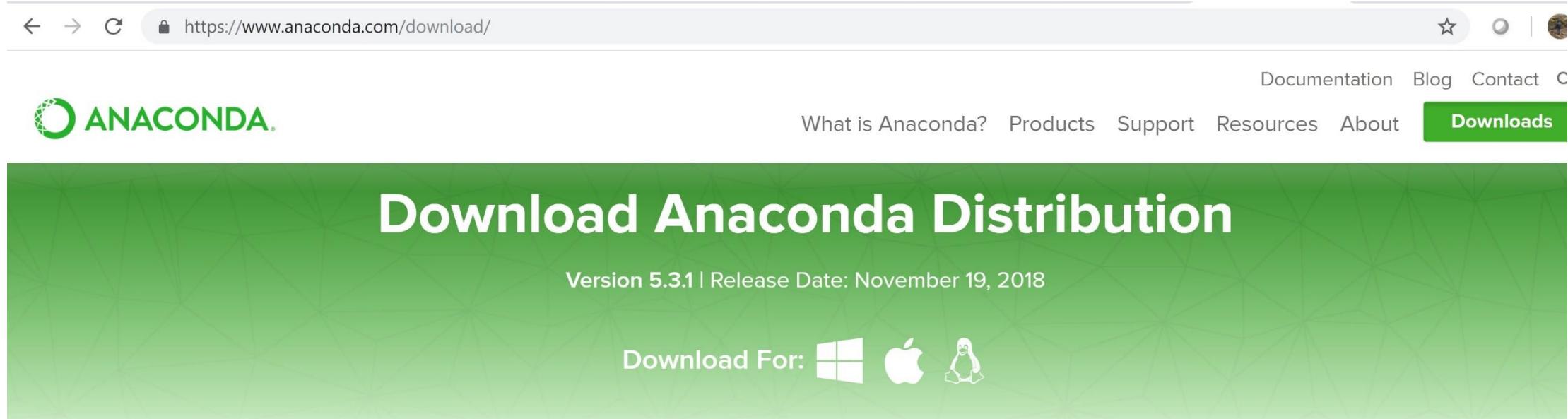


Python and Text Mining

Gates

Get Python3 Via Anaconda

www.anaconda.com/download



A screenshot of a web browser displaying the Anaconda download page. The URL in the address bar is <https://www.anaconda.com/download/>. The page features a green header with the Anaconda logo and navigation links for Documentation, Blog, Contact, What is Anaconda?, Products, Support, Resources, About, and Downloads. The main content area has a green background with a geometric pattern and displays the text "Download Anaconda Distribution" in large white letters, "Version 5.3.1 | Release Date: November 19, 2018" in smaller white letters, and "Download For:" followed by icons for Windows, Apple, and Linux.

← → C https://www.anaconda.com/download/ ☆ ○ | 🌐

ANACONDA. Documentation Blog Contact C

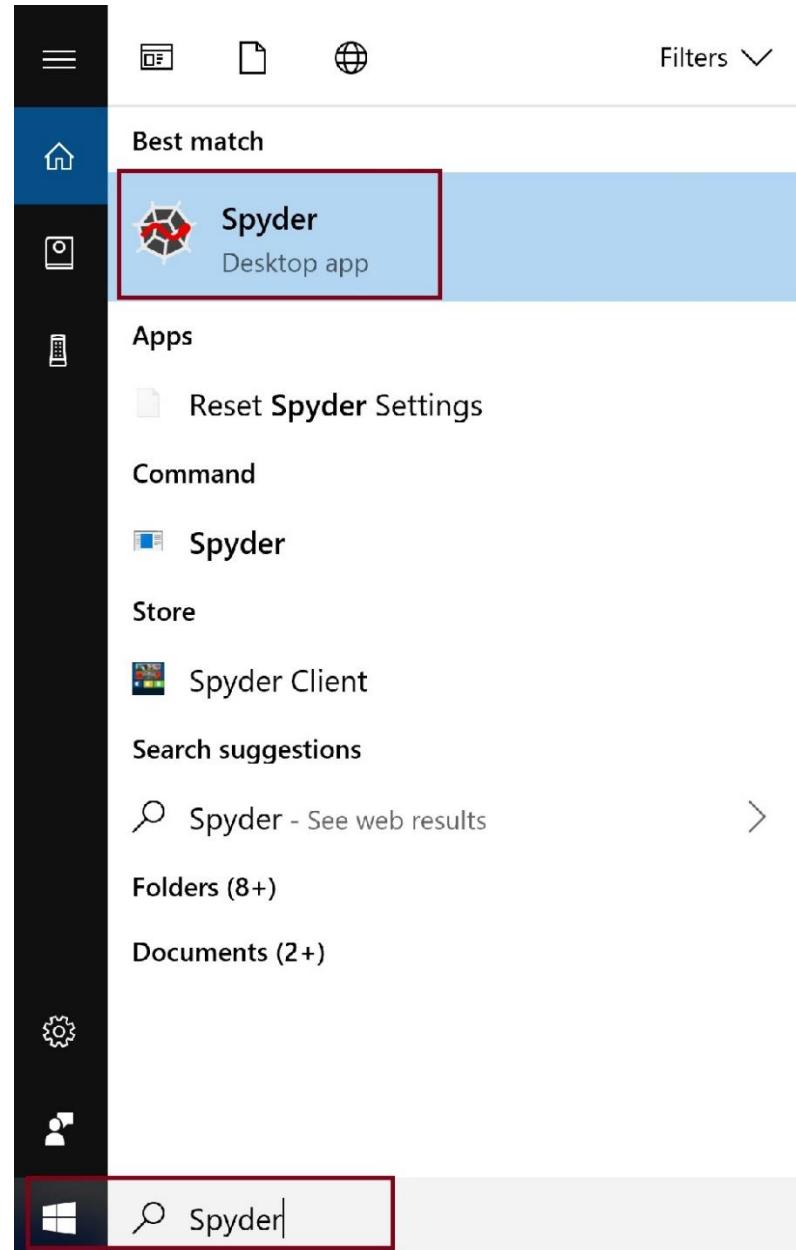
What is Anaconda? Products Support Resources About Downloads

