In [3]:
```
import nltk
```

In [4]:
```
nltk.download_shell()
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Download which package (l=list; x=cancel)?
    Downloading package stopwords to
        C:\Users\AMIT_MERUGU\AppData\Roaming\nltk_data...
      Unzipping corpora\stopwords.zip.
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
```

In [25]:
```
messages = [line.rstrip() for line in open('smsspamcollection/SMSSpamCollection'
```

In [33]:
```
messages[0]
```

Out[33]:
```
'ham\tGo until jurong point, crazy.. Available only in bugis n great world la e
buffet... Cine there got amore wat...'
```

In [29]:
```
print(len(messages))
```

```
5574
```

In [31]:
```
for mess_no,message in enumerate(messages[:10]):
    print(mess_no,message)
    print('\n')
```

```
0 ham   Go until jurong point, crazy.. Available only in bugis n great world la 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 rate)T&C's apply 08452810075over1
8'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 li
ke some fun you up for it still? Tb ok! XxX std chgs to send, Â£1.50 to rcv


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


7 ham   As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' ha
s been set as your callertune for all Callers. Press *9 to copy your friends Call
ertune


8 spam  WINNER!! As a valued network customer you have been selected to receivea
Â£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours o
nly.


9 spam  Had your mobile 11 months or more? U R entitled to Update to the latest c
olour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
```

In [35]:
```
import pandas as pd
```

In [37]:
```
messages = pd.read_csv('smsspamcollection/SMSSpamCollection', sep='\t', names=['
messages.head()
```

Out[37]:

	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 [41]:
```
messages.describe()
```

Out[41]:

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

In [43]:
```
messages.groupby('label').describe()
```

Out[43]:

	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 [45]:
```
messages['length'] = messages['message'].apply(len)
```

In [47]:
```
messages.head()
```

Out[47]:

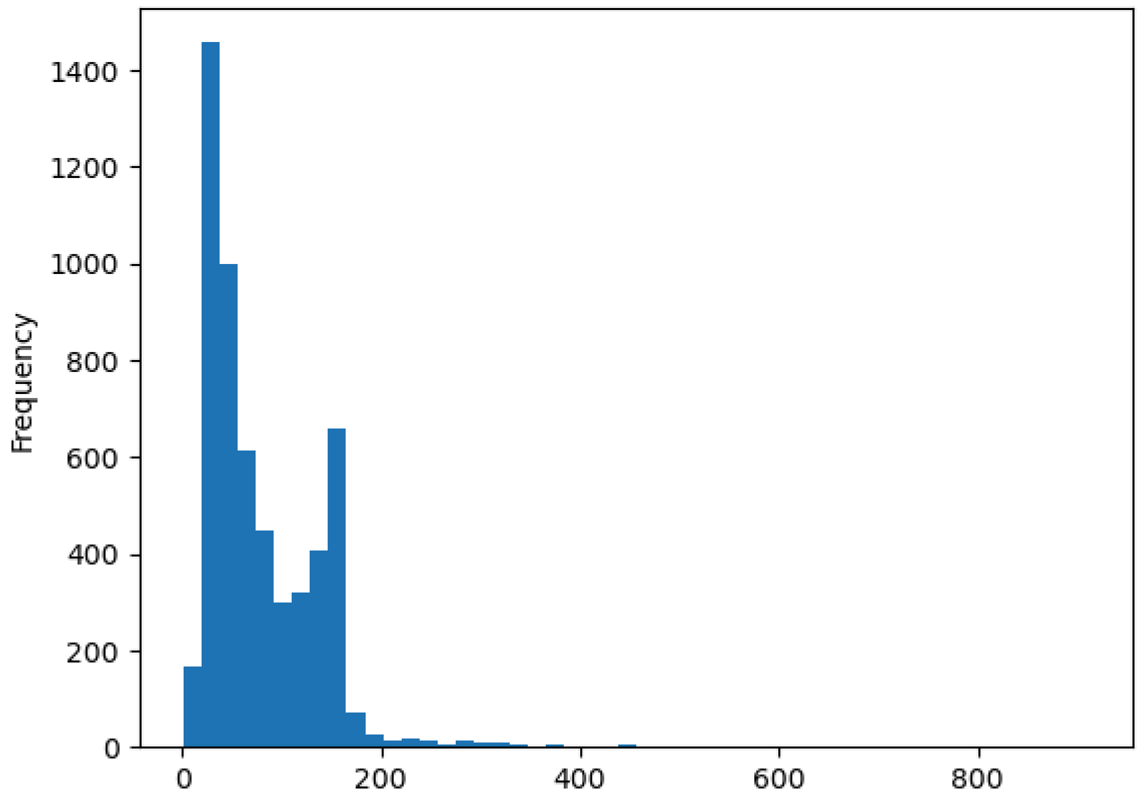
	label	message	length
0	ham	Go until jurong point, crazy.. Available only ...	111
1	ham	Ok lar... Joking wif u oni...	29
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	155
3	ham	U dun say so early hor... U c already then say...	49
4	ham	Nah I don't think he goes to usf, he lives aro...	61

In [49]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [53]:
```
%matplotlib inline
```

In [55]:
```
messages['length'].plot.hist(bins=50)
```

Out[55]:
```
<Axes: ylabel='Frequency'>
```



In [57]:
```
messages['length'].describe()
```

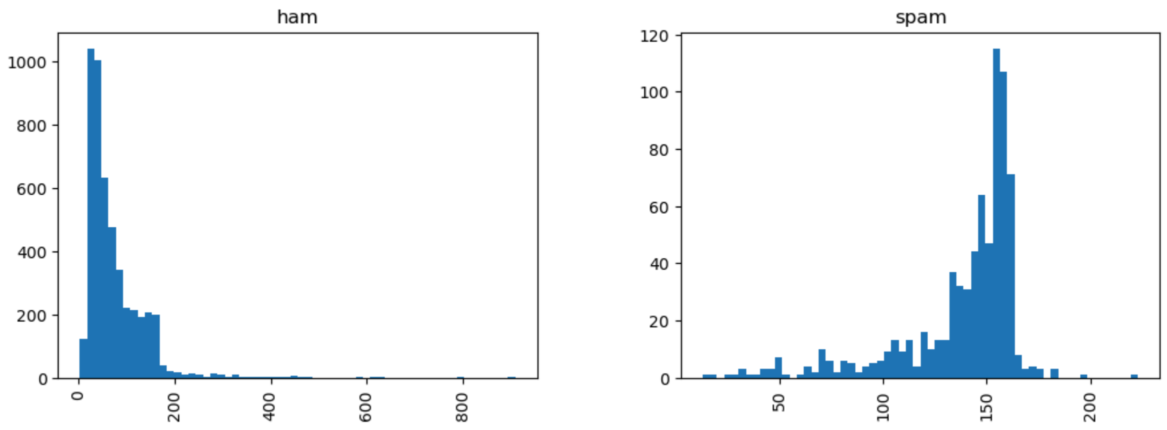
Out[57]:
```
count    5572.000000
mean       80.489950
std        59.942907
min         2.000000
25%        36.000000
50%        62.000000
75%       122.000000
max       910.000000
Name: length, dtype: float64
```

In [63]:
```
messages[messages['length'] == 910]['message'].iloc[0]
```

Out[63]:
```
"For me the love should start with attraction.i should feel that I need her eve
ry time around me.she should be the first thing which comes in my thoughts.I wo
uld start the day and end it with her.she should be there every time I dream.lo
ve will be then when 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 all my 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 craziest things for her.love will be when I do
n't have to proove anyone that my girl is the most beautiful lady on the whole 
planet.I will always be singing praises for her.love will be when I start up ma
king chicken curry and end up makiing sambar.life will be the most beautiful th
en.will get every morning and thank god for the day because she is with me.I wo
uld like to say a lot..will tell later.."
```

In [65]:
```
messages.hist(column='length', by='label', bins=60, figsize=(12,4))
```

Out[65]:
```
array([<Axes: title={'center': 'ham'}>, <Axes: title={'center': 'spam'}>],
      dtype=object)
```



```
In [67]: import string
```

```
In [69]: mess = 'Sample message! Notice: it has punctuation.'
```

```
In [71]: string.punctuation
```

```
Out[71]: '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
In [73]: nopunch = [c for c in mess if c not in string.punctuation]
```

```
In [77]: from nltk.corpus import stopwords
```

```
In [79]: stopwords.words('english')
```

```
Out[79]: ['i',
          'me',
          'my',
          'myself',
          'we',
          'our',
          'ours',
          'ourselves',
          'you',
          "you're",
          "you've",
          "you'll",
          "you'd",
          'your',
          'yours',
          'yourself',
          'yourselves',
          'he',
          'him',
          'his',
          'himself',
          'she',
          "she's",
          'her',
          'hers',
          'herself',
          'it',
          "it's",
          'its',
          'itself',
          'they',
          'them',
          'their',
          'theirs',
          'themselves',
          'what',
          'which',
          'who',
          'whom',
          'this',
          'that',
          "that'll",
          'these',
          'those',
          'am',
          'is',
          'are',
          'was',
          'were',
          'be',
          'been',
          'being',
          'have',
          'has',
          'had',
          'having',
          'do',
          'does',
          'did',
          'doing',
          'a',
          'an',
          'the',
          'and',
          'but',
          'if',
          'or',
          'because',
          'as',
          'until',
          'while',
          'of',
          'at',
          'by',
          'for',
          'with',
          'about',
          'against',
          'between',
          'into',
          'through',
          'during',
          'before',
          'after',
          'above',
          'below',
          'to',
          'from',
          'up',
          'down',
          'in',
          'out',
          'on',
          'off',
          'over',
          'under',
          'again',
          'further',
          'then',
          'once',
          'here',
          'there',
          'when',
          'where',
          'why',
          'how',
          'all',
          'any',
          'both',
          'each',
          'few',
          'more',
          'most',
          'other',
          'some',
          'such',
          'no',
          'nor',
          'not',
          'only',
          'own',
          'same',
          'so',
          'than',
          'too',
          'very',
          's',
          't',
          'can',
          'will',
          'just',
          'don',
          "don't",
          'should',
          "should've",
          'now',
          'd',
          'll',
          'm',
          'o',
          're',
          've',
          'y',
          'ain',
          'aren',
          "aren't",
          'couldn',
          "couldn't",
          'didn',
          "didn't",
          'doesn',
          "doesn't",
          'hadn',
          "hadn't",
          'hasn',
          "hasn't",
          'haven',
          "haven't",
          'isn',
          "isn't",
          '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 [81]:
```
nopunch
```

Out[81]:
```
['S',
 'a',
 'm',
 'p',
 'l',
 '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']
```

In [83]:
```
nopunch = ''.join(nopunch)
```

In [85]:
```
nopunch
```

Out[85]:
```
'Sample message Notice it has punctuation'
```

In [87]:
```
x = ['a' ,'b' ,'c', 'd']
```

In [89]:
```
'++++'.join(x)
```

Out[89]:
```
'a++++b++++c++++d'
```

In [91]:
```
nopunch.split()
```

Out[91]:
```
['Sample', 'message', 'Notice', 'it', 'has', 'punctuation']
```

In [95]:
```
clean_mess = [word for word in nopunch.split() if word.lower() not in stopwords.
```

In [97]:
```
clean_mess
```

Out[97]:
```
['Sample', 'message', 'Notice', 'punctuation']
```

In [99]:
```
def text_process(mess):
    """
    1.remove punc
    2.remove stop words
    3. return list of clean text words
    """
    nopunch = [char for char in mess if char not in string.punctuation]
    nopunch = ''.join(nopunch)
    return [word for word in nopunch.split() if word.lower not in stopwords.word
```

In [101…
```
messages.head()
```

Out[101…

	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 [103…
```
messages['message'].head(5).apply(text_process)
```

Out[103…
```
0    [Go, until, jurong, point, crazy, Available, o...
1                        [Ok, lar, Joking, wif, u, oni]
2    [Free, entry, in, 2, a, wkly, comp, to, win, F...
3    [U, dun, say, so, early, hor, U, c, already, t...
4    [Nah, I, dont, think, he, goes, to, usf, he, l...
Name: message, dtype: object
```

In [113…
```
"""We will do 3 steps using the bag-of-words model:"
  1. Count how many times does a word occur in each message(known as term freque
  2. Weigh the counts, so that frequent tokens get lower weight(inverse document
   3. Normalize the vectors to unit lenght, to abstract from the originial text
```

Out[113…
```
'We will do 3 steps using the bag-of-words model:"\n  1. Count how many times d
oes a word occur in each message(known as term frequency)\n  2. Weigh the count
s, so that frequent tokens get lower weight(inverse document frequency)\n   3. 
Normalize the vectors to unit lenght, to abstract from the originial text lengt
h (L2 norm)'
```

In [119…
```
"CountVectorization and Spark Matrix"
```

Out[119…
```
'CountVectorization and Spark Matrix'
```

In [121…
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [123…
```
bow_transformer = CountVectorizer(analyzer=text_process).fit(messages['message']
```

In [127…
```
print(len(bow_transformer.vocabulary_))
```
```
11747
```

In [139…
```
mess4 = messages['message'][3]
```

In [141…
```
print(mess4)
```
```
U dun say so early hor... U c already then say...
```

In [143…
```
bow1 = bow_transformer.transform([mess4])
```

In [145…
```
print(bow1)
```
```
  (0, 4221)	2
  (0, 4828)	1
  (0, 5476)	1
  (0, 6427)	1
  (0, 6447)	1
  (0, 7427)	1
  (0, 9832)	2
  (0, 10174)	1
  (0, 10703)	1
```

In [137…
```
print(bow1.shape)
```
```
(1, 11747)
```

In [149…
```
bow_transformer.get_feature_names_out()[9832]
```
Out[149…
```
'say'
```

In [153…
```
messages_bow = bow_transformer.transform(messages['message'])
```

In [154…
```
print('Shape of Sparse Matrix:', messages_bow.shape)
```
```
Shape of Sparse Matrix: (5572, 11747)
```

In [155…
```
messages_bow.nnz
```
Out[155…
```
79463
```

In [159…
```
"check what is sparsity ?"
```
Out[159…
```
'check what is sparsity ?'
```

In [161…
```
from sklearn.feature_extraction.text import TfidfTransformer
```

In [163…
```
tfidTransformer = TfidfTransformer().fit(messages_bow)
```

In [185…
```
tfidf4 = tfidTransformer.transform(bow1)
```

In [167…
```
print(tfidTransformer.transform(bow1))
```

```
  (0, 10703)	0.2214828525636521
  (0, 10174)	0.19345051326676527
  (0, 9832)	0.5147493130794172
  (0, 7427)	0.41952836023632145
  (0, 6447)	0.3046289560740644
  (0, 6427)	0.28629349827015765
  (0, 5476)	0.2841540501592932
  (0, 4828)	0.25442769469153637
  (0, 4221)	0.3902711884065556
```

In [177…
```
print(tfidTransformer.idf_[bow_transformer.vocabulary_['university']])
```

```
8.527076498901426
```

In [179…
```
messages_tfidf = tfidTransformer.transform(messages_bow)
```

In [181…
```
from sklearn.naive_bayes import MultinomialNB
```

In [183…
```
spam_detect_model = MultinomialNB().fit(messages_tfidf, messages['label'])
```

In [187…
```
spam_detect_model.predict(tfidf4)[0]
```

Out[187…
```
'ham'
```

In [189…
```
print(messages['label'][3])
```

```
ham
```

In [191…
```
all_pred = spam_detect_model.predict(messages_tfidf)
```

In [193…
```
all_pred
```

Out[193…
```
array(['ham', 'ham', 'spam', ..., 'ham', 'ham', 'ham'], dtype='<U4')
```

In [195…
```
from sklearn.model_selection import train_test_split
```

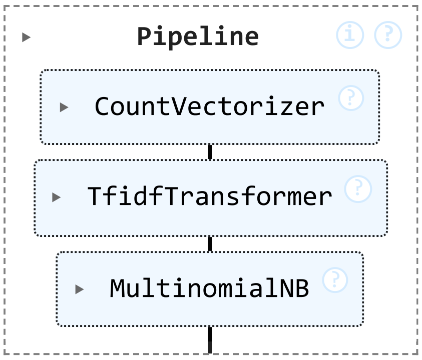
In [197…
```
msg_train, msg_test, label_train, label_test = train_test_split(messages['messag
```

In [201…
```
from sklearn.pipeline import Pipeline
```

In [205…
```
pipeline = Pipeline([
    ('bow', CountVectorizer(analyzer=text_process)),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])
```

In [207…
```
pipeline.fit(msg_train, label_train)
```

Out[207…



In [209…
```
predictions = pipeline.predict(msg_test)
```

In [211…
```
from sklearn.metrics import classification_report
```

In [213…
```
print(classification_report(label_test, predictions))
```

```
              precision    recall  f1-score   support

         ham       0.94      1.00      0.97      1586
        spam       1.00      0.58      0.73       253

    accuracy                           0.94      1839
   macro avg       0.97      0.79      0.85      1839
weighted avg       0.95      0.94      0.94      1839
```

In [ ]: