Text classification - SMS spam detection

A classification task with Bag of Words, term frequency–inverse document frequency, and Naive Bayes model

In [1]:
```python
# Import libraries
import chardet
import html
import pandas as pd
import string

import nltk
nltk.download('omw-1.4')
nltk.download('wordnet')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

import re

from sklearn import metrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\C\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\C\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\C\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

1. Import Dataset

In [2]:
```python
# Read the dataset
with open('sms-spam1.csv', 'rb') as f:
    result = chardet.detect(f.read())  # or readline if the file is large

df_sms = pd.read_csv('sms-spam1.csv', encoding=result['encoding'])
df_sms.head(5)
```

Out[2]:

	target	text
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...

2. Text pre-processing

In [3]:
```python
# print row with HTML elements "&lt;#&gt;"

df_sms.loc[[78]]
```

Out[3]:

	target	text
78	ham	Does not operate after <#> or what

In [4]:
```python
#convert HTML characters and print result

df_sms_clean = df_sms.applymap(lambda x: html.unescape(x))
df_sms_clean.loc[[78]]
```

Out[4]:

	target	text
78	ham	Does not operate after <#> or what

In [5]:
```python
#create 2 df of spam / ham

spam_messages = df_sms_clean[df_sms_clean["target"] == "spam"]["text"]
ham_messages = df_sms_clean[df_sms_clean["target"] == "ham"]["text"]
print(f"Number of spam messages: {len(spam_messages)}")
print(f"Number of ham messages: {len(ham_messages)}")
```

```
Number of spam messages: 747
Number of ham messages: 4827
```

In [6]:
```python
# helper function for processing text
def text_preprocess(message):
    # Remove punctuations
    nopunc = [char for char in message if char not in string.punctuation]

    # Join the characters again
    nopunc = "".join(nopunc)
    nopunc = nopunc.lower()

    # Remove any stopwords and non-alphabetic characters
    nostop = [
        word
        for word in nopunc.split()
        if word.lower() not in stopwords.words("english") and word.isalpha()
    ]
    return nostop
```

In [7]:
```python
# Remove punctuations/stopwords from all messages
df_sms_clean["text"] = df_sms_clean["text"].apply(text_preprocess)
df_sms_clean.head()
```

Out[7]:

	target	text
0	ham	[go, jurong, point, crazy, available, bugis, n...
1	ham	[ok, lar, joking, wif, u, oni]
2	spam	[free, entry, wkly, comp, win, fa, cup, final,...
3	ham	[u, dun, say, early, hor, u, c, already, say]
4	ham	[nah, dont, think, goes, usf, lives, around, t...

In [8]:
```python
# Convert messages (as lists of string tokens) to strings
df_sms_clean["text"] = df_sms_clean["text"].agg(lambda x: " ".join(map(str, x)))
df_sms_clean.head()
```

Out[8]:

	target	text
0	ham	go jurong point crazy available bugis n great ...
1	ham	ok lar joking wif u oni
2	spam	free entry wkly comp win fa cup final tkts may...
3	ham	u dun say early hor u c already say
4	ham	nah dont think goes usf lives around though

In [9]:
```python
# lemmatize words to reduce words to their dictionary meaning
lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    words = text.split()
    words = [lemmatizer.lemmatize(word,pos='v') for word in words]
    return ' '.join(words)
df_sms_clean["text"] = df_sms_clean["text"].apply(lemmatize_words)
```

In [10]:
```python
# Convert spam and ham labels to 0 / 1
FactorResult = pd.factorize(df_sms_clean["target"])
df_sms_clean["target"] = FactorResult[0]
df_sms_clean.head()
```

Out[10]:

	target	text
0	0	go jurong point crazy available bugis n great ...
1	0	ok lar joke wif u oni
2	1	free entry wkly comp win fa cup final tkts may...
3	0	u dun say early hor u c already say
4	0	nah dont think go usf live around though

In [ ]:

3. Classification

In [11]:
```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression

import seaborn as sns
import matplotlib.pyplot as plt
```

In [12]:
```python
# Initialize count vectorizer
vectorizer = CountVectorizer()
vectorizer.fit(df_sms_clean["text"])