Download Anaconda Distribution

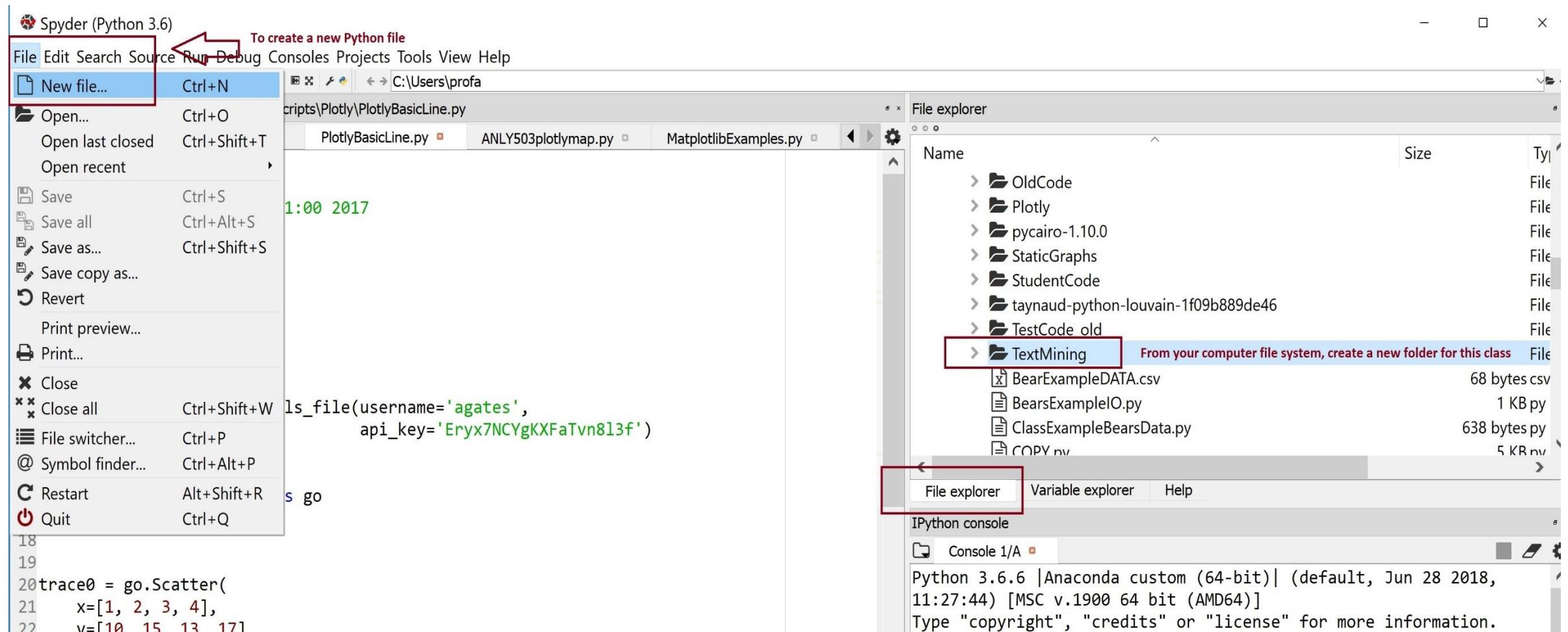
Version 5.3.1 | Release Date: November 19, 2018

Download For:   

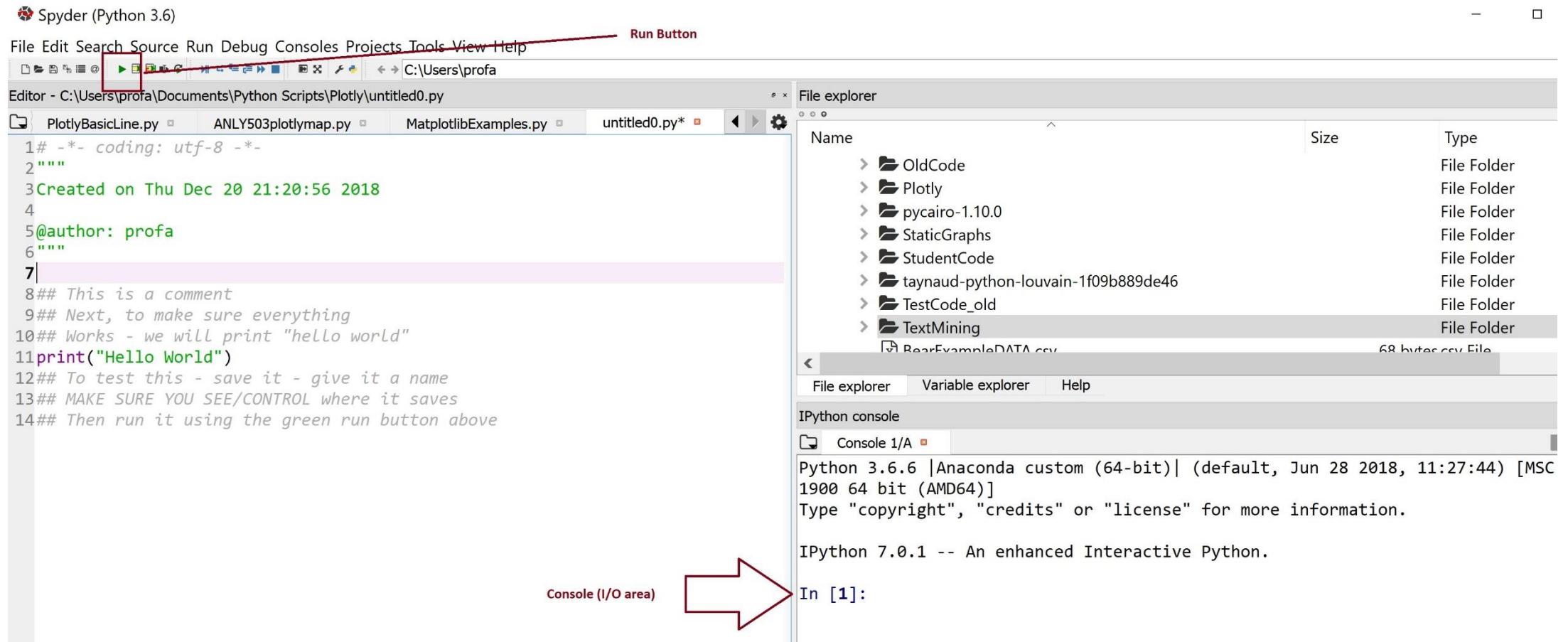
Open Python in Spyder



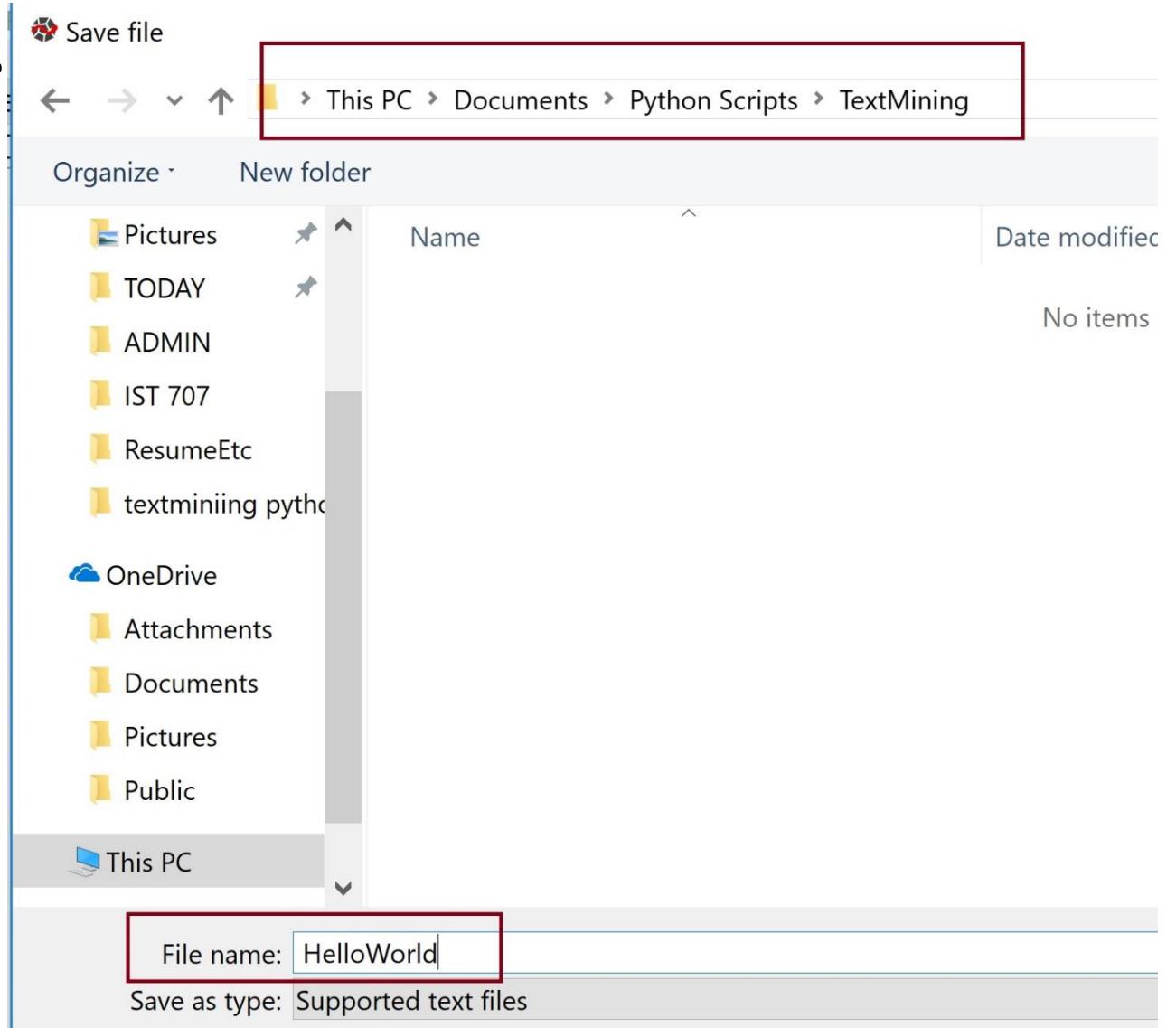
Inside Spyder Part 1



Save as – Name it - and Run It



Save and Name File - ** Make sure you save it to the right location.



Success!

The screenshot shows a Python development environment with the following components:

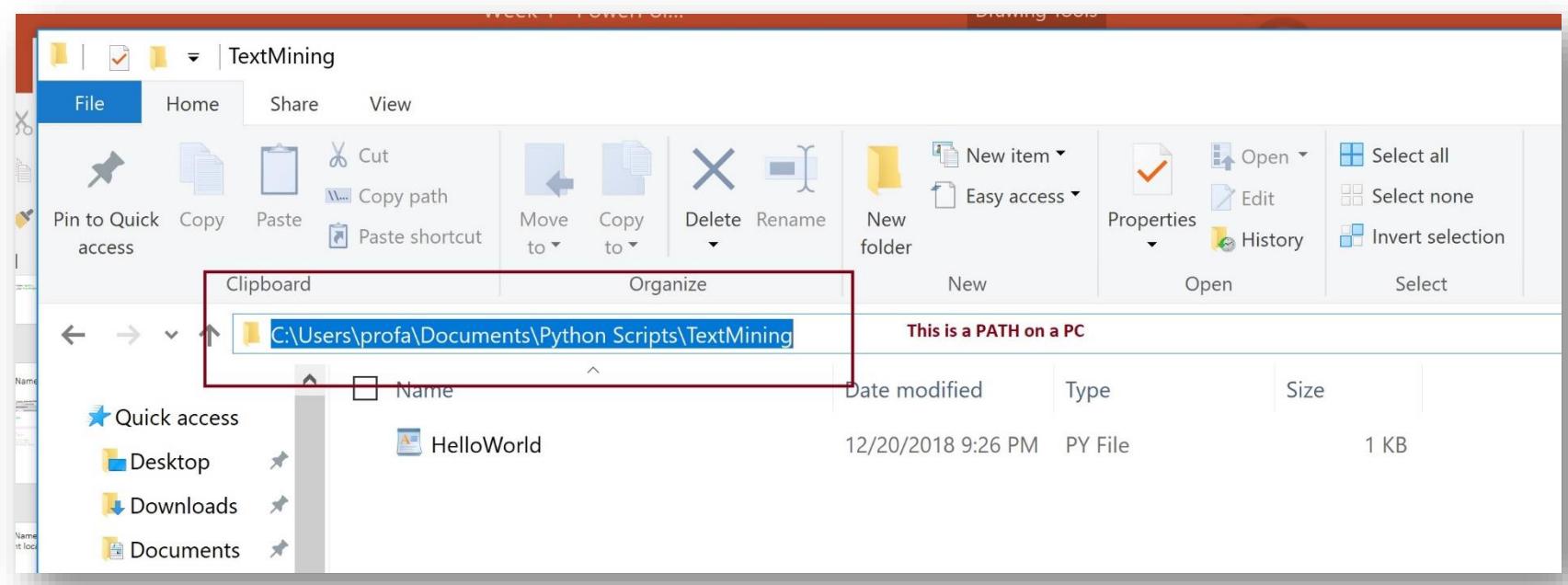
- File Explorer:** Shows a list of files in the directory `C:\Users\profa\Documents\Python Scripts\TextMining`. The file `HelloWorld.py` is listed with a size of 346 bytes.
- Code Editor:** Displays the contents of `HelloWorld.py`. The code is as follows:

```
1# -*- coding: utf-8 -*-
2"""
3Created on Thu Dec 20 21:20:56 2018
4
5@author: profa
6"""
7
8## This is a comment
9## Next, to make sure everything
10## Works - we will print "hello world"
11print("Hello World")
12## To test this - save it - give it a name
13## MAKE SURE YOU SEE/CONTROL where it saves
14## Then run it using the green run button above
```

- IPython Console:** Shows the output of running the script.
 - Python 3.6.6 [Anaconda custom (64-bit)] (default, Jun 28 2018, 11:27:44) [MSC 1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
 - IPython 7.0.1 -- An enhanced Interactive Python.
 - In [1]: `runfile('C:/Users/profa/Documents/Python Scripts/TextMining/HelloWorld.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining')`
 - >Hello World
 - It worked!

Paths and Files

- 1) What is a path?
- 2) Where are your Python files?
- 3) How can you read in data from a file that is “local”?
- 4) How can you read in data from a file that is on a path?



Paths, Files, Folders, and Data

The image shows two windows illustrating file paths, folders, and data.

File Explorer Window:

- Toolbar:** File, Home, Share, View, Pin to Quick access, Copy, Paste, Cut, Copy path, Move to, Copy to, Delete, Rename, New item, New folder, Open, Select all, Select none, Invert selection.
- Address Bar:** This PC > OS (C:) > Users > profa > Documents > Python Scripts > TextMining >
- Left Sidebar:** Quick access, Desktop, Downloads.
- Content Area:** A list of items:
 - DATA** (File folder, Date modified: 12/20/2018 9:36 PM, Type: File folder, Size: 1 KB)
 - HelloWorld (PY File, Date modified: 12/20/2018 9:26 PM, Type: PY File, Size: 1 KB)

Spyder Python IDE Window:

- Title Bar:** Spyder (Python 3.6)
- Menu Bar:** File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help
- Toolbar:** Standard icons for file operations.
- Editor:** Editor - C:\Users\profa\Documents\Python Scripts\TextMining\HelloWorld.py
 - Code content:

```
1# -*- coding: utf-8 -*-
2"""
3Created on Thu Dec 20 21:20:56 2018
4
5@author: profa
6"""
7
8## This is a comment
9## Next to make sure everything
```
- File Explorer:** Name, Size, Type
 - DATA** (File Folder, Size: 346 bytes, Type: File)
 - HelloWorld.py (PY File, Size: 346 bytes, Type: File)

Create a Mini Data file

- 1) Use Excel to build a min datafile.
- 2) Save it as .csv
- 3) Put it in DATA
- 4) Open and print it using Spyder

	A	B	C	D	E
1	FEMALEID	AGE	HEIGHT	WEIGHT	WAIST
2	295	23	64.3	114.8	67.2
3	2739	32	66.4	149.3	82.5
4	2992	25	62.3	107.8	66.7
5	3745	55	62.3	160.1	93
6	4486	27	59.6	127.1	82.6
7	4488	29	63.6	123.1	75.4
8	4878	25	59.8	111.7	73.6
9	4880	22	63.3	156.3	81.4
10	4881	41	67.9	218.8	99.4
11	4835	32	61.4	110.2	67.7
12	4842	31	66.7	188.3	100.7

Save As

Documents > Python Scripts > TextMining > DATA

Search DATA

Organize New folder

ADMIN

IST 707

ResumeEtc

textmining python



location or PATH

No items match your search.

Microsoft Excel

OneDrive

Attachments

Documents

Pictures

Public

This PC

Network

File name: FemaleHealthData

Save as type: CSV (Comma delimited)

Spyder (Python 3.6)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\profa\Documents\Python Scripts\TextMining\DATA

Editor - C:\Users\profa\Documents\Python Scripts\TextMining\HelloWorld.py

:Line.py ANLY503plotlymap.py MatplotlibExamples.py HelloWorld.py

1# -*- coding: utf-8 -*-
2"""
3Created on Thu Dec 20 21:20:56 2018
4
5@author: profa
6"""
7
8## This is a comment
9## Next, to make sure everything
10## Works - we will print "hello world"
11print("Hello World")
12## To test this - save it - give it a name

File explorer Variable explorer Help

PATH TO THE DATA FILE

```

2 """
3 Created on Thu Dec 20 21:52:25 2018
4
5 @author: profa
6 """
7
8 ## Reading in basic data using pandas
9 ## READ ABOUT pandas!
10 import pandas as pd
11 ## the "as" allows you to set a "nickname"
12 ## My nickname is pd
13 ## I can now call any pandas methods using pd.whatever....
14 MyDataFrame = pd.read_csv("DATA/FemaleHealthData.csv")
15 print(MyDataFrame.head())
16

```

2	DATA	File Folder	12/20/2018 9:49 PM
3	FemaleHealthData.csv	532 bytes csv File	12/20/2018 9:49 PM
4	HelloWorld.py	348 bytes py File	12/20/2018 9:54 PM
5	ReadData1.py	385 bytes py File	12/20/2018 9:54 PM

File explorer Variable explorer Help

IPython console

Console 1/A

[MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.0.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/HelloWorld.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining')
Hello World

In [2]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/ReadData1.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining')

	FEMALEID	AGE	HEIGHT	WEIGHT	WAIST
0	295	23	64.3	114.8	67.2
1	2739	32	66.4	149.3	82.5
2	2992	25	62.3	107.8	66.7
3	3745	55	62.3	160.1	93.0
4	4486	27	59.6	127.1	82.6

Regular Expressions and Text

Gates

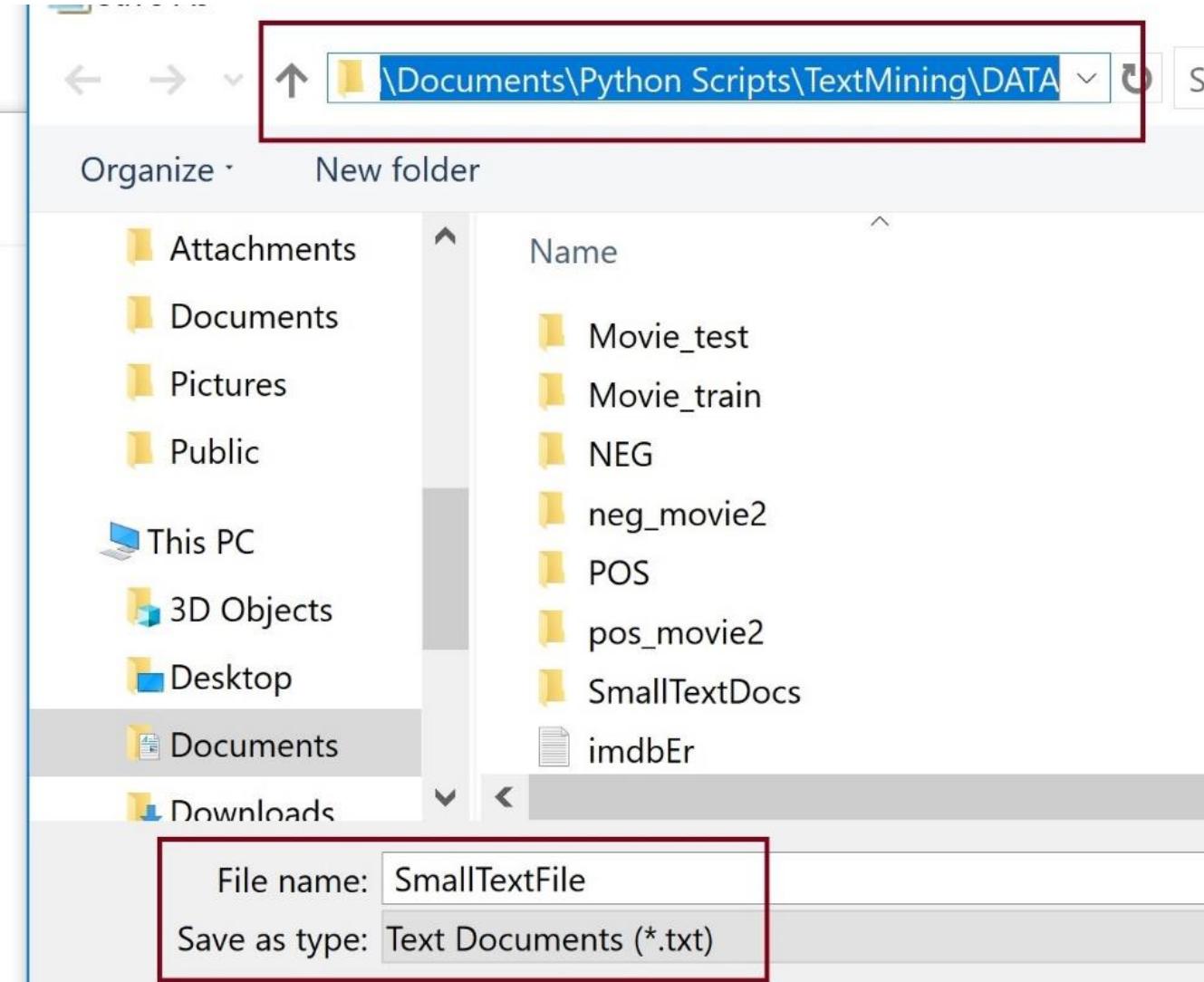
Always Create A Small Test Example

~~to code it You can apply these concepts~~



File Edit Format View Help

When you test any new method
always - really always - create small
examples for yourself. This helps you
to better understand mining and analytics.
Text Mining is fun!



Using Python's re for Regular Expressions

```
425#####
426##  
427### Regular Expressions Small Examples  
428###  
429#####  
430print("REGULAR EXPRESSIONS....")  
431import re  
432## Read in the data  
433MyText = open('DATA/SmallTextFile.txt')  
434MyList=[]  
435for line in MyText:  
436    print("The next line is, ", line)  
437    ##Strip (remove) whitespace from end  
438    line = line.rstrip()  
439    line = re.sub('[!@#$-.]', '', line)  
440    ## Remove all extra whitespace  
441    line=re.sub( '\s+', ' ', line).strip()  
442    ## Remove newline  
443    line = line.strip("\n")  
444    line = line.lower()  
445    print("Now the line is: ", line)  
446    MyList.append(line)  
447    if re.search('mining', line):  
448        print("yes")  
449  
450print("MyList is: ", MyList)  
451## Join  
452Final="".join(MyList)  
453print("The final string is: ", Final)
```

The screenshot shows a Jupyter Notebook interface with two panes. The left pane displays the Python code above. The right pane shows the results of the code execution in an IPython console.

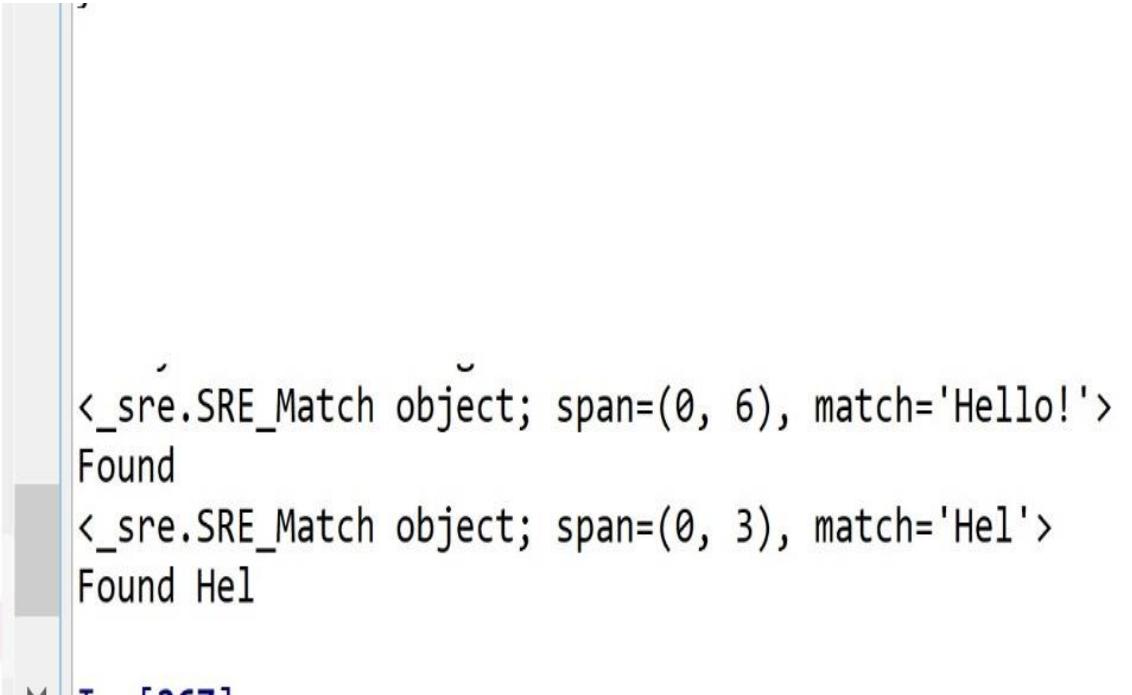
File explorer: SmallTextFile.txt (174 bytes txt File, 1/1/2019 1:20 PM)

IPython console:

```
REGULAR EXPRESSIONS....  
The next line is, When you test any new method  
Now the line is: when you test any new method  
The next line is, always - really always - create small  
Now the line is: always really always create small  
The next line is, examples for yourself. This helps you  
Now the line is: examples for yourself this helps you  
The next line is, to better understand mining and analytics.  
Now the line is: to better understand mining and analytics  
yes  
The next line is, Text Mining is fun!  
Now the line is: text mining is fun  
yes  
MyList is: ['when you test any new method', 'always really always create small', 'examples for yourself this helps you', 'to better understand mining and analytics', 'text mining is fun']  
The final string is: when you test any new method always really always create small examples for yourself this helps you to better understand mining and analyticstext mining is fun
```

Searching with re

```
456## Searching at the beginning of a string
457MyText2 = "Hello! This is an example :)"
458Result=re.search('Hello!', MyText2)
459print(Result)
460if(Result):
461    print("Found")
462
463Result2=re.search('Hel', MyText2)
464print(Result2)
465if(Result2):
466    print("Found Hel")
467
```



The screenshot shows a terminal window with the following content:

```
<_sre.SRE_Match object; span=(0, 6), match='Hello!'>
Found
<_sre.SRE_Match object; span=(0, 3), match='Hel'>
Found Hel
```

Notes:

1) From code on the previous slide, you should also have a `.close()` at some point:

Example:

`MyText.close()`

2) If you forget to close your files, odd things will start to happen ;)

3) No one really “memorizes re”. In general, when you want to do something, you think about what you want to do and then you Google.

4) This link is a HUGE tutorial and is a big help – bookmark it....

RE: <https://www.py4e.com/html3/11-regex>

5) I like this tutorial better:

<https://docs.python.org/3/library/re.html>

NLTK in Python

- 1) Get NLTK
- 2) Practice with a pretend hand-made text file

Installers

conda install ?

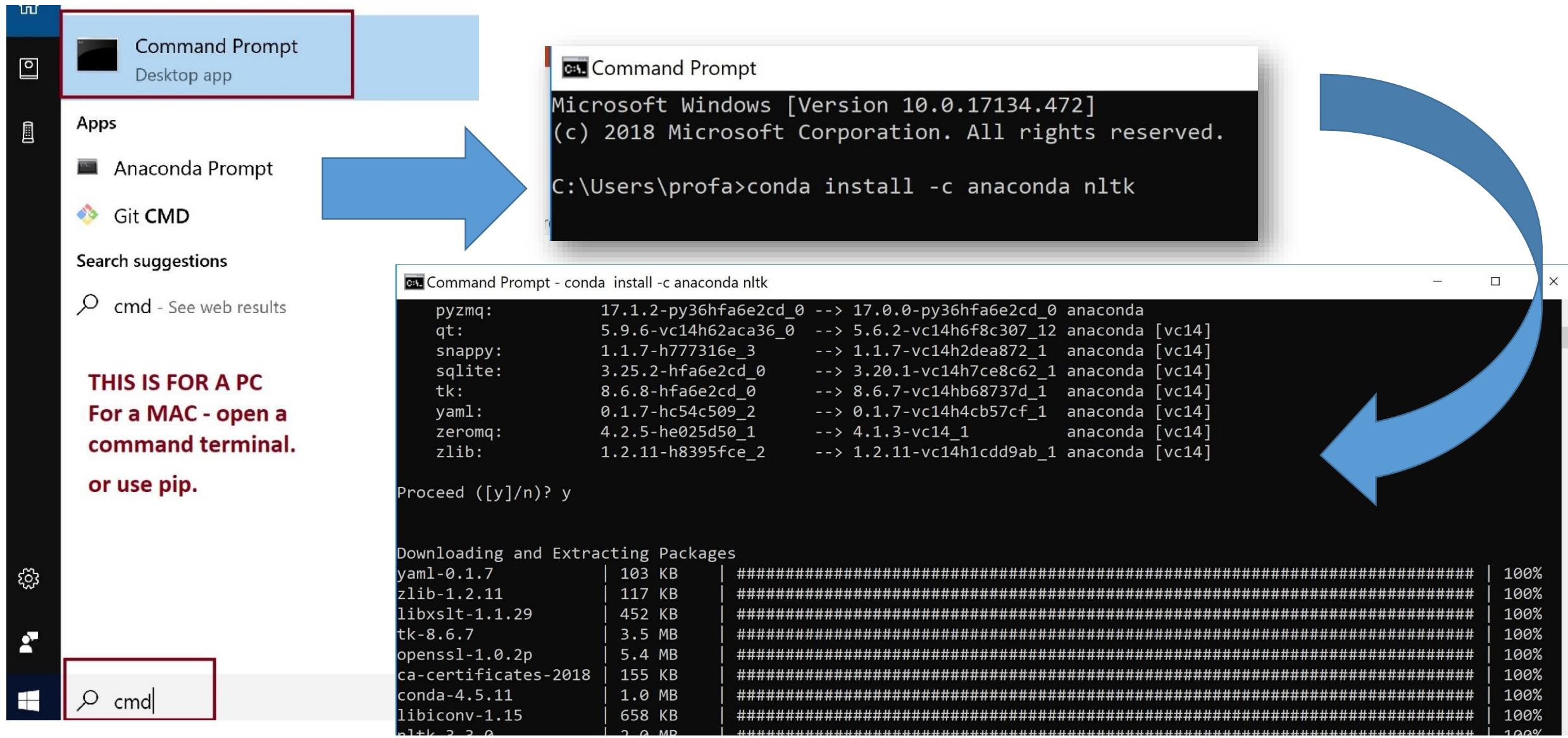
linux-ppc64le	v3.3.0
osx-32	v3.0.4
linux-64	v3.3.0
win-32	v3.3.0
osx-64	v3.3.0
linux-32	v3.3.0
win-64	v3.3.0

<https://anaconda.org/anaconda/nltk>

To install this package with conda run:

```
conda install -c anaconda nltk
```

Command Terminals and getting NLTK



```
8## nltk examples
9import nltk
10from nltk.tokenize import word_tokenize
11
12text="To be or not to be"
13
14tokens = [t for t in text.split()]
15print(tokens)
16
17freq = nltk.FreqDist(tokens)
18
19for key,val in freq.items():
20    print (str(key) + ':' + str(val))
21
22freq.plot(20, cumulative=False)
23
24mytext = "Hiking is fun! Hiking with dogs is more fun :)"
25print(word_tokenize(mytext))
26
27
```

File explorer Variable explorer Help

IPython console

Console 1/A

```
In [9]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/nltkExample1.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining')
['To', 'be', 'or', 'not', 'to', 'be']
<generator object bigrams at 0x0000026B52CCF7D8>

In [10]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/nltkExample1.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining')
['To', 'be', 'or', 'not', 'to', 'be']
To:1
be:2
or:1
not:1
to:1
['Hiking', 'is', 'fun', '!', 'Hiking', 'with', 'dogs', 'is', 'more',
'fun', ':', ')']
```

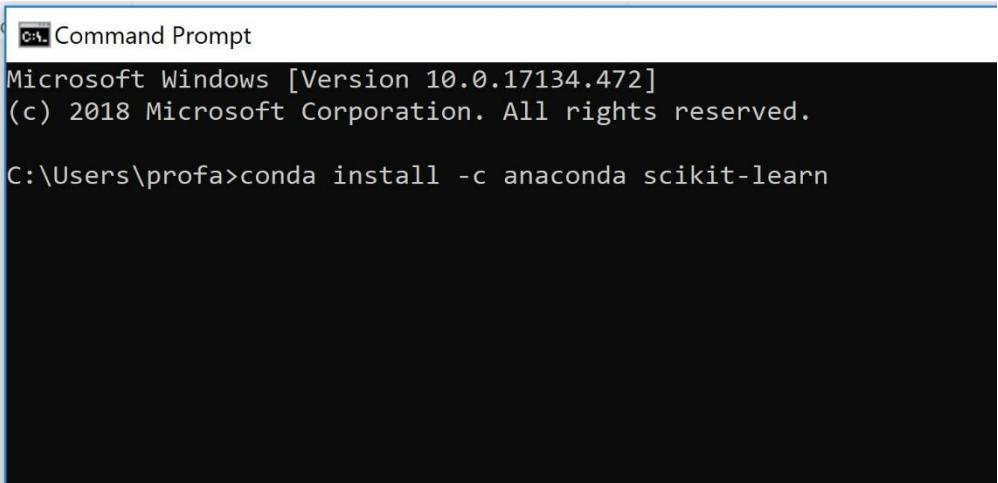
Getting sklearn

Ref:

