Lab 12 07-10-2022

October 7, 2022

1 SMS Spam Classification Using Naive Bayes

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

1.1 Importing dataset

[2]:
```
df = pd.read_csv('spam.csv', encoding = 'latin1')
print(df)
```

```
        v1                                                 v2 Unnamed: 2  \
0      ham  Go until jurong point, crazy.. Available only …        NaN
1      ham                      Ok lar… Joking wif u oni…        NaN
2     spam  Free entry in 2 a wkly comp to win FA Cup fina…        NaN
3      ham  U dun say so early hor… U c already then say…        NaN
4      ham  Nah I don't think he goes to usf, he lives aro…        NaN
…     …                                                …        …
5567  spam  This is the 2nd time we have tried 2 contact u…        NaN
5568   ham              Will Ì_ b going to esplanade fr home?        NaN
5569   ham  Pity, * was in mood for that. So…any other s…        NaN
5570   ham  The guy did some bitching but I acted like i'd…        NaN
5571   ham                         Rofl. Its true to its name        NaN

     Unnamed: 3 Unnamed: 4
0           NaN        NaN
1           NaN        NaN
2           NaN        NaN
3           NaN        NaN
4           NaN        NaN
…          …        …
5567        NaN        NaN
5568        NaN        NaN
5569        NaN        NaN
5570        NaN        NaN
5571        NaN        NaN
```

```
[5572 rows x 5 columns]
```

1.2 Preprocessing

1.2.1 Removing unwanted columns

```
[3]: df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis = 1,inplace=True)
```

1.2.2 Renaming v1 & v2

```
[4]: df = df.rename(columns={"v1":"label","v2":"sms"})
```

```
[5]: df.info()
     df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   label   5572 non-null   object
 1   sms     5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

```
[5]:        label                     sms
     count   5572                    5572
     unique     2                    5169
     top      ham  Sorry, I'll call later
     freq    4825                      30
```

1.2.3 creating a column length

```
[6]: df['length'] = df['sms'].apply(len)
     df.head()
```

```
[6]:   label                                                sms  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
```

1.3 Training and Testing

1.3.1 Creating training and test dataset

[7]:
```
x = df['sms']
y = df['label']
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25,␣
↪random_state=0)
print(x_train)
```

```
872                          I'll text you when I drop x off
831     Hi mate its RV did u hav a nice hol just a mes…
1273    network operator. The service is free. For T &…
3314    FREE MESSAGE Activate your 500 FREE Text Messa…
4929    Hi, the SEXYCHAT girls are waiting for you to …
                               …
4931                  Match started.india  &lt;#&gt;  for 2
3264    44 7732584351, Do you want a New Nokia 3510i c…
1653    I was at bugis juz now wat… But now i'm walk…
2607    :-) yeah! Lol. Luckily i didn't have a starrin…
2732    How dare you stupid. I wont tell anything to y…
Name: sms, Length: 4179, dtype: object
```

1.3.2 CountVectorizer()

[8]:
```
from sklearn.feature_extraction.text import CountVectorizer
count_vector = CountVectorizer()
```

1.3.3 Fit & Transform

[15]:
```
train_data = count_vector.fit_transform(x_train)
test_data = count_vector.transform(x_test)
```

1.4 Implementing Naive Bayes

[11]:
```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(train_data, y_train)
```

[11]:
```
MultinomialNB()
```

[13]:
```
print(test_data)
```

```
  (0, 442)      1
  (0, 3277)     1
  (0, 2042)     1
  (0, 2798)     1
```

```
(0, 3718)     1
(0, 964)      1
(0, 3842)     1
(0, 2130)     1
(0, 3736)     1
(1, 3957)     1
(1, 3645)     1
(1, 1445)     1
(1, 2661)     1
(2, 3718)     2
(2, 3645)     1
(2, 3227)     1
(2, 2498)     1
(2, 3185)     1
(2, 3391)     1
(2, 485)      1
(2, 1722)     1
(2, 1202)     2
(2, 3198)     1
(2, 522)      1
(2, 1535)     1
:     :
(1392, 3718)  4
(1392, 3645)  1
(1392, 4140)  1
(1392, 534)   1
(1392, 1630)  1
(1392, 3797)  1
(1392, 4142)  1
(1392, 1982)  1
(1392, 2473)  1
(1392, 142)   1
(1392, 2569)  1
(1392, 919)   1
(1392, 548)   1
(1392, 2635)  1
(1392, 722)   1
(1392, 3211)  1
(1392, 3012)  1
(1392, 4069)  1
(1392, 1225)  1
(1392, 3300)  1
(1392, 1133)  1
(1392, 3509)  1
(1392, 344)   1
(1392, 1405)  1
(1392, 203)   1
```

1.4.1 Prediting the test set result

[16]:
```
y_pred = classifier.predict(test_data)
print(y_pred)
```

```
['ham' 'ham' 'ham' … 'ham' 'ham' 'spam']
```

1.4.2 Making confusion matrix

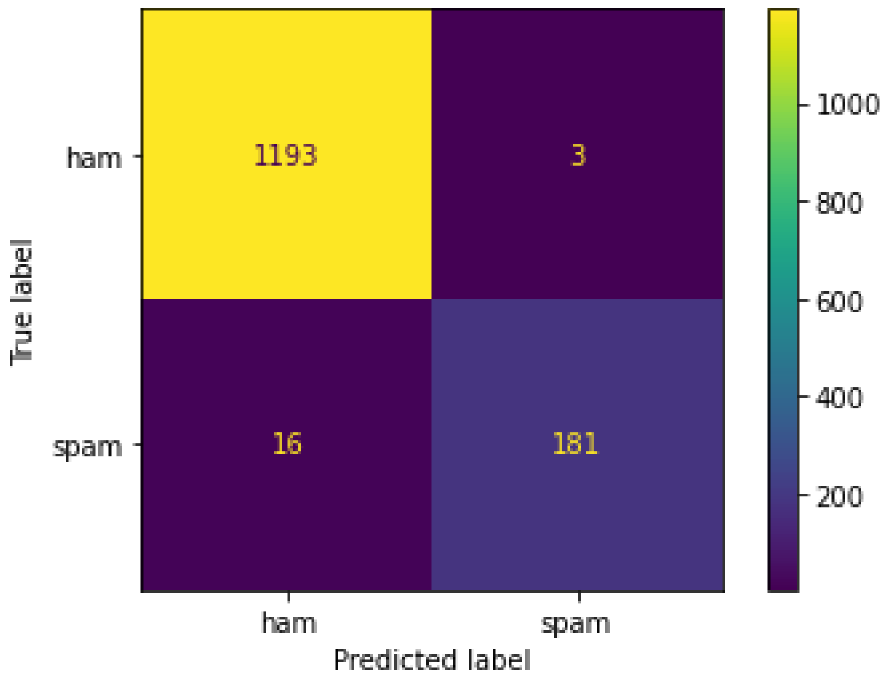
[18]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1193    3]
 [  16  181]]
```

1.4.3 Displaying CM

[19]:
```
from sklearn.metrics import ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=classifier.classes_)
disp.plot()
```

[19]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f16ab290ee0>
```



1.4.4 Accuracy calculation

[20]:
```
training_score = classifier.score(train_data, y_train)
test_score = classifier.score(test_data, y_test)
print(training_score)
print(test_score)
```

```
0.9928212491026561
0.9863603732950467
```

[21]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
  ↪f1_score
print('Accuracy score: {}'.format(accuracy_score(y_test, y_pred)))
print('Precision score: {}'.format(precision_score(y_test,␣
  ↪y_pred,pos_label="spam")))
print('Recall score: {}'.format(recall_score(y_test, y_pred,pos_label="spam")))
print('F1 score: {}'.format(f1_score(y_test, y_pred,pos_label="spam")))
```

```
Accuracy score: 0.9863603732950467
Precision score: 0.9836956521739131
Recall score: 0.9187817258883249
F1 score: 0.9501312335958005
```

1.5 Word Cloud Representation

[22]:
```
!pip install wordcloud
from wordcloud import WordCloud
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting wordcloud
  Downloading
wordcloud-1.8.2.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (458
kB)
     |                 | 458 kB 2.5 MB/s eta 0:00:01
|                              | 71 kB 2.5 MB/s eta 0:00:01
Requirement already satisfied: matplotlib in
/opt/anaconda3/lib/python3.9/site-packages (from wordcloud) (3.5.1)
Requirement already satisfied: numpy>=1.6.1 in
/opt/anaconda3/lib/python3.9/site-packages (from wordcloud) (1.21.5)
Requirement already satisfied: pillow in /opt/anaconda3/lib/python3.9/site-
packages (from wordcloud) (9.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (1.3.2)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: pyparsing>=2.2.1 in
```

```
/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (3.0.4)
Requirement already satisfied: packaging>=20.0 in
/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: cycler>=0.10 in
/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.9/site-
packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.2.2
```

1.5.1 Get the spam & ham messages

[24]:
```
spam = df[df['label']=='spam']['sms'].str.cat(sep=', ')
ham = df[df['label']=='ham']['sms'].str.cat(sep=', ')
```

1.5.2 Initialize the word cloud

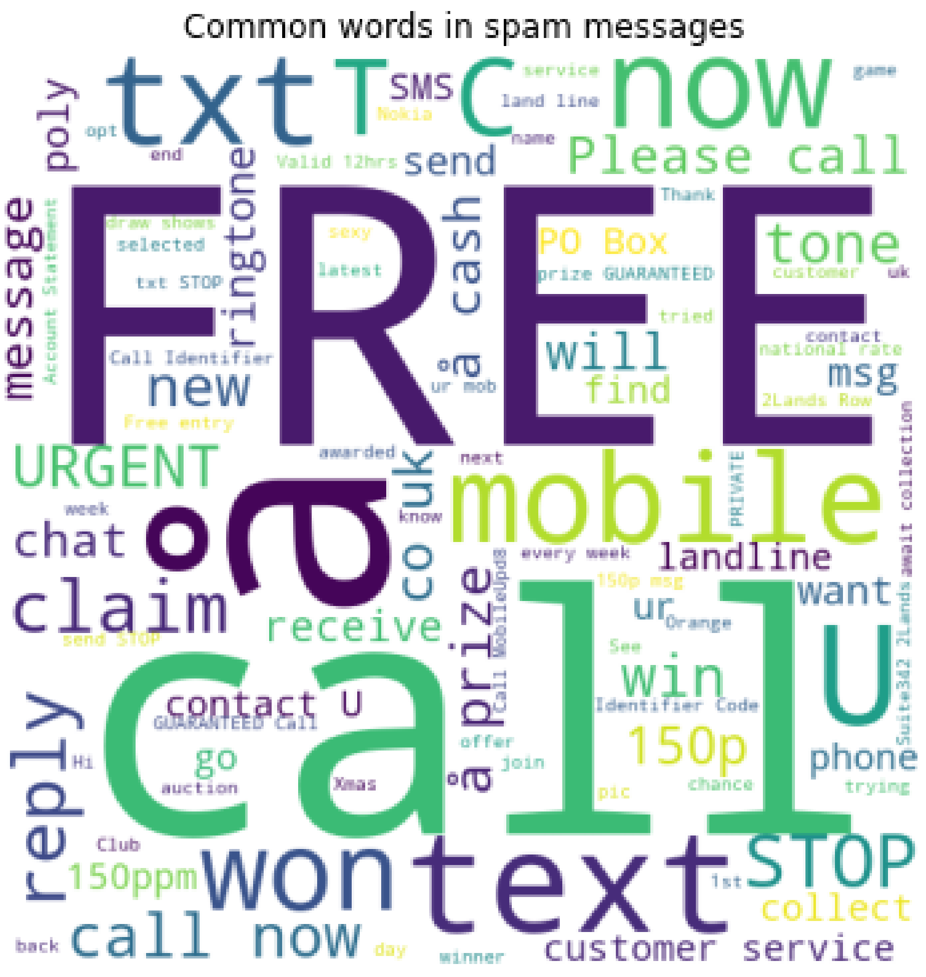
[25]:
```
wc = WordCloud(width = 500, height = 500, min_font_size = 10, background_color␣
 ↪='white')
```

1.5.3 Generate the world clouds for each type of message

[26]:
```
spam_wc = wc.generate(spam)
```

1.5.4 plot the world cloud for spam

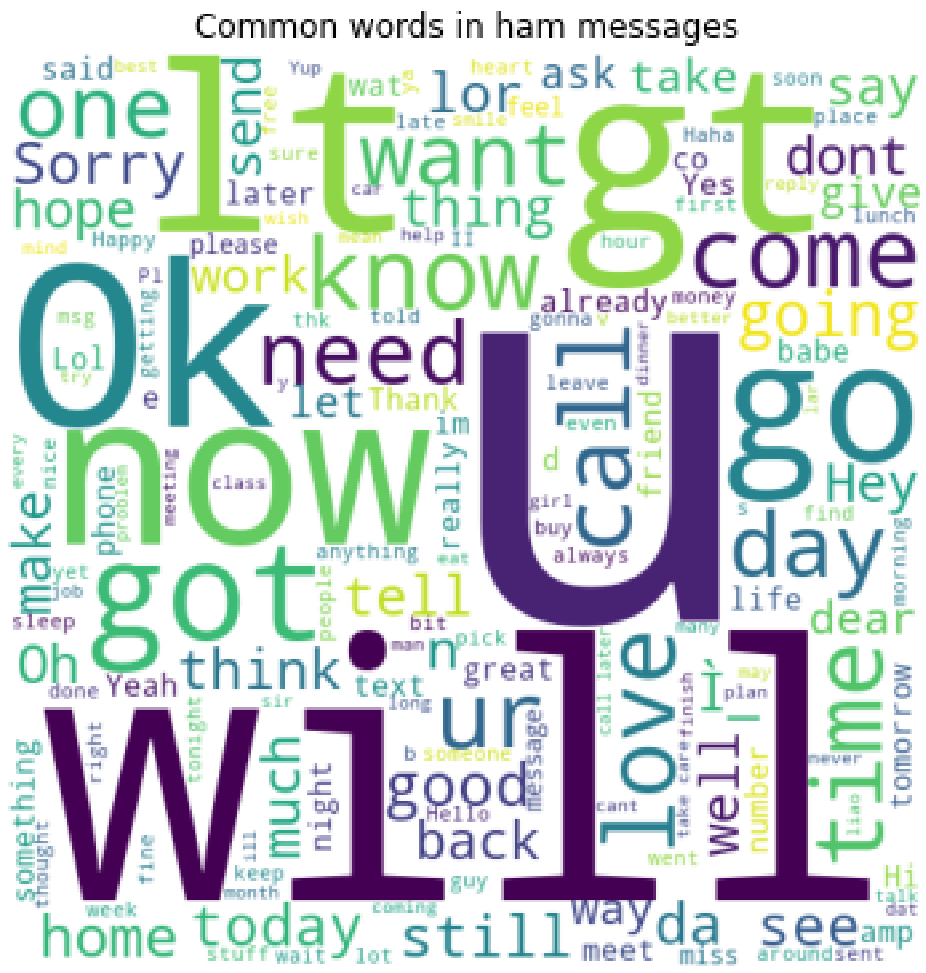
[27]:
```
plt.figure(figsize = (5, 5), facecolor = None)
plt.imshow(spam_wc)
plt.axis("off")
plt.title("Common words in spam messages")
plt.tight_layout(pad = 0)
plt.show()
```



1.5.5 plot the world cloud for ham

```
[29]: ham_wc = wc.generate(ham)
      plt.figure(figsize = (5, 5), facecolor = None)
      plt.imshow(ham_wc)

      plt.axis("off")
      plt.title("Common words in ham messages")
      plt.tight_layout(pad = 0)
      plt.show()
```


Common words in ham messages
u
will
Ok
lt
gt
now
go
got
want
know
come
call
need
one
Sorry
hope
going
love
time
day
ur
good
think
still
today
home
da
see
back
tell
work
thing
dont
say
take
ask
lor
send
give
Hey
much
well
make
Oh
way
dear
life
night
great
text
tomorrow
number
Yeah
let
Thank
Lol
later
please
already
babe
friend
really
something
message
Hello
phone
amp
meet
miss
Hi
Yes
said
wat
feel
im
anything
always
leave
even
buy
girl
find
pick
bit
someone
keep
guy
went
cant
plan
around
sent
lot
wait
stuff
week
month
coming
talk
dat
fine
right
tonight
sleep
yet
job
done
sir
every
nice
class
meeting
problem
people
morning
Happy
mind
wish
sure
free
best
Yup
heart
soon
place
smile
late
Haha
car
first
reply
lunch
hour
help
mean
money
told
thk
msg
gonna
better
dinner
getting
try
Pl
co
e
d
n
s
b
v
y
lar
liao
never
may
long
man
eat
finish
call later
take care
thought
ill