# Fetch the number vocabulary set
print(f"Total number of vocab words: {len(vectorizer.vocabulary_)}")
```

```
Total number of vocab words: 7170
```

In [ ]:

In [13]:
```python
messages = df_sms_clean["text"]

# Take top 2500 features
cv = CountVectorizer(max_features=2500, ngram_range=(1,3))
X = cv.fit_transform(messages).toarray()
y = df_sms_clean["target"]
```

In [14]:
```python
#split data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=15, stratify=y)
```

In [15]:
```python
# Apply TDF-IDF

tf = TfidfVectorizer(ngram_range=(1,3), max_features=2500)
X = tf.fit_transform(messages).toarray()
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=15, stratify=y)
```

Naive Bayes model

In [16]:
```python
# create model with create Naive Bayes model
model = MultinomialNB()
```

In [17]:
```python
# fit data into model
model.fit(x_train, y_train)
```

Out[17]:

```
▾ MultinomialNB
MultinomialNB()
```

In [18]:
```python
# predict the labels of training and test sets
train_pred = model.predict(x_train)
test_pred = model.predict(x_test)
```

Logistic regression model

In [19]:
```python
# create second model with logistic regression
logreg = LogisticRegression(random_state=16)

# fit the model with data
logreg.fit(x_train, y_train)

test_pred_log = logreg.predict(x_test)
```

4. Evaluation

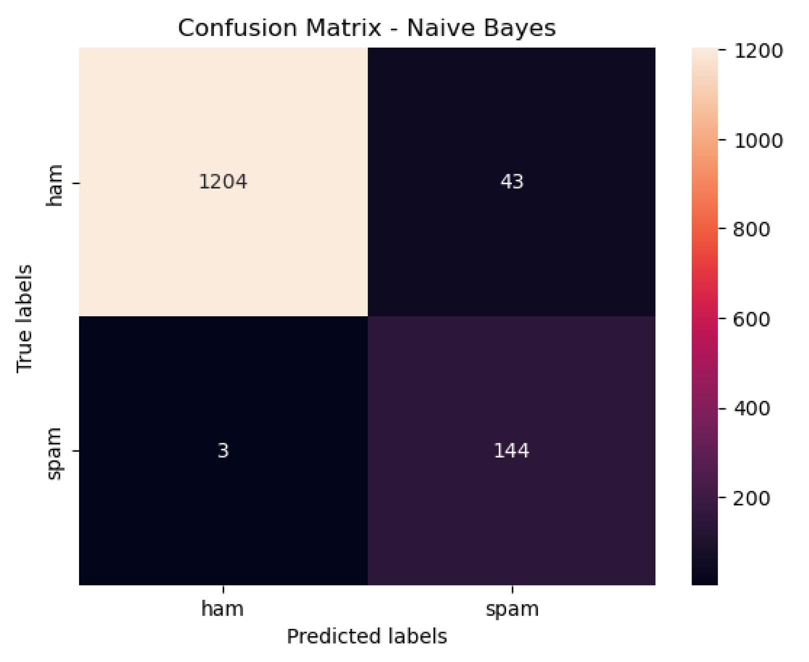
In [20]:
```python
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

import seaborn as sns
import pylab as pl
```

In [21]:
```python
print('---Train results')
print(classification_report(train_pred, y_train))
print('---Naive Bayes Test results')
print(classification_report(test_pred, y_test))
print('--- Logistic regression Test results')
print(classification_report(test_pred_log, y_test))
```

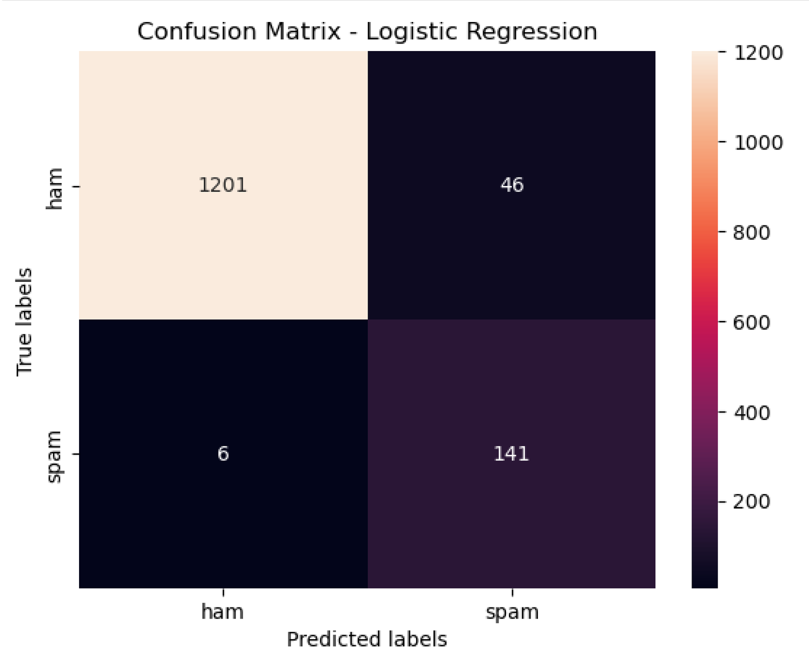
```
---Train results
              precision    recall  f1-score   support

           0       1.00      0.97      0.99      3722
           1       0.81      1.00      0.90       458

    accuracy                           0.97      4180
   macro avg       0.91      0.98      0.94      4180
weighted avg       0.98      0.97      0.98      4180

---Naive Bayes Test results
              precision    recall  f1-score   support

           0       1.00      0.97      0.98      1247
           1       0.77      0.98      0.86       147

    accuracy                           0.97      1394
   macro avg       0.88      0.97      0.92      1394
weighted avg       0.97      0.97      0.97      1394

--- Logistic regression Test results
              precision    recall  f1-score   support

           0       1.00      0.96      0.98      1247
           1       0.75      0.96      0.84       147

    accuracy                           0.96      1394
   macro avg       0.87      0.96      0.91      1394
weighted avg       0.97      0.96      0.96      1394
```

Create visual representation of confusion matrix with seaborn.heatmap

In [25]:
```python
cm = confusion_matrix(test_pred, y_test)

ax= plt.subplot()
sns.heatmap(cm, annot=True, fmt='g', ax=ax);

ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix - Naive Bayes');
ax.xaxis.set_ticklabels(['ham', 'spam']); ax.yaxis.set_ticklabels(['ham', 'spam']);
```



In [27]:
```python
cm_log = confusion_matrix(test_pred_log, y_test)
ax= plt.subplot()
sns.heatmap(cm_log, annot=True, fmt='g', ax=ax);

ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix - Logistic Regression');
ax.xaxis.set_ticklabels(['ham', 'spam']); ax.yaxis.set_ticklabels(['ham', 'spam']);
```



Test model with sample SMS message

Input any sample SMS message to get an output of spam or not spam

In [24]:
```python
print('Predicting input...')

# enter any sample SMS message to test model
test_message = ["win prize money now call landline redem"]

message_vector = tf.transform(test_message)
category = model.predict(message_vector)
print("The message IS", "spam" if category == 1 else "NOT spam")
```

```
Predicting input...
The message IS spam
```

In [ ]:

In [ ]:

Coursework Assignment: Text classification

Notes:
** code snippets will be denoted in `consolas font with grey shading`
** All text, code and work in this project and report are original and none of the content has been generated, corrected or inspired by a large language model.

I. Introduction

1. Introduction to the domain-specific area

The domain for this project will be spam classification. In the age of technology, people may receive all kinds of messages of many forms such as: SMS text message, emails, traditional mail, etc. The number of messages a person received may also be increasing over time.

Lately, SMS marketing has been a hot topic in the advertising industry with growth potential forecasted at "Steady CAGR of 21.26% by 2030".[1] Another report estimates a value of "USD 26292.28 Million by 2030" and "CAGR of 23.08% from 2022 to 2030."[2]

The question is how to better filter out the ever-increasing number of spam, unsolicited marketing, automated messages, or unwanted messages a person may receive? A person in theory may manually sift through all the messages, but that may be labor, time intensive, or unrealistic.

So, the proposed solution is to create an NLP model that can identify if a message is spam or not spam.

2. Objectives of the project

The objective of this project is to create an NLP based text classifier and apply it to a real SMS corpus dataset to result in an output label of spam or not.

The acquired dataset will need to pre-processed and arranged into a suitable format for the implementation part. We will mainly be using the applying NLP theories such as removing stopwords and punctuation. Another important step to consider is the usage of a lemmatizer so we can identify and convert the words into their base form. This will provide an advantage as the variations of the same words will be considered as one word by the Bag of Words model.

1 https://www.globenewswire.com/en/news-release/2023/04/25/2653791/0/en/Short-Message-Service-Marketing-Market-Set-to-Achieve-Phenomenal-Growth-of-USD-38442-44-Million-with-a-Steady-CAGR-of-21-26-by-2030-Size-Share-Trends-Demand-Growth-and-Opportunity-.html

2 https://www.verifiedmarketresearch.com/product/sms-marketing-software-market/

For implementation, the idea is to identify and count the frequency of certain unique keywords that appear in a sample of SMS message. We will also enhance the Bag of Words by using Term Frequency-Inverse Document Frequency (TF-IDF). This method is used for assigning the weight factor to the words in the corpus by identifying how many times in the message vs how messages contain the word.

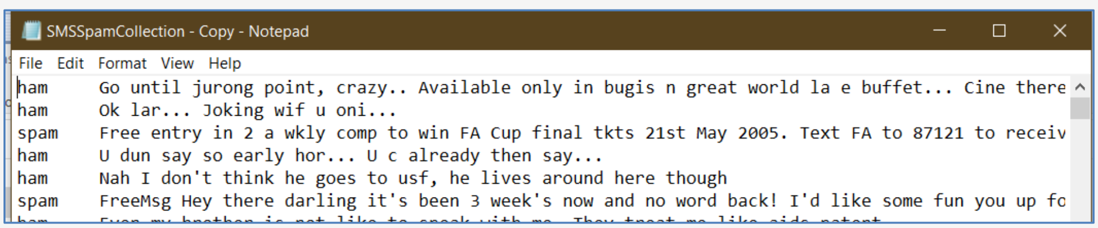
The Naïve Bayes model will be used as the classifier as it is suitable for text classification tasks. This is already part of the `sklearn.naive_bayes import MultinomialNB`, so we can use that directly.

3. Description of the selected dataset

The dataset was obtained from the UCI Machine Learning Repository[3], labeled as SMS Spam Collection with a donate date of 6/21/2012. This dataset contains a mix of 5574 real text messages, composing of 425 SMS spam messages and 3,375 randomly chosen ham messages from the NUS SMS Corpus (NSC). NUS SMS Corpus is a larger dataset of 10,000 legitimate messages collected for research by the Department of Computer Science at the National University of Singapore.

The downloaded dataset came without formatting, so it was required to open in Excel and covert to CSV for further processing.

It is also noted that there are 2 columns of information as seen in the below sample:



The first column is the label and the second column is the text part of the message. Both are separated by a tab.

Upon further examination of this dataset, a few things stand out:

1. Presence of HTML Character Entities, which will need to be converted later. Example snippet to provide context (line 80 of dataset): `Does not operate after  <#>  or what`
2. Difference in regional language use between spam and ham sets. The spam set was collected in the UK from while ham set was collected in Singapore. Although both are in English, distinct language, slang, and word choices are noticeable as in the below 2 examples and screenshot example:

3 https://archive.ics.uci.edu/dataset/228/sms+spam+collection

a.) (spam): WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.

b.) (ham): U dun say so early hor... U c already then say...

	target	text
4	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
7	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv
10	spam	WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
11	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
13	spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
14	spam	URGENT! You have won a 1 week FREE membership in our £100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
17	spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL

	target	text
2	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
3	ham	Ok lar... Joking wif u oni...
5	ham	U dun say so early hor... U c already then say...
6	ham	Nah I don't think he goes to usf, he lives around here though
8	ham	Even my brother is not like to speak with me. They treat me like aids patent.
9	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
12	ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.

This has the potential to create a bias with the model as it will be easy to identify the two types by the unique use of vocabulary, but the results won't be clear until the project progresses to the evaluation phase.

A wordclould of the 2 categories are shown below for a view of the most common words / vocabulary:

Spam wordcloud

Ham wordcloud

4. Evaluation methodology

After going through the steps of vectorizing for bag of words model, applying TDF-IDF, applying test train split, creating and fitting the Naive Bayes model, and creating a second Logistic regression model, we ended up with the outputs from the `model.predict`.

The reason for selecting a second different classification model is to gather more results, and in terms of research, increase sample size. This has the advantage to interpret the results of the models and identify the likeness or unlikeness of both in order to evaluate if any mistakes in methodology were introduced. In a sense, it can be used as extra verification of the results given the same input.

Evaluation metrics that were used are precision, recall, f1 score, and confusion matrix metrics. The “target” column of the dataset had the 2 labels of spam and ham, which were then factorized and converted to 0 or 1 accordingly.

The `sklearn.datasets import make_classification / classification_report` was used to print out the statistics of both train and test sets.

Some variations of the parameters of the `train_test_split(X, y, test_size=0.20, random_state=11, stratify=y)` such as test size, and random states were modified, but the results mostly were around the same and within a tight range.

III. Conclusions

9. Evaluation

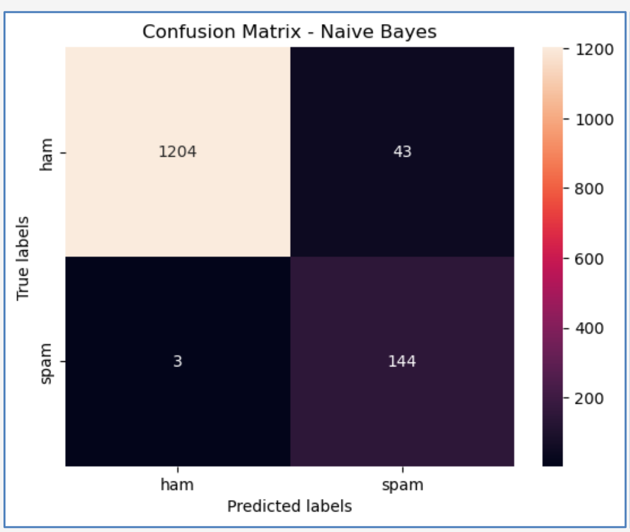
A few metrics were used to evaluate the results of the model.

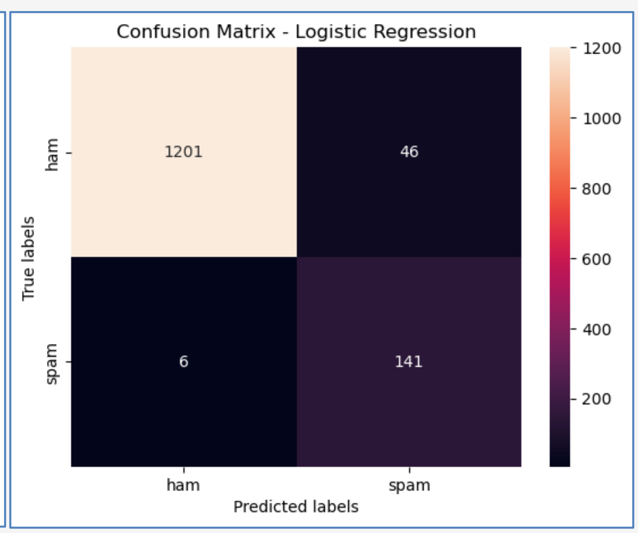
As in part "4. Evaluation", we can tell how well the two models are doing by printing out the results of the `classification_report`

```
---Train results
              precision    recall  f1-score   support

           0       1.00      0.97      0.99      3722
           1       0.81      1.00      0.90       458

    accuracy                           0.97      4180
   macro avg       0.91      0.98      0.94      4180
weighted avg       0.98      0.97      0.98      4180

---Naive Bayes Test results
              precision    recall  f1-score   support

           0       1.00      0.97      0.98      1247
           1       0.77      0.98      0.86       147

    accuracy                           0.97      1394
   macro avg       0.88      0.97      0.92      1394
weighted avg       0.97      0.97      0.97      1394

--- Logistic regression Test results
              precision    recall  f1-score   support

           0       1.00      0.96      0.98      1247
           1       0.75      0.96      0.84       147

    accuracy                           0.96      1394
   macro avg       0.87      0.96      0.91      1394
weighted avg       0.97      0.96      0.96      1394
```

The bottom two results represent the test results, and we can see that both perform very similarly with Naïve Bayes performing slightly better in terms of precision.

Confusion matrix in a heatmap visual representation for the two models:





This heatmap representation of the results provides a better visual representation. We can see that the models are more likely to misclassify spam as ham, then the other way around. Ham classification precision is quite high with minimal error in prediction. Since the objective was to capture and label potential spam, I would conclude that the results are quite favorable.

In this context of SMS message filtering, classifying spam as ham is not as disruptive as misclassifying ham as spam as the consequences of missing an important ham SMS message is much greater than receiving a spam SMS message.

10. Evaluation of the project and its results

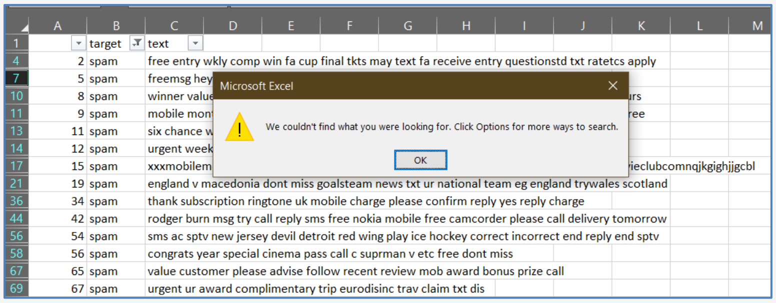
The results in the end were quite favorable as can be seen by the evaluation metrics. The two models were able to classify the SMS message as originally intended at the start of the project.

Admittedly, there are a few things that would improve this project. The first is to source another SMS dataset where the ham and spam categories were collected in the same place and timeframe.

As outlined before, the ham and spam data were collected from two geographic locations and the time of collection of the two sets are unknown. There are some specific words and vocabulary that are specific to the locations that may have made it easier for the two classification models to separate into its intended category. It is hypothesized that the unique word choice of the geographic locations would create unique words in the bag of words model, which would only be present in one of the sets.

For example, the word "lor" in the dataset was quite noticeable and prevalent. In Singapore "Singlish"[4] the term is mainly used as "discourse particles that are mentioned at the end of sentences."[5]

No examples of "lor" as a single word was located via search in Excel of the lemmatized set with only filtering for spam samples,



[4] https://medium.com/@visakanv/lah-leh-lor-and-so-on-d5ec2b258fd6
[5] https://www.timeout.com/singapore/things-to-do/common-singlish-words-you-need-to-know-to-speak-like-a-local

This is only speculation as I could not source or locate a SMS spam dataset that could fit the criteria for testing. It is speculated that only proprietary datasets from mobile telecommunication companies would be able to meet the criteria of data collection in same location and same timeframe.

Another thing that can be improved is if the ratio of spam and ham messages in the dataset were within the range of real-life ratio. One research paper noted that the ratio of spam to ham in the dataset is 14% to 86%.[6] A second research paper had a ratio in their dataset of 15.4%[7].

The acquired dataset for this project sits at around 12.6% spam ratio. If we were going by the route of the proprietary dataset that was collected in the same geographical location and at the same timeframe, an accurate ratio of the spam vs ham would be ideal for performance during the training phase.

Project examples and inspiration:

https://blog.paperspace.com/nlp-spam-detection-application-with-scikitlearn-xgboost/

https://www.makeuseof.com/spam-classifier-natural-language-processing-build-from-scratch/

[6] https://www.researchgate.net/figure/Ratio-of-ham-and-spam-messages_fig2_342821988
[7] https://www.scitepress.org/Papers/2020/100224/100224.pdf