

SMS Spam Collection Dataset

Data source- UCI Machine Learning Repository (<https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection>)

```
In [1]: import nltk

In [2]: # to download the necessary datasets

nltk.download()

showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml

True

In [2]: # Using UCI datasets- contains collection of more than 5000 sms

messages = [line.rstrip() for line in open('smsspamcollection/SMSSpamCollection')]

In [3]: print(len(messages))

5574

In [4]: # numbering the text using enumerate

for message_no, message in enumerate(messages[:10]):
    print(message_no, message)
    print('\n')

0 ham    Go until jurong point, crazy.. Available only in bugis n great world is e buffet... Cine there got amore wat...

1 ham    Ok lar... Joking wif u oni...

2 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 r
ate)TVC's apply 08452810075over18's

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 around here though

5 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

6 ham    Even my brother is not like to speak with me. They treat me like aids patient.

7 ham    As per your request 'Melle Melle (Oru Minnaminunginte Nurgunu Vettam)' has been set as your callertune for all Callers.
Press *9 to copy your friends Callertune

8 spam   WINNER!! As a valued network customer you have been selected to receive a £900 prize reward! To claim call 09061701461. C
lain code KLM341. Valid 12 hours only.

9 spam   Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mo
bile Update Co FREE on 08002966030

In [7]: # We notice that the data is tab separated, so we can make a dataframe using pandas

In [5]: import pandas as pd

In [6]: messages = pd.read_csv('smsspamcollection/SMSSpamCollection', sep='\t',
names=['label', 'message'])

In [7]: messages.head()

   label      message
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...

In [8]: len(messages)

5574

In [9]: messages.describe()

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

In [10]: messages.groupby('label').describe()

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

In [13]: messages['length'] = messages['message'].apply(len)

In [14]: messages.head()

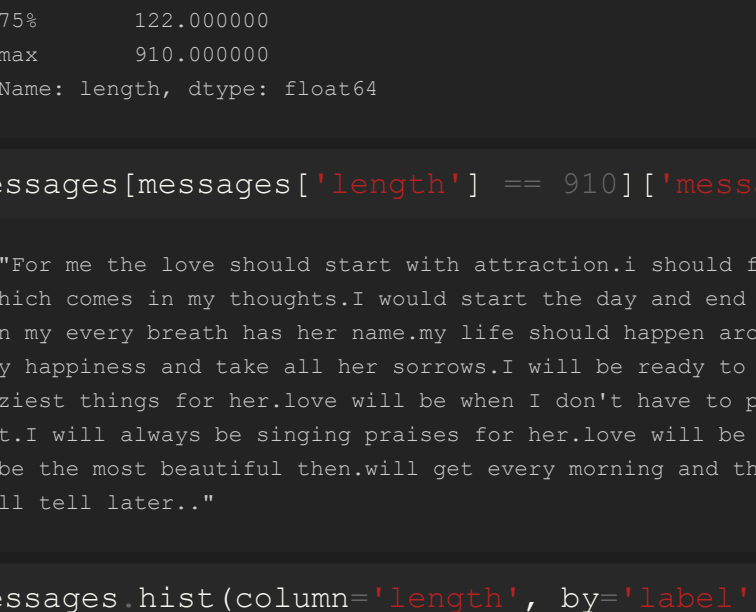
   label      message      length
0  ham    Go until jurong point, crazy.. Available only ...    111
1  ham    Ok lar... Joking wif u oni...                29
2  spam   Free entry in 2 a wkly comp to win FA Cup fina...   155
3  ham    U dun say so early hor... U c already then say...    49
4  ham    Nah I don't think he goes to usf, he lives aro...    61

Visualization
```

```
In [15]: import matplotlib.pyplot as plt
import seaborn as sns
matplotlib.inline

In [16]: messages['length'].plot(bins=50, kind='hist')

<matplotlib.axes._subplots.AxesSubplot at 0x7fca23ba3510>
```



```
In [17]: messages.length.describe()

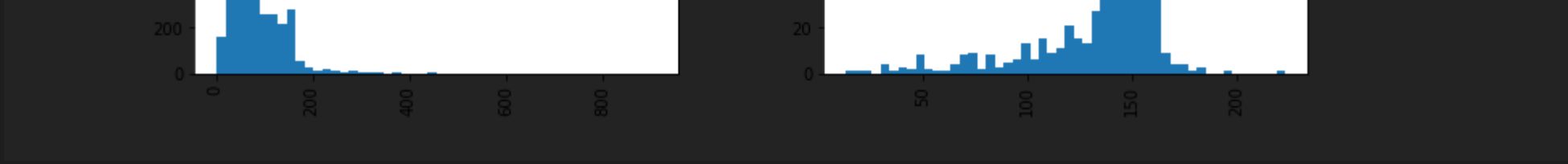
count      5572.000000
mean        80.489950
std         59.848907
min          2.000000
25%         36.000000
50%         62.000000
75%        122.000000
max         910.000000
Name: length, dtype: float64
```

```
In [18]: messages[messages['length'] == 910]['message'].iloc[0]
```

```
"For me the love should start with attraction..I should feel that I need her every time around me..she should be the first thing w
hich comes in my thoughts..I would start the day and end it with her..she should be there every time I dream..love will be then whe
n my every breath has her name..my life should happen around her..my life will be named to her..I would cry for her..will give ail m
y happiness and take all her sorrows..I will be ready to fight with anyone for her..I will be in love when I will be doing the cra
ziest things for her..love will be when I don't have to prove to anyone that my girl is the most beautiful lady on the whole plane
t..I will always be singing praises for her..love will be when I start up making chicken curry and end up making sambar..life will
be the most beautiful then..will get every morning and thank god for the day because she is with me..I would like to say a lot..wi
ll tell later..."
```

```
In [19]: messages.hist(column='length', by='label', bins=50, figsize=(12,4))

array([<matplotlib.axes._subplots.AxesSubplot object at 0x7fca25a13d90>,
      <matplotlib.axes._subplots.AxesSubplot object at 0x7fca26f23250>],
      dtype=object)
```



Text Pre-Processing

```
In [20]: import string

mess = 'Sample message! Notice: it has punctuation.'

# Check characters to see if they are in punctuation
nopunc = [char for char in mess if char not in string.punctuation]
nopunc
```

```
['s',
 'a',
 'm',
 'e',
 's',
 's',
 'a',
 'g',
 'e',
 ' ',
 ' ',
 'm',
 'e',
 's',
 's',
 'a',
 'g',
 'e',
 ' ',
 ' ',
 'n',
 'o',
 't',
 'i',
 'c',
 'e',
 ' ',
 ' ',
 'i',
 't',
 ' ',
 'h',
 'a',
 's',
 ' ',
 'p',
 'u',
 'n',
 'c',
 't',
 'u',
 'a',
 't',
 'i',
 'o',
 'n',
 '']
```

```
# joining the characters to make it a string

nopunc = ''.join(nopunc)
nopunc
```

```
'Sample message Notice it has punctuation'
```

```
from nltk.corpus import stopwords

In [26]: from nltk.corpus import stopwords

stopwords.words('english')[0:10]
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

```
stopwords.words('english')[0:10]
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

```
nopunc.split()
```

```
['Sample', 'message', 'Notice', 'it', 'has', 'punctuation']
```

```
# Now just removing stopwords from it

clean_mess = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]

In [30]: clean_mess
```

```
['Sample', 'message', 'Notice', 'punctuation']
```

Now putting these together to make a function and then apply it our dataframe later

```
In [31]: def text_process(mess):
    """
    Takes in a string of text, then performs the following:
    1. Remove all puctuations
    2. Remove all stopwords
    3. Returns a list of the cleaned text
    """

    nopunc = [char for char in mess if char not in string.punctuation]

    #Now making it a string
    nopunc = ''.join(nopunc)

    #Now removing any stopwords
    return [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]

In [32]: messages.head()
```

```
   label      message      length
0  ham    Go until jurong point, crazy.. Available only ...    111
1  ham    Ok lar... Joking wif u oni...                29
2  spam   Free entry in 2 a wkly comp to win FA Cup fina...   155
3  ham    U dun say so early hor... U c already then say...    49
4  ham    Nah I don't think he goes to usf, he lives aro...    61
```

```
In [34]: messages[messages['length'] == 910].apply(text_process)
```

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

Normalizing Text

Read theory in the Lecture notebook

```
In [35]: from sklearn.feature_extraction.text import CountVectorizer

In [36]: bow_transformer = CountVectorizer(analyzer = text_process).fit(messages['message'])

# Print total number of vocabal words
print(len(bow_transformer.vocabulary_))

11425
```

Let's take one text message and get its bag-of-words counts as a vector, putting to use our new bow_transformer

```
In [37]: message4 = messages['message'][3]
print(message4)

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

now let's see it's vector representation:

```
In [39]: bow4 = bow_transformer.transform([message4])
print(bow4)
print(bow4.shape)
```

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

This means that there are 7 unique words in message number 4 (after removing common stop words) 2 of them appear twice, the rest only once. Let's go ahead and check and confirm which ones appear twice:

```
In [40]: print(bow_transformer.get_feature_names() [4068])
print(bow_transformer.get_feature_names() [9554])

u
say
```

Now we can use .transform on our Bag-of-words (bow) transformed object and transform the entire DataFrame of messages. Let's go ahead and check out how the bag-of-words counts for the entire SMS corpus is a large, sparse matrix:

```
In [41]: messages_bow = bow_transformer.transform(messages['message'])

In [42]: print('Shape of Sparse Matrix:', messages_bow.shape)
print('Amount of Non-Zero occurrences:', messages_bow.nnz)
```

```
Shape of Sparse Matrix: (5572, 11425)
Amount of Non-Zero occurrences: 50548

In [43]: sparsity = (100 * messages_bow.nnz / (messages_bow.shape[0] * messages_bow.shape[1]))
print('Sparsity: {}'.format(round(sparsity)))

Sparsity: 0
```

```
In [44]: from sklearn.feature_extraction.text import TfidfTransformer

In [45]: tfidf_transformer = TfidfTransformer().fit(messages_bow)
tfidf4 = tfidf_transformer.transform(bow4)
print(tfidf4)
```

```
(0, 9554)      0.5385626262927564
(0, 7186)      0.4389365653379857
(0, 6222)      0.3187216892949149
(0, 6204)      0.2395393923697416
(0, 5261)      0.23723957402868723
(0, 4629)      0.246198013906087187
(0, 4068)      0.4083258993384067
```

we'll go ahead and check what is the IDF of the word "u" and of word "university"

```
In [46]: print(tfidf_transformer.idf_[bow_transformer.vocabulary_['u']])
print(tfidf_transformer.idf_[bow_transformer.vocabulary_['university']])

3.2800524247409408
8.527076498901426
```

to transform the entire bag-of-words corpus into TF-IDF corpus at once:

```
In [48]: messages_tfidf = tfidf_transformer.transform(messages_bow)
print(messages_tfidf.shape)

(5572, 11425)
```

Training a model

we'll be using scikit-learn, choosing Naive Bayes classifier

```
In [50]: from sklearn.naive_bayes import MultinomialNB

In [51]: spam_detect_model = MultinomialNB().fit(messages_tfidf, messages['label'])
```

Let's try classifying our single random message and checking how we do:

```
In [53]: print('predicted: ', spam_detect_model.predict(tfidf4)[0])
print('expected: ', messages.label[3])

predicted: ham
expected: ham
```

Model Evaluation

```
In [54]: all_predictions = spam_detect_model.predict(messages_tfidf)
print(all_predictions)

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

In [55]: from sklearn.metrics import classification_report

In [56]: print(classification_report(messages['label'], all_predictions))
```

```
              precision    recall  f1-score   support

     ham         0.98         1.00         0.99         4825
     spam         1.00         0.85         0.92         747

 accuracy          0.99
 macro avg         0.99         0.92         0.95
weighted avg         0.98         0.98         0.98
```

This model is actually not good because we used all of the data for training and predicting. Doing train test split now

Train Test Split

```
In [57]: from sklearn.model_selection import train_test_split

In [58]: msg_train, msg_test, label_train, label_test = train_test_split(messages['message'],
                                messages['label'], test_size=0.2)

In [59]: print(len(msg_train), len(msg_test), len(msg_train) + len(msg_test))

4457 1115 5572
```

Creating a Data Pipeline

```
In [60]: from sklearn.pipeline import Pipeline

In [62]: pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process)), #strings to token integer counts
    ('tfidf', TfidfTransformer()), #integer counts to weighted TF-IDF scores
    ('classifier', MultinomialNB()), #train on TF-IDF vectors w/ Naive Bayes classifier
])
```

Now we can directly pass message text data and the pipeline will do our pre-processing for us! We can treat it as a model/estimator API:

```
In [67]: pipeline.fit(msg_train, label_train)
```

```
Pipeline(memory=None,
       steps=[('bow', CountVectorizer(analyzer=<function text_process at 0x7fca276a84d0>,
                                     binary=False, decode_error='strict',
                                     dtype=<class 'numpy.int64'>, encoding='utf-8',
                                     input='content', lowercase=True, max_df=1.0,
                                     max_features=None, min_df=1,
                                     ngram_range=(1, 1), preprocessor=None,
                                     stop_words=None, strip_accents=None,
                                     token_pattern='(?u)\b\w+\b',
                                     tokenizer=None, vocabulary=None)),
              ('tfidf', TfidfTransformer(norm='l2', smooth_idf=True,
                                     sublinear_tf=False, use_idf=True)),
              ('classifier', MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)),
              verbose=False])

In [68]: predictions = pipeline.predict(msg_test)

In [69]: print(classification_report(predictions, label_test))
```

```
              precision    recall  f1-score   support

     ham         1.00         0.96         0.98        1011
     spam         0.73         1.00         0.84         104

 accuracy          0.86
 macro avg         0.86         0.98         0.91
weighted avg         0.97         0.97         0.97
```

In []: