Data Science Project Report

SMS Spam Detection

May 6, 2019

Professor	Dr Atif Tahir
Project Member 1	Abdul Munim Khan(K15-2897)
Project Member 2	Muhammad Moiz Arif(K15-2146)
Submission Date	May 06, 2019

Task	Performed By
Setting Research Goal	Both
Retrieving Data	Member 1
Data Preparation	Member 1
Data Exploration	Member 2
Data Modeling	both
Data Presentation	both

Contents

1 Introduction 3

2 Setting Research Goal 3

3 Retrieving Data 3

4 Data Preparation 3

5 Data Modeling 4

6 Presentation and Automation 4

7 Presentation and Automation 5

1 Introduction

This document contains the steps of Data Science Process for the project that is SMS Spam Detection.

2 Setting Research Goal

The purpose of the project is to classify the text messages based on the prediction done by the trained model that is trained by the given dataset. We are doing the message classification as Spam or Ham that is the message is from legal sender or it's a chain of same messages in order to just generate traffic on the targeted network or with some negative intention. We will use Naïve Bayes algorithm for the text classification as it is the best algorithm used in textual analysis. The Naïve Bayes algorithm use the Bayes law which is stated following:

$P(y|x) = (P(x|y) * P(y))/P(x) - - - - - - - - - - - - - (i)$

The probability of y given that the event x occurs. We are using this law for spam classification as follows.

$P(spam|W1, W2, W3) = (P1 * P2 * P3)/(P1 * P2 * P3) * (1 - P1) * (1–P2) * (1–P3)$

The above formula is just elaboration of how we will predict the spam message after the training from dataset.

3 Retrieving Data

The data has been taken from the online dataset provider repository that is UCI Machine Learning Repository. The dataset is present at the following link.

https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection#

Given Dataset has following properties: Downloaded dataset has 5574 lines of messages Data/"Review Data1".png

```
#open file
message = [line.rstrip() for line in open('SMSSpamCollection.csv')]
print(len(message))
```

```
5574
```

Our Dataset has total 5572 messages, having 2 unique labels HAM and SPAM. Most of the messages are HAM.Data/"Review

```
# Describe Message Statistics
message.describe()
```

	labels	message
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

Data2".png

Our dataset has following numeric values Data/"Review Data3".png

```
#describe different statistics of message
message.length.describe()
```

```
count     5572.000000
mean        80.489950
std         59.942907
min          2.000000
25%         36.000000
50%         62.000000
75%        122.000000
max        910.000000
Name: length, dtype: float64
```

4 Data Preparation

According to the UCI repository dataset information, the data is collected from different resources and so it must need to be is particular order so that the model can be train from specified attributes. Therefore, for data cleaning only the textual analysis involves so we removed the extra spaces from the messages and those words that are not in the dictionary that is removal of slang words.

We have downloaded nltk library to get dictionary of stopwardsPreparation1.png

```
#Remove Extra Unimportant words
message['message'].head(5).apply(text_process)
```

```
0    [Go, jurong, point, crazy, Available, bugis, n...
1                       [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, 2, wkly, comp, win, FA, Cup, fin...
3        [U, dun, say, early, hor, U, c, already, say]
4    [Nah, dont, think, goes, usf, lives, around, t...
Name: message, dtype: object
```

From figure below $(747 * 100/5572) =$ 13.46% text messages from the given dataset are spam and the rest of the messages that is 86.54% of the messages are ham. Also the messages are repeating (some messages), so the unique messages in both ham and spam are less than the total count. The frequency of the most repeated text is thirty it means that the message 'Sorry, I'll call later' arises thirty times in the messages.

The frequency of the most repeated text is thirty it means that the message 'Sorry, I'll call later' arises thirty times in the messages.

```
message.describe()
```

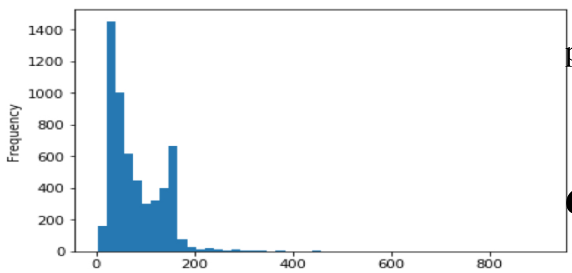
	labels	message
count	5572	5572
unique	2	5169
top	ham	Sorry, I'll call later
freq	4825	30

```
message.groupby('labels').describe()
```

	message			
	count	unique	top	freq
labels				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

```
message['length'].plot(bins=50,kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xe6e1ac8>
```



5 Data Modeling

We have tokenize the data that is tokenize the messages into the list of words. Ex We have the message in the .csv file as 'Ok lar joking wif u oni'. After tokenization this message becomes list of words as shown. [Ok, lar, joking, wif, u, oni].

Secondly, we tokenize the messages by stemming parts of speech. For Ex. In the above message given the wif is wife and u is you. Therefore replacing the correct word from its short. The results of the trained model are given as follows. The following fig showing the predicted and expected value of the message 3 in the dataset. As discuss we fit model into count vectorizerModeling1.png

```
In [28]: #fit in vector and print its length
bow_transformer = CountVectorizer(analyzer=text_process).fit(message['message'])
print(len(bow_transformer.vocabulary_))

11425

In [30]: #extract message 3 and put in variable message4
message4=message['message'][3]
print(message4)

U dun say so early hor... U c already then say...
```

We have find sparsity, because every message won't contain all words in dictionary

```
In [31]: #transform message4 into vector
bow4=bow_transformer.transform([message4])
print(bow4)
print(bow4.shape)

  (0, 4068)	2
  (0, 4629)	1
  (0, 5261)	1
  (0, 6204)	1
  (0, 6222)	1
  (0, 7186)	1
  (0, 9554)	2
(1, 11425)
```

```
In [35]: #sparsity calculate
sparsity =(100.0 * messages_bow.nnz/(messages_bow.shape[0]*messages_bow.shape[1]))
print('sparsity:{}'.format(round(sparsity)))

sparsity:0
```

We have fit data using MultinomialNB, and start predicting.

```
In [39]: #The multinomial Naive Bayes classifier is suitable for classification with discrete features
#(e.g., word counts for text classification).
#The multinomial distribution normally requires integer feature counts.
#However, in practice, fractional counts such as tf-idf may also work.
from sklearn.naive_bayes import MultinomialNB
spam_detect_model = MultinomialNB().fit(messages_tfidf,message['labels'])

In [40]: print('predicted:',spam_detect_model.predict(tfidf4)[0])
print('expected:',message['labels'][3])

predicted: ham
expected: ham
```

6 Presentation and Automation

The results of the trained model are given as follows. The following fig showing the predicted and expected value of the message 3 in the dataset.

```
In [174]: print('predicted:',spam_detect_model.predict(tfidf4)[0])
print('expected:',message['labels'][3])

('predicted:', 'ham')
('expected:', 'ham')
```

The overall results for the entire dataset is shown below.

```
In [175]: all_predictions = spam_detect_model.predict(messages_tfidf)
          print(all_predictions)

['ham' 'ham' 'spam' ... 'ham' 'ham' 'ham']
```

7 Presentation and Automation

	precision	recall	f1-score	support
ham	1.00	0.96	0.98	1008
spam	0.70	1.00	0.83	107
micro avg	0.96	0.96	0.96	1115
macro avg	0.85	0.98	0.90	1115
weighted avg	0.97	0.96	0.96	1115

Final Stats