In [ ]:

In [ ]:

In [ ]:

In [1]:
```
import nltk
```

```
C:\Users\my pc\anaconda3\lib\site-packages\numpy\_distributor_init.py:30: UserWarning: loaded more than 1 DLL from .
libs:
C:\Users\my pc\anaconda3\lib\site-packages\numpy\.libs\libopenblas.FB5AE2TYXYH2IJRDKGDGQ3XBKLKTF43H.gfortran-win_amd
64.dll
C:\Users\my pc\anaconda3\lib\site-packages\numpy\.libs\libopenblas64__v0.3.23-gcc_10_3_0.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
C:\Users\my pc\anaconda3\lib\site-packages\scipy\__init__.py:155: UserWarning: A NumPy version >=1.18.5 and <1.25.0
is required for this version of SciPy (detected version 1.25.0
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

PORTER STEMMER : NLTK

PAGE 51

In [2]:
```
from nltk.stem.porter import PorterStemmer
```

In [3]:
```
p_stemmer = PorterStemmer()
```

Here, i'm taking 'Words' - left hand variable :

In [4]:
```
words = ['run','runner','ran','runs','easily','fairly']
```

In [5]:
```
# Here, 'runs' changed to 'run', 'easily' changed to 'easili' and also fairly to fiarli :

for word in words:
    print(word+ '----->' +p_stemmer.stem(word))
```

```
run----->run
runner----->runner
ran----->ran
runs----->run
easily----->easili
fairly----->fairli
```

SNOWBALL STEMMER :

PAGE 52

In [6]:
```
from nltk.stem.snowball import SnowballStemmer
```

In [7]:
```
# Here, NLTK -> Accepts Multiple Languages. So, we need to mention 'Language' :
# SnowBall is an a Bit Advanced :

s_stemmer = SnowballStemmer(language = 'english')
```

In [8]:
```
# Here, 'runs' became 'run' and 'fairly' became 'fair' :
# Means, We are seeing something bit Adavanced :

for word in words:
    print(word + ' ----> ' + s_stemmer.stem(word))
```

```
run ----> run
runner ----> runner
ran ----> ran
runs ----> run
easily ----> easili
fairly ----> fair
```

LEMMATIZATION : 'SPACY' :

PAGE 52

In [9]:
```
# The Advanced one we call it as an a 'LEMMATIZATION' :
# it is a bit Advanced One :

# NLTK : 1) PorterStemmer, 2) SnowballStemmer.
# SPACY : 1) Lemmatization.
```

In [10]:
```
import spacy
```

In [11]:
```
# Loading English Language :

nlp = spacy.load('en_core_web_sm')
```

In [12]:
```
# Now Declaring 'nlp' of 'u' string Data :
```

In [13]:
```
doc1 = nlp(u"I am running in a race because i love to run since i ran runs running")
```

In [14]:
```
# Compared to NLTK, 'SPACY' is an Advanced Library :


for token in doc1:
    print(token.text, '\t', token.pos_, '\t', token.lemma, '\t', token.lemma_)
```

```
I        PRON    4690420944186131903     I
am       AUX     10382539506755952630    be
running          VERB    12767647472892411841    run
in       ADP     3002984154512732771     in
a        DET     11901859001352538922    a
race     NOUN    8048469955494714898     race
because          SCONJ   16950148841647037698    because
i        PRON    4690420944186131903     I
love     VERB    3702023516439754181     love
to       PART    3791531372978436496     to
run      VERB    12767647472892411841    run
since    SCONJ   10066841407251338481    since
i        PRON    4690420944186131903     I
ran      VERB    12767647472892411841    run
runs     NOUN    12767647472892411841    run
running          VERB    12767647472892411841    run
```

In [15]:
```
# In the above Sum :

# token.text -> means, I
# token.pos_ -> means, PRON,AUX,ADP,VERB,DET.....
# token.lemma -> means, It's Adjecent Word, means for the particular word 'I'. Means, One particular Number is going t
# token.lemma_ -> Means, _Underscore means, What is actual data represented.
#              eg: This is an a Number 4690420944186131903 and 'I' is it's Original Data. Inside this Number - 'I' is it
```

In [16]:
```
# Compared to 'NLTK', 'SPACY' is an a Advanced Library :
```

In [ ]:

```

```

In [17]:

```
# Vocabulary means, Something like 'Nearest Words'. Means, Matching an a Nearest Word (or)
# Identifying the given words in the given documents. like, eg: 'Solar' - one particular word.
# Means, I have an a Document like 2500 Words, In this particular 2500 Words - How many times 'Solar' has been Repeate
# And Where it has been repeated,,, We need to identify in these 2500 Words.

# These particular things from where we are going to be seen here means, Exactly from "PHRASE MATCHER" :
```

PHRASE MATCHER :

PAGE 55

In [18]:

```
# First i need to import 'Spacy' :

import spacy
```

In [19]:

```
from spacy.matcher import Matcher
```

In [20]:

```
nlp = spacy.load("en_core_web_sm")
```

In [21]:

```
matcher = Matcher(nlp.vocab)
```

In [22]:

```
# Now i'm taking an a different pattern like,
# Here, i declared 'IS_PUNCTUATION' in {} DICTIONARY FORMAT :
# page 56

pattern1 = [{'LOWER' : 'solarpower'}]
pattern2 = [{'LOWER' : 'solar'},{'IS_PUNCT' : True}, {'LOWER' : 'POWER'}]
pattern3 = [{'LOWER' : 'solar'}, {'LOWER' : 'power'}]
```

In [23]:
```
# 'None' is a key - Here we used, but not working :

matcher.add('SolarPower',[pattern1, pattern2, pattern3])
```

In [24]:
```
doc2 = nlp(u'The Solar Power industry continues to grow a solarpower increases, solar-power is excellent')
```

In [25]:
```
# Already Matcher we have declared, Now we need to take 'found-matcher'- 'f' small letter
```

In [26]:
```
found_matcher = matcher(doc2)
```

In [27]:
```
# This is what exactly, we are working here :
# This is called Matcher ID, Has 'Sarting and Ending' for each Matcher ID :

print(found_matcher)
```

```
[(8656102463236116519, 1, 3), (8656102463236116519, 8, 9)]
```

In [28]:
```
# Now i want to See What Words we have Matched for that, Here i'm Writting particular 'for loop' :
```

In [29]:
```
for match_id, start, end in found_matcher:
    string_id = nlp.vocab.strings[match_id]
    spam = doc2[start : end]
    print(match_id, string_id, start, end, spam.text)
```

```
8656102463236116519 SolarPower 1 3 Solar Power
8656102463236116519 SolarPower 8 9 solarpower
```

In [ ]:
```
```

In [ ]:
```
```

In [30]:
```
from spacy.matcher import PhraseMatcher
```

In [31]:
```
matcher = PhraseMatcher(nlp.vocab)
```

In [32]:
```
phrase_list = ['economics','money','Political','Reagan','tax']
```

In [33]:
```
# Now we declare the path of the DataSet :
```

In [34]:
```
with open("D:/P J PRASANTH PYTHON/DataS/Reaganomics.txt") as f:
    doc4 = nlp(f.read())
```

In [35]:
```
phrase_patterns = [nlp(text) for text in phrase_list]
```

In [36]:
```
phrase_patterns
```

Out[36]:
```
[economics, money, Political, Reagan, tax]
```

In [37]:
```
matcher.add('EconMatcher',None,*phrase_patterns)
found_matchers = matcher(doc4)
```

In [38]:

```python
# These many words, it has been Matched :

found_matchers
```

```
 (3680293220734633682, 1429, 1430),
 (3680293220734633682, 1434, 1435),
 (3680293220734633682, 1448, 1449),
 (3680293220734633682, 1480, 1481),
 (3680293220734633682, 1486, 1487),
 (3680293220734633682, 1532, 1533),
 (3680293220734633682, 1544, 1545),
 (3680293220734633682, 1551, 1552),
 (3680293220734633682, 1559, 1560),
 (3680293220734633682, 1586, 1587),
 (3680293220734633682, 1614, 1615),
 (3680293220734633682, 1715, 1716),
 (3680293220734633682, 1720, 1721),
 (3680293220734633682, 1735, 1736),
 (3680293220734633682, 1758, 1759),
 (3680293220734633682, 1810, 1811),
 (3680293220734633682, 1838, 1839),
 (3680293220734633682, 1918, 1919),
 (3680293220734633682, 1948, 1949),
```

In [39]:
```
for match_id, start, end in found_matchers:
    string_id = nlp.vocab.strings[match_id]
    span = doc4[start:end]
    print(match_id,string_id,start,end,span.text)
```

```
3680293220734633682 EconMatcher 216 217 Reagan
3680293220734633682 EconMatcher 229 230 economics
3680293220734633682 EconMatcher 234 235 economics
3680293220734633682 EconMatcher 239 240 economics
3680293220734633682 EconMatcher 245 246 Reagan
3680293220734633682 EconMatcher 256 257 economics
3680293220734633682 EconMatcher 262 263 Reagan
3680293220734633682 EconMatcher 287 288 tax
3680293220734633682 EconMatcher 291 292 tax
3680293220734633682 EconMatcher 300 301 money
3680293220734633682 EconMatcher 404 405 Reagan
3680293220734633682 EconMatcher 414 415 Reagan
3680293220734633682 EconMatcher 455 456 Political
3680293220734633682 EconMatcher 465 466 money
3680293220734633682 EconMatcher 543 544 money
3680293220734633682 EconMatcher 565 566 Reagan
3680293220734633682 EconMatcher 576 577 Reagan
3680293220734633682 EconMatcher 580 581 tax
3680293220734633682 EconMatcher 587 588 tax
```

In [ ]:

In [ ]:

In [ ]:

In [40]:
```
print(nlp.Defaults.stop_words)
```

```
{'regarding', 'first', 'would', 'sixty', 'wherein', 'they', 'herein', 'who', "'re", 'back', 'hereby', 'them', 'fort
y', 'three', 'whole', "n't", 'now', 'am', 'since', '‘ll', 'themselves', 'was', 'on', 'put', 'may', 'latterly', 'las
t', 'enough', 'few', 'just', 'seeming', 'four', 'not', 'sometimes', 'well', 'but', 'me', 'might', "'d", 'fifteen',
'whereby', 'nobody', '’ve', 'nothing', 'such', 'too', 'our', "'ll", 'my', 'when', 'some', 'between', 'never', 'nowhe
re', 'eleven', 'is', 'almost', 'seemed', 'himself', '‘re', 'toward', 'empty', 'doing', 'towards', 'beside', 'beside
s', 'are', 'alone', 'to', 'whether', '’s', 'else', 'already', 'onto', 're', 'most', 'again', 'thru', 'made', 'everyt
hing', 'thereupon', 'up', 'what', 'cannot', 'his', 'same', 'several', 'there', 'throughout', 'whatever', 'very', 'ab
out', 'of', 'n‘t', 'that', 'call', 'next', 'have', 'amount', 'during', 'two', 'anyhow', 'indeed', 'the', 'wherever',
'through', 'within', 'whom', 'mostly', "'s", 'more', 'except', 'top', 'can', 'you', 'by', 'along', 'under', '‘s', 'w
ere', 'all', 'and', 'an', 'here', 'using', '‘ve', 'out', "'m", 'however', 'because', 'amongst', 'do', '‘d', 'its',
'a', 'him', 'even', 'say', 'upon', 'anyone', 'becomes', 'ca', 'for', 'somewhere', 'it', 'somehow', 'she', 'least',
'become', 'done', 'he', 'any', 'should', 'whose', 'really', 'everywhere', 'which', 'used', 'above', 'formerly', 'te
n', 'whenever', 'bottom', 'give', 'various', 'before', 'make', 'though', 'this', '‘m', 'hereupon', 'others', 'yourse
lves', 'as', 'been', 'myself', 'why', 'at', 'we', 'otherwise', 'together', 'unless', 'across', 'anyway', 'either',
'moreover', 'nine', 'neither', 'could', 'thence', 'must', 'always', 'does', 'yourself', 'part', 'thus', 'seems', '’l
l', 'whereafter', 'beyond', 'much', 'serious', 'namely', 'go', 'third', 'therefore', 'until', 'both', 'against', 'i
n', '’m', 'her', 'eight', 'keep', 'latter', 'often', 'own', 'move', 'twelve', 'please', 'twenty', 'itself', 'hence',
'someone', 'although', 'side', 'fifty', 'hundred', 'after', 'n’t', 'something', 'below', 'thereafter', 'whereas', 'o
r', 'be', 'front', 'being', 'each', 'one', 'these', 'so', 'noone', 'therein', 'meanwhile', 'their', 'thereby', 'als
o', "'ve", 'herself', 'than', 'quite', 'had', 'has', 'around', 'ever', 'did', 'elsewhere', 'rather', 'everyone', 'on
ce', 'other', 'without', 'every', 'afterwards', 'became', 'due', 'six', 'among', 'us', 'via', 'with', '’d', 'perhap
s', 'i', 'then', 'whoever', 'only', 'beforehand', '’re', 'your', 'yet', 'see', 'further', 'another', 'from', 'show',
'ours', 'will', 'anywhere', 'if', 'those', 'whereupon', 'name', 'seem', 'where', 'per', 'take', 'none', 'off', 'your
s', 'over', 'former', 'hereafter', 'hers', 'anything', 'nor', 'behind', 'whither', 'nevertheless', 'mine', 'ourselve
s', 'less', 'many', 'becoming', 'five', 'get', 'still', 'whence', 'how', 'while', 'sometime', 'into', 'full', 'no',
'down'}
```

In [41]:
```
len(nlp.Defaults.stop_words)
```

Out[41]:
```
326
```

In [42]:
```
nlp.vocab['is'].is_stop
```

Out[42]:
```
True
```

In [43]:
```
nlp.vocab['mystery'].is_stop
```
Out[43]:
```
False
```

In [44]:
```
# im sms "btw" = by the way <shortcut>
# PAGE 64
```

In [45]:
```
nlp.Defaults.stop_words.add('btw')
```

In [46]:
```
nlp.vocab['btw'].is_stop = True
```

In [47]:
```
len(nlp.Defaults.stop_words)
```
Out[47]:
```
327
```

In [48]:
```
nlp.vocab['beyond'].is_stop
```
Out[48]:
```
True
```

In [102]:
```
# Now

