Readme file

I. **NLTK Library-import nltk**

First we import important libraries such as nltk, regex.
Then we will use nltk.tokenize.
from nltk.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer

2. **Methodology**

So in the methodology section, we have used various forms of the regex, to solve each part. Firstly we will open the data frame using the panda's data frame, and then we will have the operations on the data frame .

```
# to read
df = pd.read_csv('a01_spam.csv')
df
```

3. **Assumptions and methodology for each part**

a) For the first part, I split each line by white spaces using

   i) msg = msg.split()

   Then I applied the regex:

   r = re.findall(r'^[AEIOUaeiou][^^]*',word)

   It will match all the matches which start from vowels.
   I have assumed that matches can have special characters in the strings like :
   <>?:"{}!@#$%^&*()_+-=~:'
   [^^]* means it will count every string but doesn't count or match strings.

   For consonants, I split each line by white spaces using

   ii) msg = msg.split()

   Then I applied the regex:
   r = re.findall(r'^[B-DF-HJ-NP-TV-Zb-df-hj-np-tv-z][^^]*',word)

   It will match all the matches which start from consonants.I have assumed that matches can have special characters in the strings like :
   <>?:"{}!@#$%^&*()_+-=~:"
   [^^]* means it will count every string but doesn't count or match strings.

b) For the second part, I split each line by white spaces using

    i) msg = msg.split()

Then I applied the regex:

r=re.findall(r'^[^(a-z^)]*[A-Z!@#$%&*()_+~`.,/;\]{}[><?:"-=0-9]*[A-Z]+[A-Z!@#$%&*()_+~`.,/;\]{}[><?:"-=0-9]*[^(a-z^)]*$',word)

It will match all the matches which are capitalized words.
I have assumed that matches can have special characters in the strings like :
A-Z!@#$%&*()_+~`.,/;\]{}[><?:"-=0-9
And It doesn't contain any a to z.
[A-Z!@#$%&*()_+~`.,/;\]{}[><?:"-=0-9]* means it will count every string but doesn't count or match strings.
[A-Z]+ shows at least one occurrence of capitalized words.

For calculating the percentage of the spam words i have gone with the formula is
=(Spam words /total spam words)*100

For calculating the percentage of the ham words i have gone with the formula is
=(Spam words /total spam words)*100

c) For the third part, I split each line by white spaces using

    ii) msg = msg.split()

Then I applied the regex:

r=re.findall(r'[A-Za-z0-9._+-]+@[A-Za-z0-9._+-]+\.[A-Za-z0-9._+-]+\.?[A-Za-z0-9._+-]+',word)

It will match all the matches which are mail id's.
I have assumed that user name can have special characters in the strings like :
._+-

[A-Za-z0-9._+-]+ shows at least one occurrence of a user name string..
@ also included in mail ids.
For calculating the percentage of the spam words i have gone with the formula is
=(Spam words /total spam words)*100

For calculating the percentage of the ham words i have gone with the formula is
=(Spam words /total spam words)*100

Total words in ham : 69047
mails in ham : 1
mails % in ham : 0.0014482888467275913

Total words in spam : 17788
mails in spam : 6
mails % in spam : 0.033730604902181244

Total messages in ham and spam : 86835
Total mails in ham and spam : 7
Overall mails % 0.008061265618702136

Separately I find the both % s:
For ham messages I got this mails:
['info@ringtoneking.co.uk', 'tddnewsletter@emc1.co.uk', 'info@txt82228.co.uk', 'Dorothy@kiefer.com', 'msg+ticket@kiosk.Valid', 'customersqueries@netvision.uk.com']

For spam messages I got this mails:
['yijue@hotmail.com']

For the third part, I split each line by white spaces using

iii) msg = msg.split()

Then I applied the regex:

r=re.findall(r'[0-9]?[0-9]{10}|[(]?[0-9]{4,5}[)]?-?[(]?[0-9]{3,4}[)]?-?[(]?[0-9]{4,5}[)]?',word)

It will match all the matches which are phone numbers.
I have assumed that numbers can have only 0-9 digits and contains -.

For that I used -?. It can matches number like '0207-083-6089'

For calculating the percentage of the spam words i have gone with the formula is
=(Spam words /total spam words)*100

For calculating the percentage of the ham words i have gone with the formula is
=(Spam words /total spam words)*100

Then I calculated the respective percentages :

```
Total words in ham :  69047
phone in ham :  1
phone % in ham :  0.0014482888467275913

Total words in spam :  17788
phone in spam :  437
phone % in spam :  2.456712390375534

Total phones in ham and spam : 86835
Total phones in ham and spam : 438
Overall phones %  0.5044049058559337
```

We got our result as :

```
Total words in ham :  69047
phones and mails in ham :  2
phones and mails % in ham :  0.0028965776934551826

Total words in spam :  17788
phones and mails in spam :  443
phones and mails % in spam :  2.4904429952777156

Total words in ham and spam : 86835
Total phones and mails in ham and spam : 445
Overall phones %  0.5124661714746358
```

d) To finding the monetary values in the message we have used the regex :

r=re.findall(r'[$£][0-9]+[.,]?[0-9]*[,]?[0-9]*',word)

Firstly, I split each line by white spaces using
msg = msg.split()
Then I applied the regex:

```
                    r=re.findall(r'[$£][0-9]+[.,]?[0-9]*[,]?[0-9]*',word)
```

It will match all the matches which are monetary values
I also take decimal values.
And also assume that there are two monetary quantities, dollar and euro.

For calculating the percentage of the spam words i have gone with the formula is
=(Spam words /total spam words)*100

For calculating the percentage of the ham words i have gone with the formula is
=(Spam words /total spam words)*100

```
Total words in ham :  69047
monetary words in ham :  17
monetary words % in ham :  0.02462091039436905

Total words in spam :  17788
monetary words in spam :  321
monetary words % in spam :  1.8045873622666968

Total messages in ham and spam : 86835
Total monetary words in ham and spam : 338
Overall monetary %  0.38924396844590314
```

e) To finding the emoticons in the message we have used the regex :

```
tk = RegexpTokenizer('[@*]?[|:;=X^]+[-"=*]?[DPp0*Oo>^)/\(@]+',gaps=False)
```

It will matches emoticons like :

```
'://', ':)', 'Xo', ':)', ':)', ':)', ':)', ':)', ':)', ':)', ':)', ':)', ':)', ';D', ';D', ';D', ':)', ':)', ':)', ':)', ':)',
':-)', ':-)', '://', ':)', ':(', ':)', ':)', ':0', ':-)', ':-D', ':)', ':)', ':-)', ':-)', ':D', '=D', ':)', ':)', ';:(',
':-)', ':)', ':)', ':)', ':)', ':)', ':)', '://', ';)', ':)', ':)', ':-D', ':)', '://', ':)', ':-)', ':-)', ';)', ';D', ':)', ':)',
'=D', ':)', ':)', ':-)', ':)', ':)', '://', ':)', ':)', ':)', ':)', ';D', '://', ':)', ':-)', ':/', ':)'
```

f) For clitics I used regex :
r = re.findall(r"\w+[‘"’]\w+",word)
I assumed that there can be numbers before [‘"’] and obviously words also that’s why I used \w+. And followed by \w+ as well.
So, It matches like :

```
"08452810075over18's", "don't", "it's", "week's", "I'd", "I'm", "don't", "I've", "I've",
"i'm", 'I‘m', "I'm", "I'm", "mom's", "I'm", "we're", "I'll", "there's"
```

g) First I take input then find the messages and number of messages starting with a given word as an input.
I used the following regex :
r = re.findall(r"^"+inp+'[a-zA-Z!@#$^%&*()_+~`.,/;\]{}[><?:"-=0-9]*',msg[0])
Msg[0] indicates, I took every first word of the message and then applied the above regex.
[a-zA-Z!@#$^%&*()_+~`.,/;\]{}[><?:"-=0-9]* followed by given input It matches all the things after that string (first one)
  Also handle the case :

```
if len(start_msgs)==0 :
   print("No results found !")
```

If we have no matches the No results will be found.

h) First I take input then find the messages and number of messages ending with a given word as an input.
I used the following regex :
   r =
re.findall(r'[a-zA-Z!@#$^%&*()_+~`.,/;\]{}[><?:"-=0-9]*'+inp+'[.!?;]*$',msg[len(msg)-1
])

Msg[len(msg)-1] indicates, I took every first word of the message and then applied the above regex.

[a-zA-Z!@#$^%&*()_+~`.,/;\]{}[><?:"-=0-9]* followed by given input It matches all the things before that string (last one)

Also handle the case :

```
if len(last_msgs)==0 :
    print("No results found !")
```

If we have no matches the No results will be found.

i) In last part I combined all the cases :

For each words in messages

I find three tuple values, like [a,b,c]

Where a = no. of capitalized

b = number of contacts and mail id

c =no. Of monetary values

Then comparing all these values with

```
if pc >= glo_money_spam:
        # spam
        sh_mon=0
    else:
        sh_mon=1
    if pa >= glo_cap_spam:
        sh_cap=0
    else:
        sh_cap=1
    if pb >= glo_mail_spam:
        sh_mail=0
    else:
        sh_mail=1
```

This will predict all the values for all three queries.

And later i took and operations:

And find other cases like ([only 2], [only 3], [only 4], [2,3], [2,4], [3,4], or [2,3,4])

And will get the results.