Detecting spam text messages with Naïve Bayes and Random Forest methods

Description and motivation of the problem and project

The project aims to use two machine-learning (ML) methods to classify whether short-message-service (SMS) messages are spam or not.

As mobile-phone usage and messaging platforms become more prevalent globally, spam-detection is a domain area likely to see increasing focus as telecommunication companies and other messaging service providers combat spam affecting their users.

The models within this approach may be adapted and applied to work with different messaging methods such as e-mail, or private messages on platforms like Twitter.

The first method of this project will train a Naïve Bayes classifier model, a commonly used approach for text-classification problems (Abu-Nimeh et al., 2007). The second method will employ a Random Forest approach, another common approach for spam-classification (Kothapally, Reddy & Kakulapati, 2021) and one which has proven to be effective and successful at the task

Hypothesis statement

A literature review of the area of the project indicates that a Random Forest approach is likely to yield a high-accuracy, and possibly outperform the Naïve Bayes model as seen by Sjarif et al. (2007).

Naïve Bayes models are more dependent on the quality and quantity of the data set, which may prove a limiting factor of this method, where we will aim to use a balanced number of ham and spam messages in training. A NB model would yield more accuracy with greater data.

Overall, the expectation is that the Random Forest model will provide a greater accuracy in classifying spam texts over the Naïve Bayes model.

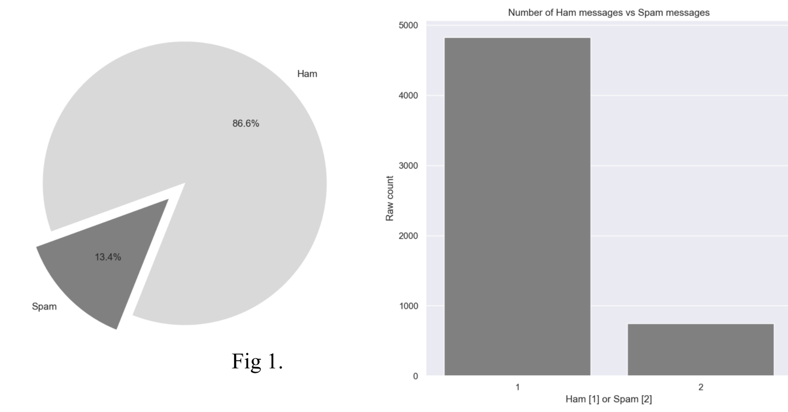

Fig 1.

Initial analysis of the dataset and description of the pre-processing methods

The dataset used, Spam SMS data set, was uploaded on www.kaggle.com as a comma-separated-value (CSV) file which contains over 5,000 rows and two columns. The columns were "category" and "message", with the second column containing a SMS message string, while the first column labelled the contents of the message as "spam" or "ham".

In order to apply and train machine learning models onto the dataset, it was required to pre-process the data to first clean the messages.

Preparing the messages prior to creation of a dictionary required the following steps:

I. Lowercasing all the text.
II. Replace integers with a "number" string.
III. Remove special characters and punctuation.
IV. Removal of common stop words.
V. Removal of whitespace.

Once a dictionary has been established, the messages must be represented as feature vectors over the dictionary words such that a set of training and test features as well as labels are created. This was portion of the project was conducted with Python.

As our dataset is in a single CSV file, the structure of the training and test feature files were such that each line of the respective files is a triplet of (row index, word index, frequency).

Analysing the initial dictionary, it became apparent that spam text messages contain more numbers within the texts than those in ham texts. This could be due to spam texts trying to lure victims with monetary incentives or requesting the victim to contact a number in order to satisfy their own aims.

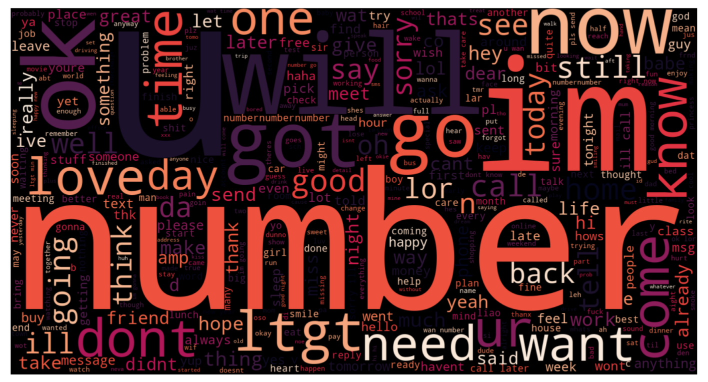

Fig 4.

The terms “call” and “free” were also among the top five most frequent words within the text messages labelled as spam, as well as other action words such as reply, txt, text, and stop (see figs. 2-4).

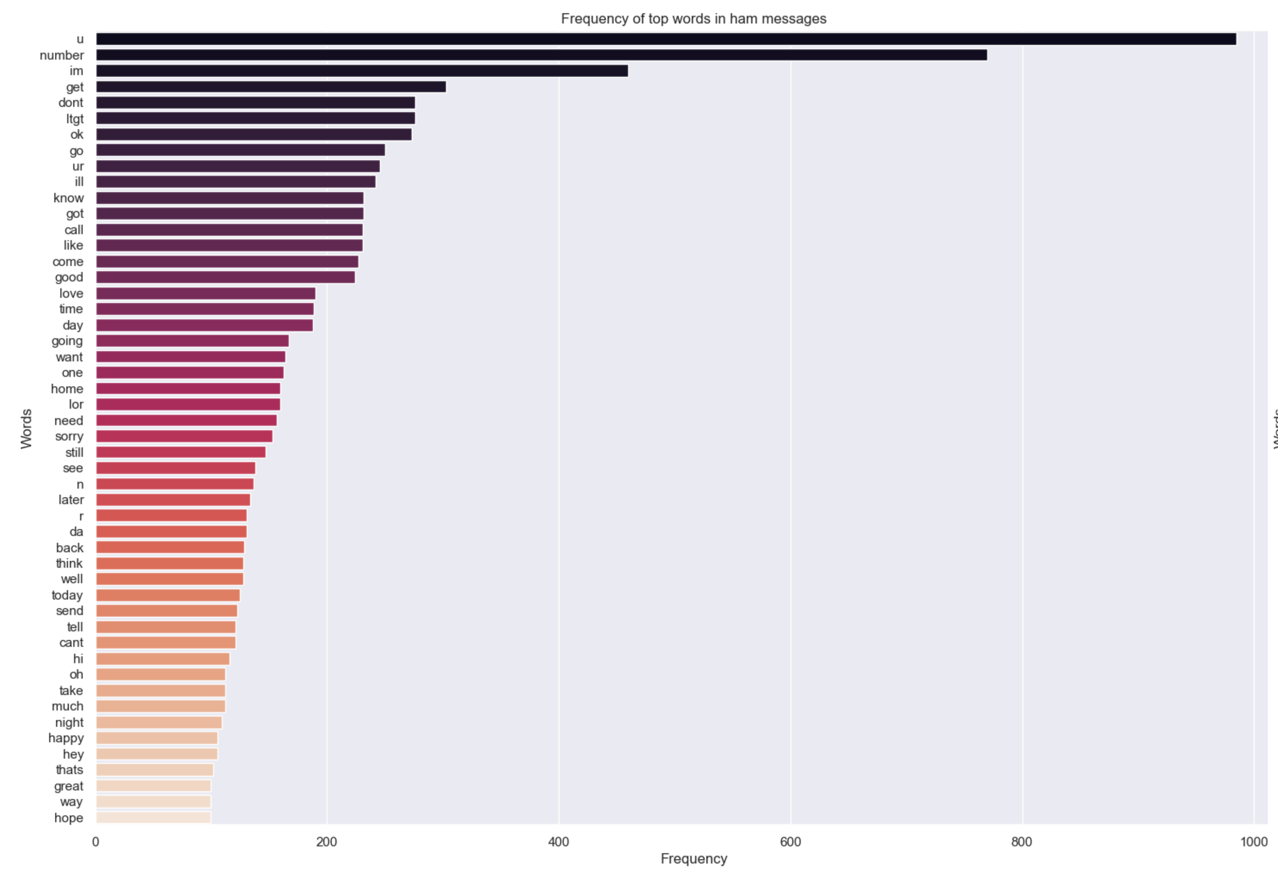

Fig 2.

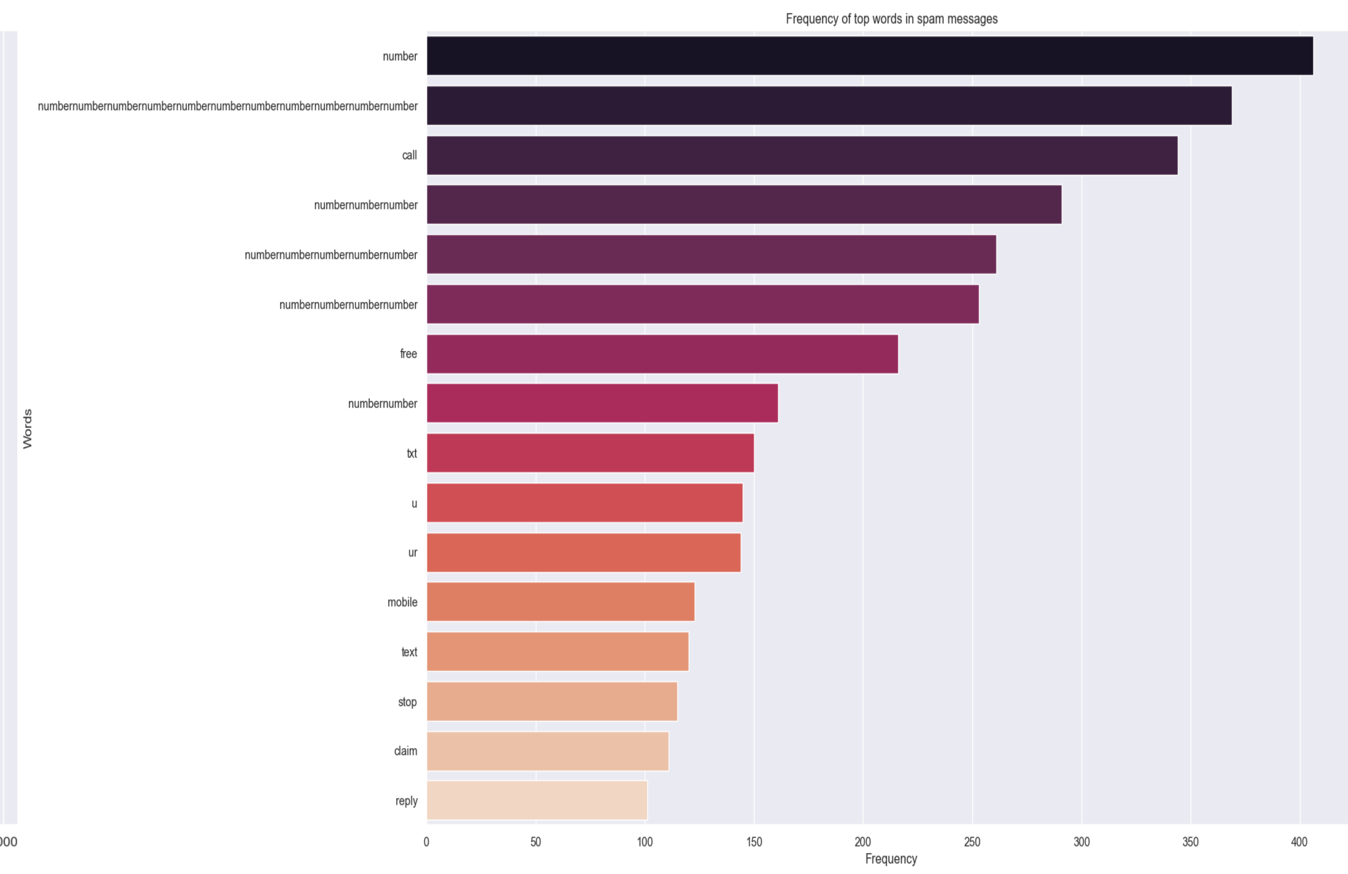

Fig 3.

The Naïve Bayes classification approach

Naive Bayes is a probabilistic machine learning model that is commonly used for classification tasks. They are based on the concept of using Bayes' theorem to predict the likelihood of an event occurring given the occurrence of certain other events. In a Naive Bayes model, the assumption is made that the presence or absence of a particular feature of the data is independent of the presence or absence of any other feature, given the class variable. These models are known for their simplicity and efficiency, as they can be trained and evaluated quickly even on large datasets. They are often used in text classification and spam filtering (Rusland et al. 2017).

Advantages

- Simple and easy to implementation given correctly structured datasets.
- Efficient and fast, only needing to calculate probabilities for each class, rather than trying to fit a complex model to the data.
- Are robust to missing data, as the model can still make predictions even if some of the features are missing.
- Resistant to overfitting, method’s simplicity helps prevent it from learning noise in the data.
- Large body of knowledge and resources available.

Disadvantages

- Simplicity may also lead to probabilities not seen in the real-world.
- Can perform poorly when the data is highly imbalanced, as the model can become biased to certain classes with greater data.
- Based on predictions rather than certainties
- While small datasets can still yield results, best results often observed with use of a larger corpus of data.
- Can be sensitive to the choice of prior probabilities as model relies on these initial estimates.

The Random Forest classification approach

Random forest models are an ensemble learning method for classification and regression tasks. They work by building a large number of decision trees, each of which is trained on a random subset of the data, and then combining the predictions made by each tree to make a final prediction. The idea behind this approach is that each tree will make slightly different predictions due to the different data it was trained on, and by combining the predictions of all of the trees, the model is able to make more accurate predictions overall. They are widely used, including for spam-detection, as well within a host of industries such as medicine, finance, and insurance.

Advantages

- Highly accurate and performant on a wide range of tasks, including classification and regression.
- Resistant to overfitting with low risk of learning noise in the data.
- Able to handle large and complex datasets
- Capable of handling missing data and are not sensitive to the scaling of the features.
- Straight forward to implement and require little tuning of hyperparameters.

Disadvantages

- Can be more costly than other methods to implement in terms of computer resources dependent on the number of trees.
- Requires a logical and sensible number of trees within the model to avoid over or underfitting.
- Simple linear relationships between trees may stumble when working with a large multitude of complex features within a dataset.

Methodology

I. Read and store training features and labels in matrices.
II. Calculate probability of each token in spam and ham messages.
III. Calculate probability that a message is spam using the ratio of spam to total messages in training set.
IV. Calculate the log probability of each token in spam and ham messages for the test set.
V. Compare log probabilities of spam and ham for each message in the test set to determine classification.
VI. Compare classifications with test labels to determine accuracy rate.
VII. Train a random forest model using the *TreeBagger* function and the training data and labels.
VIII. Use the predict method of the *TreeBagger* object to generate predictions for the test set.
IX. Convert predicted labels and test labels to categorical variables.
X. Compute confusion matrix using the *confusionmat* function.
XI. Calculate accuracy by dividing the sum of the diagonal elements of the confusion matrix by the total number of elements in the matrix.

For the Naïve Bayes model, a trial and error approach highlighted a kernel smoothing factor of +0.5 yielded the highest accuracy (see fig. 5).

Smoothing Factor	Error	Accuracy
1	49.10%	50.90%
0.5	**48.60%**	**51.40%**
1.5	49.55%	50.45%
5	**51.14%**	**48.87%**

Fig 5.

The Random Forest model employed 100 decision trees, and yielded a high percentage accuracy on the test set as seen in the below confusion matrix (see figs. 6 & 7).

	PP	NN
P	220	10
N	2	212

Fig 6.

	PP	NN
P	99%	5%
N	1%	95%

Fig 7.

Analysis and evaluation of results

The Random Forest model had a testing accuracy considerably higher than that of the Naive Bayes model, with the respective models exhibiting accuracy scores of 97% and 51% respectively.

There are several factors which may have contributed to this significant difference in scores between the two models.

Firstly, when conducting the pre-processing of the raw dataset, a decision was made to reduce integer values within text messages to the string "number". As a result of this, if there was present a series of integers such that a digit a number bigger than a single-digit number was represented, it would come in the form "numbernumbernumber..." for the number of individual numbers present in the whole number. This is likely to have led to some bias within the Naive Bayes model during the training phase.

This is because as initial analysis highlighted, messages containing spam contained groups of integers possibly referencing monetary values or telephone numbers, while individual numbers were also shown as present.

To improve on the model's accuracy then, it may have been more correct during the pre-processing to replace integers between 0-9 with their word form such as "zero" "one", and decoupling the result such that 123 would be represented as "one two three" instead of "numbernumbernumber" or "onetwothree".

Further, the split of spam and ham values in the overall corpus of messages within the dataset was such that spam messages only accounted for 13.4% of the overall body of messages. This led to a desire to split the dataset such that there would be an equal number of spam and ham messages in both the training and test sets, significantly reducing the number of messages which were finally used.

The models would likely have benefitted from further tuning, with the help of a greater number of evaluation methods such as an F1 score.

Lessons learned and future work

- More critically assess the impact of decisions when conducting pre-processing.
- Aim to source a greater volume of data which contains a higher proportion of the target (see Fig 1.)
- Make greater use of evaluation techniques such as F1 scores and recall scores.
- Attempt to use other ML and statistical methods to improve results
- Compare usage of Bernoulli or Multinomial document/message models in performance
- Employ the models on the unused ham messages, as a further model test

References

- Abu-Nimeh, Saeed & Nappa, Dario & Wang, Xinlei & Nair, Suku. (2007). A comparison of machine learning techniques for phishing detection. ACM International Conference Proceeding Series. 269. 60-69. 10.1145/1299015.1299021.
- Kothapally, Nithesh Reddy & Kakulapati, Vijayalakshmi. (2021). Classification of Spam Messages using Random Forest Algorithm. Xi'an Dianzi Keji Daxue Xuebao/Journal of Xidian University. 15. 495-505. 10.37896/jxu15.8/048.
- Nilam Nur Amir Sjarif, Nurulhuda Firdaus Mohd Azmi, Suriayati Chuprat, Haslina Md Sarkan, Yazriwati Yahya, Suriani Mohd Sam, SMS Spam Message Detection using Term Frequency-Inverse Document Frequency and Random Forest Algorithm, Procedia Computer Science, Volume 161, 2019, 509-515, https://doi.org/10.1016/j.procs.2019.11.150.
- Rusland, N.F., Wahid, N., Kasim, S. & Hafit, H. 2017, "Analysis of Naïve Bayes Algorithm for Email Spam Filtering across Multiple Datasets", IOP conference series. Materials Science and Engineering, vol. 226, no. 1.