nlp.vocab['beyond'].is_stop = False
```

In [103]:
```
nlp.vocab['beyond'].is_stop
```
Out[103]:
```
False
```

In [ ]:
```

```

Parts of Speech :

In [104]:
```
# That's exactly what spacy is designed to do : we put in row text , and get back a Doc(Document) objects,
# that comes with a Variety of 'Annotations'
```

In [136]:
```
import numpy as np
```

In [137]:
```
import pandas as pd
```

In [156]:
```
df = pd.read_csv("D:/P J PRASANTH PYTHON/DataS/SMSSpamCollection.tsv",sep= '\t')
```

In [157]:
```
df
```

Out[157]:

```
             ham  Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
   0         ham                                                                                Ok lar... Joking wif u oni...
   1        spam                                                              Free entry in 2 a wkly comp to win FA Cup fina...
   2         ham                                                            U dun say so early hor... U c already then say...
   3         ham                                                            Nah I don't think he goes to usf, he lives aro...
   4        spam                                                             FreeMsg Hey there darling it's been 3 week's n...
 ...         ...                                                                                                          ...
5566        spam                                                            This is the 2nd time we have tried 2 contact u...
5567         ham                                                                         Will ü b going to esplanade fr home?
5568         ham                                                            Pity, * was in mood for that. So...any other s...
5569         ham                                                           The guy did some bitching but I acted like i'd...
5570         ham                                                                                   Rofl. Its true to its name

5571 rows × 2 columns
```

In [158]:
```
df.head()
```

Out[158]:

	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
0	ham	Ok lar... Joking wif u oni...
1	spam	Free entry in 2 a wkly comp to win FA Cup fina...
2	ham	U dun say so early hor... U c already then say...
3	ham	Nah I don't think he goes to usf, he lives aro...
4	spam	FreeMsg Hey there darling it's been 3 week's n...

In [159]:
```
len(df)
```

Out[159]:
```
5571
```

In [160]:
```
df['ham'].unique()
```

Out[160]:
```
array(['ham', 'spam'], dtype=object)
```

In [161]:
```
df['ham'].value_counts()
```

Out[161]:
```
ham     4824
spam     747
Name: ham, dtype: int64
```

In [163]:
```
import nltk
```

In [165]:
```
messages = [line.rstrip() for line in open('D:/P J PRASANTH PYTHON/DataS/SMSSpamCollection.tsv')]
```

In [166]:
```
print(len(messages))
```

```
5574
```

In [167]:
```
# page 65
```

In [168]:
```
# No Space between ('\t\n')

for message in enumerate(messages[:10]):
    print(message)
    print('\t\n')
```

```
(0, 'ham\tGo until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore w
at...')


(1, 'ham\tOk lar... Joking wif u oni...')


(2, "spam\tFree entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry que
stion(std txt rate)T&C's apply 08452810075over18's")


(3, 'ham\tU dun say so early hor... U c already then say...')


(4, "ham\tNah I don't think he goes to usf, he lives around here though")


(5, "spam\tFreeMsg 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, Â£1.50 to rcv")


(6, 'ham\tEven my brother is not like to speak with me. They treat me like aids patent.')


(7, "ham\tAs 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")


(8, 'spam\tWINNER!! As a valued network customer you have been selected to receivea Â£900 prize reward! To claim cal
l 09061701461. Claim code KL341. Valid 12 hours only.')


(9, 'spam\tHad your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Fr
ee! Call The Mobile Update Co FREE on 08002986030')
```

In [169]:
```
messages = pd.read_csv('D:/P J PRASANTH PYTHON/DataS/SMSSpamCollection.tsv', sep = '\t', names = ['Labels','Message']
```

In [170]:
```
messages.head()
```

Out[170]:

	Labels	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 [171]:
```
messages
```

Out[171]:

	Labels	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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

```
In [172]: len(messages)
```

```
Out[172]: 5572
```

```
In [173]: messages.describe()
```

Out[173]:

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

```
In [174]: messages.groupby('Labels').describe()
```

Out[174]:

	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

```
In [175]: messages['length'] = messages['Message'].apply(len)
In [177]: messages.head()
```

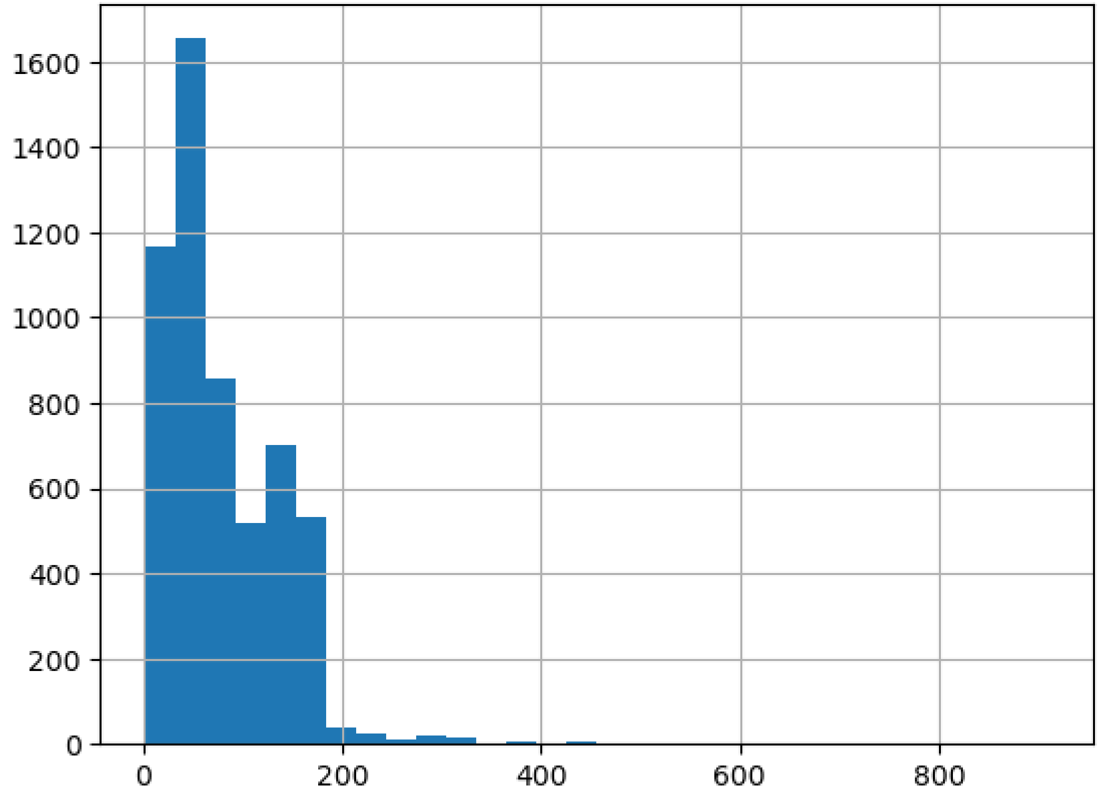
Out[177]:

	Labels	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 [178]: import seaborn as sns
```

In [179]:
```
messages['length'].hist(bins = 30)
```

Out[179]:
```
<AxesSubplot:>
```



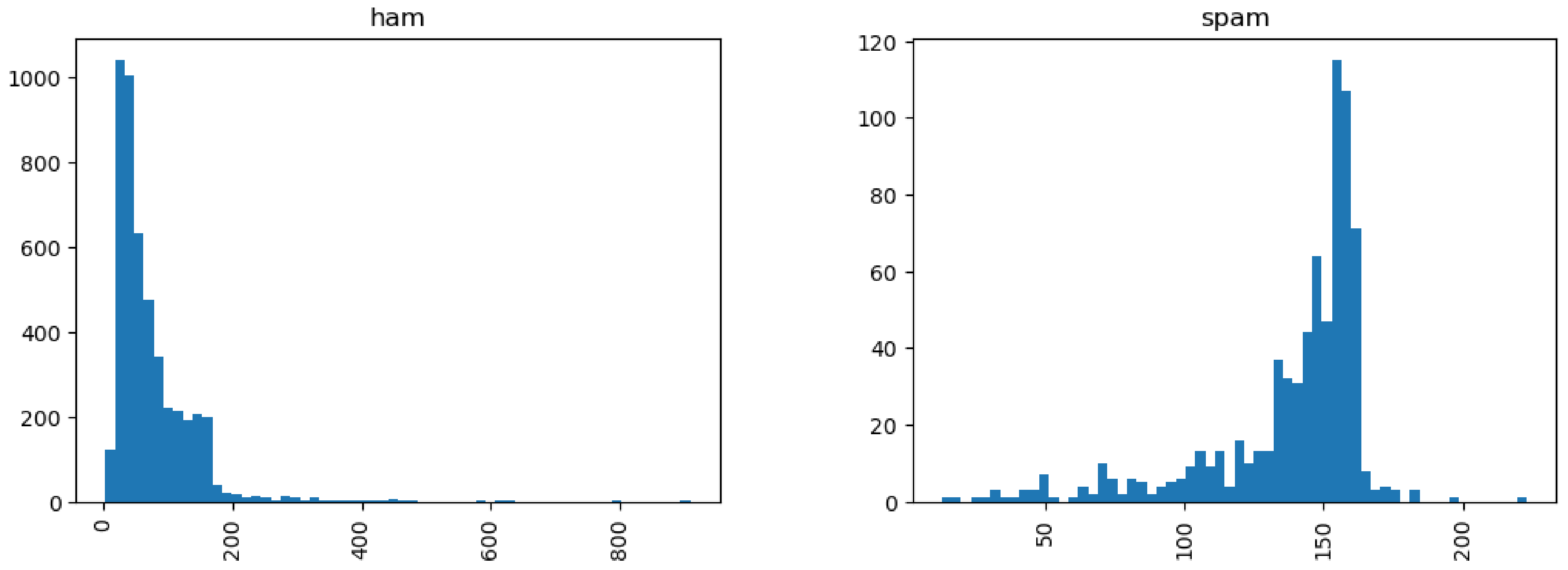
In [180]:
```
messages[messages['length']==910]
```

Out[180]:

	Labels	Message	length
1085	ham	For me the love should start with attraction.i...	910

```
In [181]: messages.hist(column = 'length', by = 'Labels', bins = 60, figsize = (12,4))

Out[181]: array([<AxesSubplot:title={'center':'ham'}>,
                 <AxesSubplot:title={'center':'spam'}>], dtype=object)
```



```
In [182]: import string

In [214]: mess = 'Sample! string, it:is'

In [215]: nopunc = [c for c in mess if c not in string.punctuation]
```

In [216]:
```
nopunc
```

Out[216]:
```
['S',
 'a',
 'm',
 'p',
 'l',
 'e',
 ' ',
 's',
 't',
 'r',
 'i',
 'n',
 'g',
 ' ',
 'i',
 't',
 'i',
 's']
```

In [217]:
```
from nltk.corpus import stopwords
```

In [218]:
```
nltk.download('stopwords')
```

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

Out[218]:
```
True
```

In [219]:
```
stopwords.words('english')
```

```
 'ma',
 'mightn',
 "mightn't",
 'mustn',
 "mustn't",
 'needn',
 "needn't",
 'shan',
 "shan't",
 'shouldn',
 "shouldn't",
 'wasn',
 "wasn't",
 'weren',
 "weren't",
 'won',
 "won't",
 'wouldn',
 "wouldn't"]
```

In [220]:
```
len(nlp.Defaults.stop_words)
```

Out[220]:
```
328
```

In [221]:
```
def text_process(mess):
    nopunc = [char for char in mess if char not in string.punctuation]

    nopunc =''.join(nopunc)

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

In [222]:
```
messages.head(2)
```

Out[222]:

	Labels	Message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29

In [223]:
```
messages['Message'].head(5).apply(text_process)
```

Out[223]:
```
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
```

In [226]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [227]:
```
bow_transformer = CountVectorizer(analyzer = text_process).fit(messages['Message'])
```

In [229]:
```
mess4 = messages['Message'][3]
```

In [230]:
```
mess4
```

Out[230]:
```
'U dun say so early hor... U c already then say...'
```

In [232]:
```
bow4 = bow_transformer.transform([mess4])
```

In [234]:
```
print(bow4)
```

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

In [236]:
```
bow_transformer.get_feature_names()[9554]
```

Out[236]:
```
'say'
```

In [237]:
```
message_bow = bow_transformer.transform(messages['Message'])
```

In [238]:
```
from sklearn.feature_extraction.text import TfidfTransformer
```

In [239]:
```
tfidf_transformer = TfidfTransformer().fit(message_bow)
```

In [240]:
```
messages_tfidf = tfidf_transformer.transform(message_bow)
```

In [241]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [243]:
```
spam_detect_model = MultinomialNB().fit(messages_tfidf, messages['Labels'])
```

In [244]:
```
all_pred = spam_detect_model.predict(messages_tfidf)
```

In [247]:
```
all_pred
```

Out[247]:
```
array(['ham', 'ham', 'spam', ..., 'ham', 'ham', 'ham'], dtype='<U4')
```

```
In [ ]: # msg_train, msg_test, label_train, label_test = train_test_split(messages['Message'], messages['Labels'],
        #test_size= 0.33, random_state = 101)
```

```
In [249]: from sklearn.model_selection import train_test_split
```

```
In [250]: msg_train, msg_test, label_train, label_test = train_test_split(messages['Message'], messages['Labels'],
                                                                            test_size= 0.33, random_state = 101)
```

```
In [251]: from sklearn.pipeline import Pipeline
```

```
In [252]: pipeline = Pipeline([
              ('bow',CountVectorizer(analyzer = text_process)),
              ('tfidf',TfidfTransformer()),
              ('mc',MultinomialNB())
          ])
```

```
In [253]: pipeline.fit(msg_train, label_train)

Out[253]: Pipeline(steps=[('bow',
                           CountVectorizer(analyzer=<function text_process at 0x000001BF66561820>)),
                          ('tfidf', TfidfTransformer()), ('mc', MultinomialNB())])
```

```
In [254]: pred = pipeline.predict(msg_test)
```

```
In [255]: from sklearn.metrics import classification_report
```

In [256]:
```
print(classification_report(label_test,pred))
```

```
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98      1624
        spam       1.00      0.67      0.81       215

    accuracy                           0.96      1839
   macro avg       0.98      0.84      0.89      1839
weighted avg       0.96      0.96      0.96      1839
```

In [ ]: