Spam Detection in SMS Messages

Ali Malenchik

Bellevue University

DSC680: Applied Data Science

Prof. Catherine Williams

September 26, 2021

Abstract

Spam detection is a valuable tool used to detect and prevent unsolicited and unwanted messages. SMS spam typically involves bulk messaging for commercial advertising or phishing links. Since sending spam is relatively cheap for spammers, it's an easily abused method of communicating business interest. Many recipients believe spam to be annoying or a violation of privacy, and consequently a method of detecting and reducing spam is a meaningful way to improve the user experience.

The goal of this project was to develop a classification algorithm that accurately predicts whether an SMS message is spam or not (i.e. "ham"). Over five thousand SMS messages collected for mobile phone spam research were analyzed to identify patterns and characteristics of spam messages. Data was cleaned and prepared using Natural Language Processing (NLP), then used for training multiple classification algorithms including Random Forest, Naïve Bayes, Logistic Regression, SVM, and Decision Tree. Each model was evaluated by making predictions on reserved data in order to determine the "best" performing.

Based on the evaluation results, the Random Forest model demonstrated the highest precision and therefore was selected as the best performing algorithm suitable for classifying spam.

Introduction

Background

Spam detection is a vital task for enhancing the communication experience. It prevents the annoyance and frustration of receiving an overwhelming number of unwanted messages. As spammers have branched out from using email to other approaches such as robo-calls and SMS messages, the tools available for spam detection must adapt and target each method in order to be effective and accurate. Machine learning can perform this task by utilizing historical messages to predict whether a message can be categorized as spam or "ham."

Problem Statement

Can a machine learning algorithm accurately predict whether an SMS message is spam?

Scope/Assumptions

This project will utilize Python to explore several binary classification algorithms using collected SMS data to predict whether the message is spam. In order to perform feature extraction, natural language processing will be utilized to tokenize and vectorize the message data. The models will be evaluated using multiple metrics in order to select the best performing classifier. Due to time restrictions, hyperparameter tuning and model optimization will not be in scope for this project.

Methods

Data Source

The dataset used for this project is a collection of SMS messages that have been collected for mobile phone spam research, found on Kaggle. The dataset is provided in CSV format and contains two columns: Category and Message. The Message column contains the raw SMS message, while the Category column is the binary target variable classifying the message as spam or not. Possible Category values are “spam” or “ham” (not spam). The messages were gathered from multiple sources:

- The Grumbletext website, which is a UK forum for cell phone users to report received spam messages
- NUS SMS Corpus from the Department of Computer Science at the National University of Singapore
- A PhD Thesis created by Caroline Tag
- SMS Spam Corpus v.0.1 Big

Data Import & Cleansing

Comma-separated data was loaded into a data frame using Pandas read_csv function. The data frame consisted of 5572 records at time of import. Multiple cleaning and pre-processing steps were taken. First, the data was analyzed for null values. No null values were found and therefore no steps were taken to remove or impute missing data. The data was then analyzed for imbalanced target classes. Though it was found that the dataset was imbalanced, it was decided not under- or over-sample to avoid the loss of valuable information and maintain the integrity of the SMS message content patterns. A new feature, “num_chars”, was added to the data frame by calculating the number of characters in each message. Another feature, “num_words”, was added

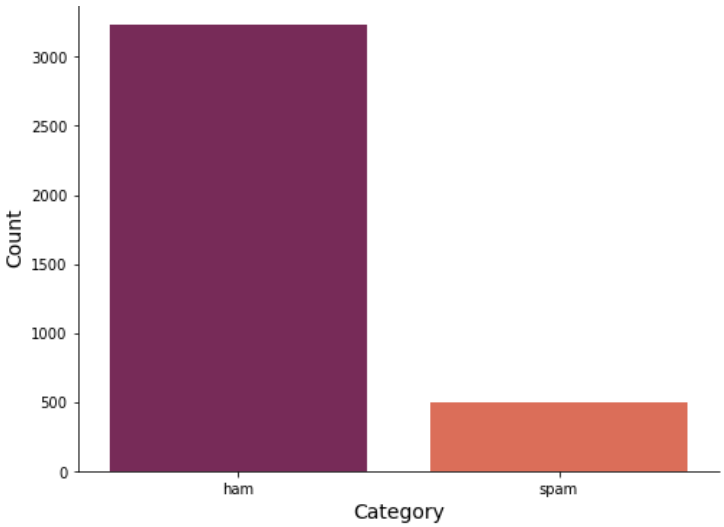
to the data frame by calculating the number of words in each message. The target variable, "Category", was transformed into binary values for use in modeling. Values of "ham" were reassigned to 0, and values of "spam" were reassigned to 1. The data was split into 33% test, 67% train using sklearn's test_train_split function. After the split, the resulting training dataset contained 3733 records. The test dataset contained 1839 records.

Exploratory Analysis

In the next step, the cleaned training data frame was used to perform exploratory data analysis. First, a count plot of the distribution of SMS message category was created using Seaborn's countplot function. The count plot demonstrates that the imbalanced classes persist in the training dataset, with "ham" (non-spam) messages accounting for more than 6 times the amount of spam messages.

Figure 1

Count Plot of SMS Message Category for Training Dataset

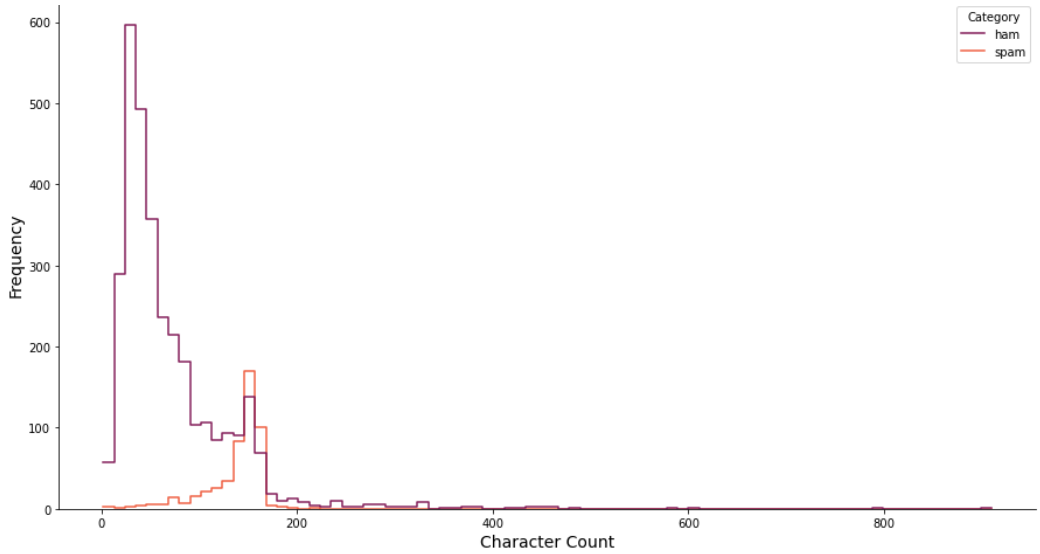


Next, a step plot of the newly calculated character count was created using Seaborn's histplot function and distinguishing between message category using different colors. From this plot it can be gathered that "ham" messages tend to be on the shorter side in terms of character

count, with the highest frequency under 50 characters. Converseley, spam messages peak frequency closer to 150 and rarely contain less than 100 characters.

Figure 2

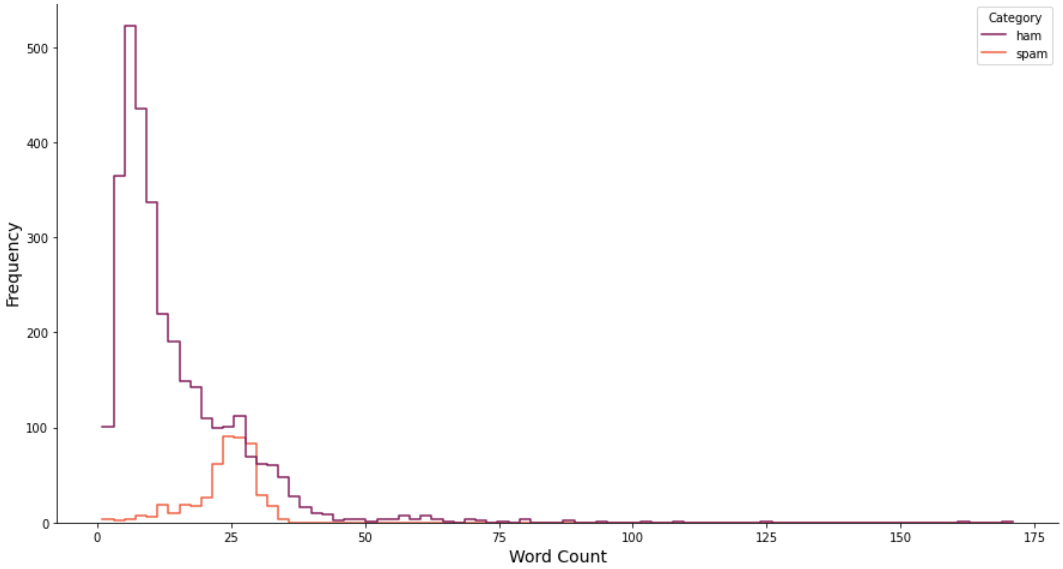
Step Plot of SMS Message Character Count by Category for Training Dataset



Similarly, the step plot of word count displayed comparable trends:

Figure 3

Step Plot of SMS Message Word Count by Category for Training Dataset



In order to compare descriptive statistics for the number of words and number of characters, box plots were created. From these plots we can observe that though the range and IQR of character & word count is narrower for spam messages, the median and quartile values are generally higher.

Figure 4

Box Plot of SMS Message Character Count by Category for Training Dataset

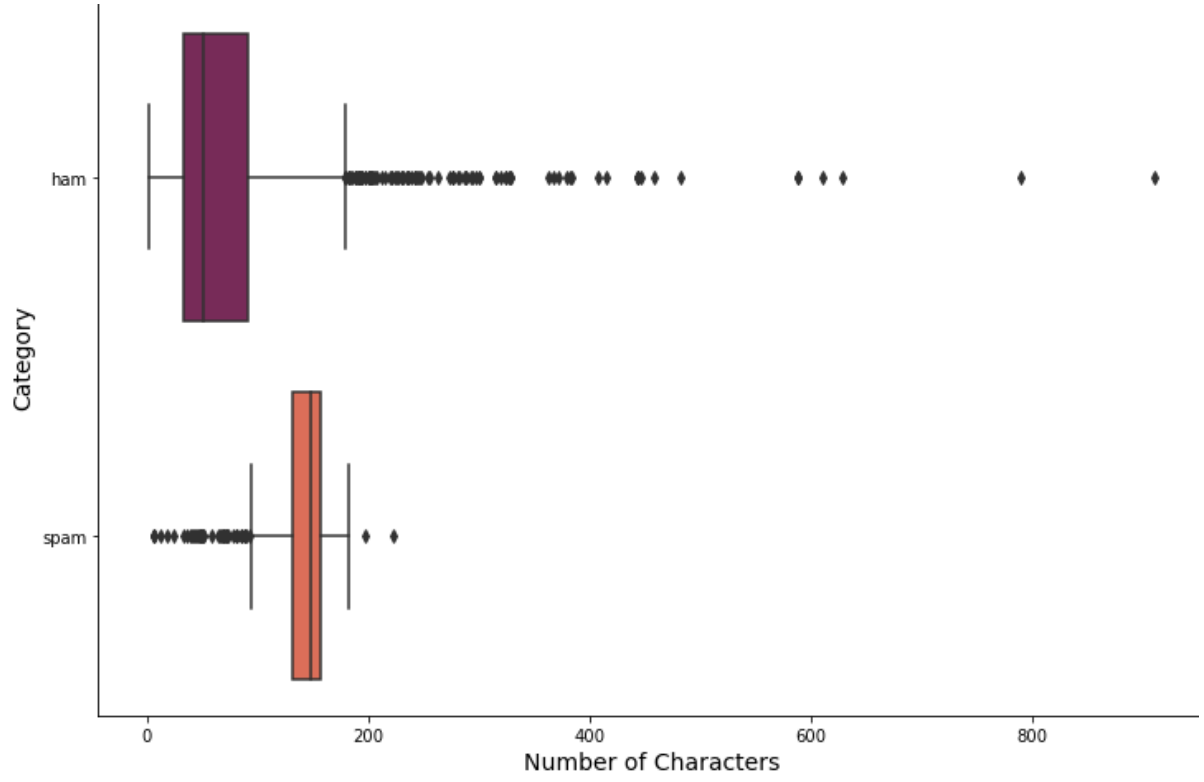
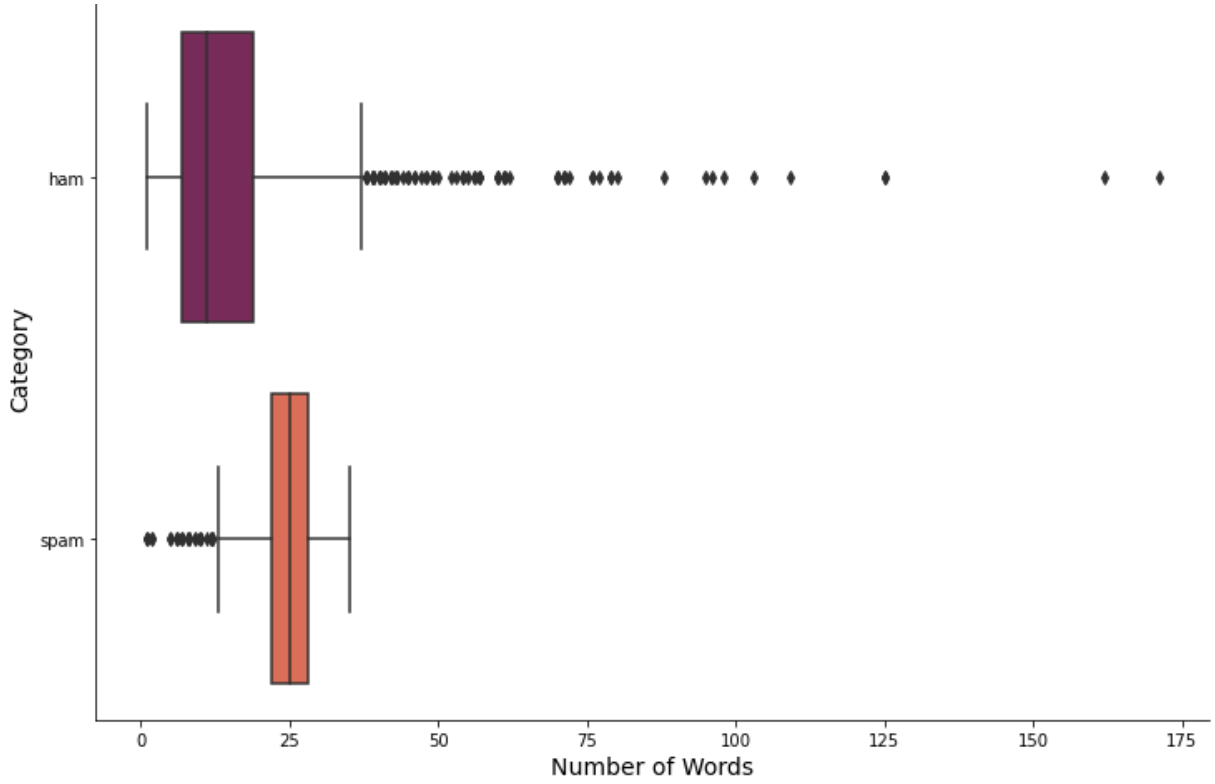


Figure 5

Box Plot of SMS Message Word Count by Category for Training Dataset



Finally, word clouds for ham messages and spam messages were created using the wordcloud library and removing stop words. From the word clouds we can observe that there is a noticeable difference in the content of ham versus spam messages. The most common words from ham messages are “u”, “will”, “lt”, “know”, “gt”, “ok”, “go”, and “come”. On the other hand, the most common words from spam messages are “free”, “call”, “txt”, “text”, “now”, and “mobile”.

Figure 6

Word Cloud for Ham SMS Messages

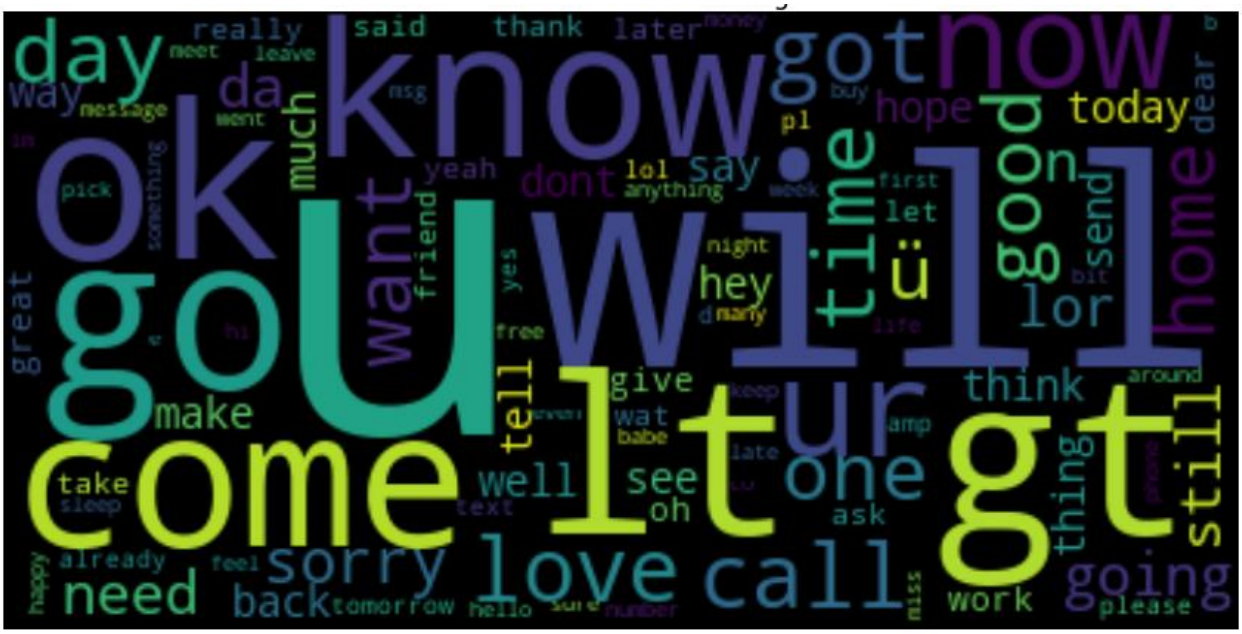


Figure 7

Word Cloud for Spam SMS Messages

Feature Selection

In order to prepare the data for modeling, the Category attribute was encoded to 0s and 1s using sklearn's LabelEncoder() function. Messages having the category of "ham" were reassigned as 0, and messages having the category of "spam" were reassigned to 1. Additionally, message data was cleaned by removing non-alphanumeric tokens, removing stop words, and converting to lower case. The message data was tokenized and lemmatized using nltk. The clean training data was then split into two: a data frame containing clean message data (X_train), and a series containing the encoded category (y_train).

Figure 8

Independent Variable (X_train)

	Clean_Message
4084	good afternoon love good see word ym tm smart ...
3517	well give co said didn t one nighters persever...
4808	private account statement show unredeemed bonu...
232	dear going rubber place
3715	oh will paid outstanding one commercial hasbro...
...	...
3686	wake lt gt morning
4068	contacted dating service someone know find cal...
5507	want inside every night
5396	want custom officer discount oh
2235	room number wan na make sure knocking right door

3733 rows × 1 columns

Figure 9

Dependent Variable (y_train)

```
4084    0
3517    0
4808    1
232     0
3715    0
       ..
3686    0
4068    1
5507    0
5396    0
2235    0
Name: Category_Encoded, Length: 3733, dtype: int32
```

After this split, count vectorization was performed using sklearn's CountVectorizer. This provides the features for model training.

Figure 10

Independent Variable after Count Vectorization (X_train)

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Model Deployment

Five models were trained using the sklearn library: Random Forest Classifier, Naïve Bayes Classifier, Logistic Regression Classifier, SVM Classifier, and Decision Tree Classifier. Due to time restrictions, hyperparameter tuning and optimization were not performed. Models were tested using the reserved unseen test data to make predictions, and predictions were used to create confusion matrices and calculate evaluation metrics (precision, recall, and f1 score).

Results

The results of evaluation showed that the Random Forest classifier had the highest precision (100%), as well as highest f1 score (93%). The Decision Tree classifier had the poorest precision and f1 score at 87%. The classifier with the best recall was Naïve Bayes (91%). The Support Vector Machine classifier demonstrated the worst recall (85%).

Figure 11

Summary of Model Evaluation Metrics

	Classifier	Precision Score	Recall Score	F1 Score
0	Random Forest	1.000000	0.869919	0.930435
3	SVM	0.976526	0.845528	0.906318
2	Logistic Regression	0.968468	0.873984	0.918803
1	Naive Bayes	0.892857	0.914634	0.903614
4	Decision Tree	0.870445	0.873984	0.872211

Discussion

Conclusion

The goal of this project was to implement an SMS spam classifier that accurately identified spam messages. Since it is critical to avoid falsely identifying valid messages as spam, precision is the best representative of a good spam classifier. Therefore, Random Forest proves to be the best performing model at 100% precision and 87% recall.

Limitations/Challenges

One of the challenges in working with this dataset was understanding how and when to apply natural language processing techniques, along with which techniques to use (vectorization, lemmatization, tokenization, etc.). Since working with free text data is out of the norm for me, this was the most time consuming piece of the project.

Next Steps

One opportunity for improvement would be to continue gathering additional spam SMS messages for use in training the algorithms with a balanced target class. The algorithm should be continually evaluated and re-trained with more up-to-date data to account for any changes in spammers' tactics.

Another enhancement to this project would be to perform hyperparameter tuning to optimize the models' performance as much as possible.