distribution whereas, that of spam messages follow a left skewed distribution. Hence it is safe to assume that the spam messages tend to be longer than the ham messages.





HAM **SPAM**

Word cloud is a novelty visual representation of text data to visualize the word content of the text. The size of each plotted word is the representative of it' frequency. 'Call' is the most frequently word used in spam messages, followed by 'Free', and 'Text'. On the other hand, 'Go' is the most used word in ham messages. Ham messages also have high frequency of 'Get', 'Know', 'Come', and 'Call'.

4.Methodology:

Words in text messages have different probabilities of occurring in spam texts and ham texts. For example, most of the spam messages contain the word, "Free" but only a few of the ham messages have it. The approach for the project is to make use of these differences in frequencies and calculate the probability of whole text being spam or ham using Bayes theorem.

If whole sentences in the text are taken as an entity, there is a high chance that the sentence will not be present in the training set. This will result in giving a zero probability despite the spam nature of the text. Hence, words are taken as attributes to calculate the probability.

Bayes theorem can be expressed as the followed mathematical equation,

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)}$$

Where,

- A and B are events and P(B)≠0
- P(A|B) is a conditional probability: the likelihood of event A occurring given that B is true
- P(B|A) is a conditional probability: the likelihood of event B occurring given that A is true
- P(A) is a marginal probability: The probability of event A independent of event B
- P(B) is a marginal probability: The probability of event B independent of event A

This theorem can be applied to the case when event A is considered as the word being spam, and event B as the occurrence of the word. Then,

$$P(Spam/Word) = \frac{P(Word/Spam) * P(Spam)}{P(Word)}$$

P(Spam/Word) is the probability that the word is spam given a word in the message.

As any word can only be either ham or spam (i.e. a binary variable), it can be rewritten as following,

$$P(Spam/Word) = \frac{P(Word/Spam).P(Spam)}{P(Word/Spam).P(Spam) + P(Word/Ham).P(Ham)}$$

The probability, P(Word/Spam) can be obtained by the ratio of total number of spam messages containing the word and frequency of all messages containing the word.

To calculate the above-mentioned probability, the probability of spam is required. If it is assumed that every incoming message is equally likely to be a spam or a ham, P(Spam)=P(Ham)=0.5. This assumption permits simplifying the formula to:

$$P(Spam/Word) = \frac{P(Word/Spam)}{P(Word/Spam) + P(Word/Ham)}$$

According to this formula, if one of the words in the test dataset was never encountered in the training phase, the probability of word being spam will be 0/0. There are two ways to avoid this issue, 1. To remove all such words from the analysis. 2. Use Laplace smoothing to add a pseudo-count. Removing words is not preferable as the remaining words in the text will solely drive the probability.

Laplace Smoothing:

A pseudo-count is an amount added to the number of observed cases to change the expected probability in a model of those data, when not known to be zero. A pseudo-count of value k weighs into the posterior distribution similarly to each category having an additional count of k.

$$P(Spam/Word) = \frac{P(Word/Spam) + k}{P(Word/Spam) + P(Word/Ham) + 2 * k}$$

Corrected Probability:

The words that were encountered only a few times during the learning phase cause a problem, because it would be an error to trust blindly the information they provide. A solution for this issue is to use a corrected probability,

$$P^{'}(Spam/Word) = \frac{strength * P(Spam) + n * P(Spam/Word)}{strength + n}$$

The strength here is the strength given to background information about incoming spam, and n is the number of occurrences of this word during the learning phase.

Combined Probability:

If it is assumed that the words in the text messages are conditionally independent i.e. the probability of a word being spam is independent of that of another word in the same text, the following combined probability can be used to determine the probability of a text message being spam.

$$P(Spam/Message) = \frac{P(Spam/Word_1) * P(Spam/Word_2) ... * P(Spam/Word_n)}{P(Spam/Word_1) * P(Spam/Word_2) ... * P(Spam/Word_n) + (1 - P(Spam/Word_1)) * (1 - P(Spam/Word_2)) ... * (1 - P(Spam/Word_n))}$$

Model parameters:

The entire dataset was split into 75% for training and 25% for testing, and a model was built on training data using the above-mentioned methodology with strength parameter as 1 (the learned corpus must contain more than 1 message with the word to put more confidence in the probability than in the default value), and Laplace smoothing pseudo-count as 1 to calculate the probability of a text message being spam. This model was used to predict the nature of text messages in the test dataset with a threshold probability of 0.5. i.e. a text message is classified as spam if the probability obtained is greater than 0.5.

5.Results:

The confusion matrix and model statistics are mentioned as following,

Label	Spam	Ham
Spam	**124**	**46**
Ham	**14**	**1209**

- Overall Accuracy : 95.69%
- Sensitivity of Spam: 0.74
- Sensitivity of Ham : 0.99
- Type I error : 0.03
- Type II error : 0.08

The model predicts ham messages (0.99) significantly more accurate than spam messages (0.74). This significant difference is due to imbalance in the dataset considered. Addition of spam data would certainly increase the sensitivity of Spam, and the total accuracy of the model.

For this scenario, type II error is more important than type I as receiving spam messages is preferable than filtering out ham messages. Type II error can be reduced by increasing the threshold probability of spam however, it will increase type I error, and reduce the overall accuracy of the model.

6.Future extensions:

Boosting: Bayesian poisoning is one of the most common issues faced by all Bayesian models. Spammers use this technique by replacing highly probable spam words with more frequent ham words or adding more sentences which has ham words. This would result in reducing the overall spam probability of the message, which will be predicted as ham by the model.
One of the ways to avoid this issue is to use boosting. Boosting would induce weights for the probabilities of words in predicting the overall probability of the text.

Markovian Models: The major assumption in building this model was that all the conditional probabilities of words in the text are independent. This assumption is not valid in the real-world cases, as words in the text are not independent. For example, the probability of occurrence of an adjective increase in presence of a noun, and text with words ‘No offer’ is not in the same context as another text with words ‘No’, and ‘Offer’.
A Markovian model adds the relative transition probabilities that given one word and predicts the next word. It is based on the theory of Markov chains, which assumes that the probability of given word depends only on the previous word. This theory can be used to predict the probability of text by considering phrases or entire sentences instead of individual words.

7.References:

https://archive.ics.uci.edu/ml/datasets/sms+spam+collection
https://techjury.net/stats-about/sms-marketing-statistics/#gref
https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering
https://en.wikipedia.org/wiki/Markovian_discrimination
https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python/
https://www.virusbulletin.com/virusbulletin/2006/02/does-bayesian-poisoning-exist
https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf
https://en.wikipedia.org/wiki/Additive_smoothing