<https://anaconda.org/anaconda/scikit-learn>

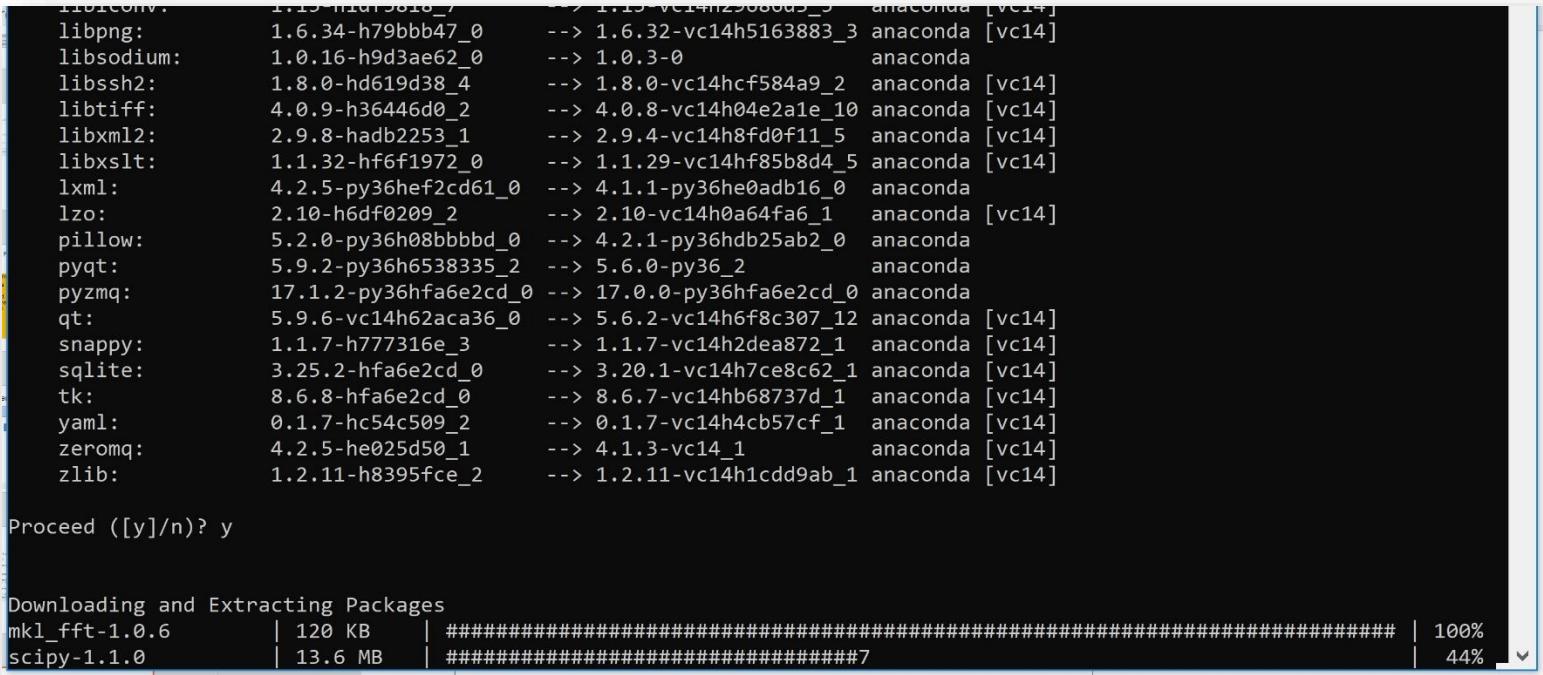
Command:

conda install -c
anaconda scikit-learn



```
Command Prompt
Microsoft Windows [Version 10.0.17134.472]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\profa>conda install -c anaconda scikit-learn
```

```
libiconv:          1.15-h1d15818_7    --> 1.15-vc14h29680d5_5  anaconda [vc14]
libpng:           1.6.34-h79bbb47_0    --> 1.6.32-vc14h5163883_3  anaconda [vc14]
libsodium:        1.0.16-h9d3ae62_0    --> 1.0.3-0                anaconda
libssh2:          1.8.0-hd619d38_4    --> 1.8.0-vc14hcf584a9_2  anaconda [vc14]
libtiff:          4.0.9-h36446d0_2    --> 4.0.8-vc14h04e2a1e_10 anaconda [vc14]
libxml2:          2.9.8-hadb2253_1    --> 2.9.4-vc14h8fd0f11_5  anaconda [vc14]
libxslt:          1.1.32-hf6f1972_0    --> 1.1.29-vc14hf85b8d4_5 anaconda [vc14]
lxml:             4.2.5-py36hef2cd61_0    --> 4.1.1-py36he0adb16_0  anaconda
lzo:               2.10-h6df0209_2     --> 2.10-vc14h0a64fa6_1   anaconda [vc14]
pillow:           5.2.0-py36h08bbbbbd_0  --> 4.2.1-py36hdb25ab2_0  anaconda
pyqt:              5.9.2-py36h6538335_2   --> 5.6.0-py36_2         anaconda
pyzmq:             17.1.2-py36hfa6e2cd_0  --> 17.0.0-py36hfa6e2cd_0 anaconda
qt:                5.9.6-vc14h62aca36_0  --> 5.6.2-vc14h6f8c307_12 anaconda [vc14]
snappy:            1.1.7-h777316e_3    --> 1.1.7-vc14h2dea872_1  anaconda [vc14]
sqlite:            3.25.2-hfa6e2cd_0    --> 3.20.1-vc14h7ce8c62_1 anaconda [vc14]
tk:                8.6.8-hfa6e2cd_0    --> 8.6.7-vc14hb68737d_1  anaconda [vc14]
yaml:              0.1.7-hc54c509_2    --> 0.1.7-vc14h4cb57cf_1  anaconda [vc14]
zeromq:            4.2.5-he025d50_1    --> 4.1.3-vc14_1         anaconda [vc14]
zlib:              1.2.11-h8395fce_2   --> 1.2.11-vc14h1cdd9ab_1 anaconda [vc14]

Proceed ([y]/n)? y

Downloading and Extracting Packages
mkl_fft-1.0.6      | 120 KB    | #####| 100% | 44% |
```

Test your sklearn

#Type this into a new Python file, save, and run:

```
from sklearn.feature_extraction.text import CountVectorizer  
print("hello")
```

If your sklearn is installed, this will print hello and will not show any errors.

Tokenize into words

```
7 import nltk
8 import pandas
9 from sklearn.feature_extraction.text import CountVectorizer
10 from nltk.tokenize import word_tokenize
11
12## Tokenization
13## Breaking text into tokens - in this case - words
14text="This is any sentence of text. It can have punctuation, CAPS!, etc."
15tokenized_word=word_tokenize(text)
16print(tokenized_word)
```

The screenshot shows a Jupyter Notebook interface with two main panes. The left pane displays Python code for tokenizing text. The right pane shows a file explorer with a folder named 'unsup' containing three files: 'labeledBow.feat', 'unsupBow.feat', and 'urls_pos.txt'. Below the file explorer is an IPython console window titled 'Console 1/A'.

```
In [16]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/Example NLTK sklearn senti and processing.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining')
['This', 'is', 'any', 'sentence', 'of', 'text', '.', 'It', 'can', 'have',
'punctuation', ',', 'CAPS', '!', ',', 'etc', '.']
```

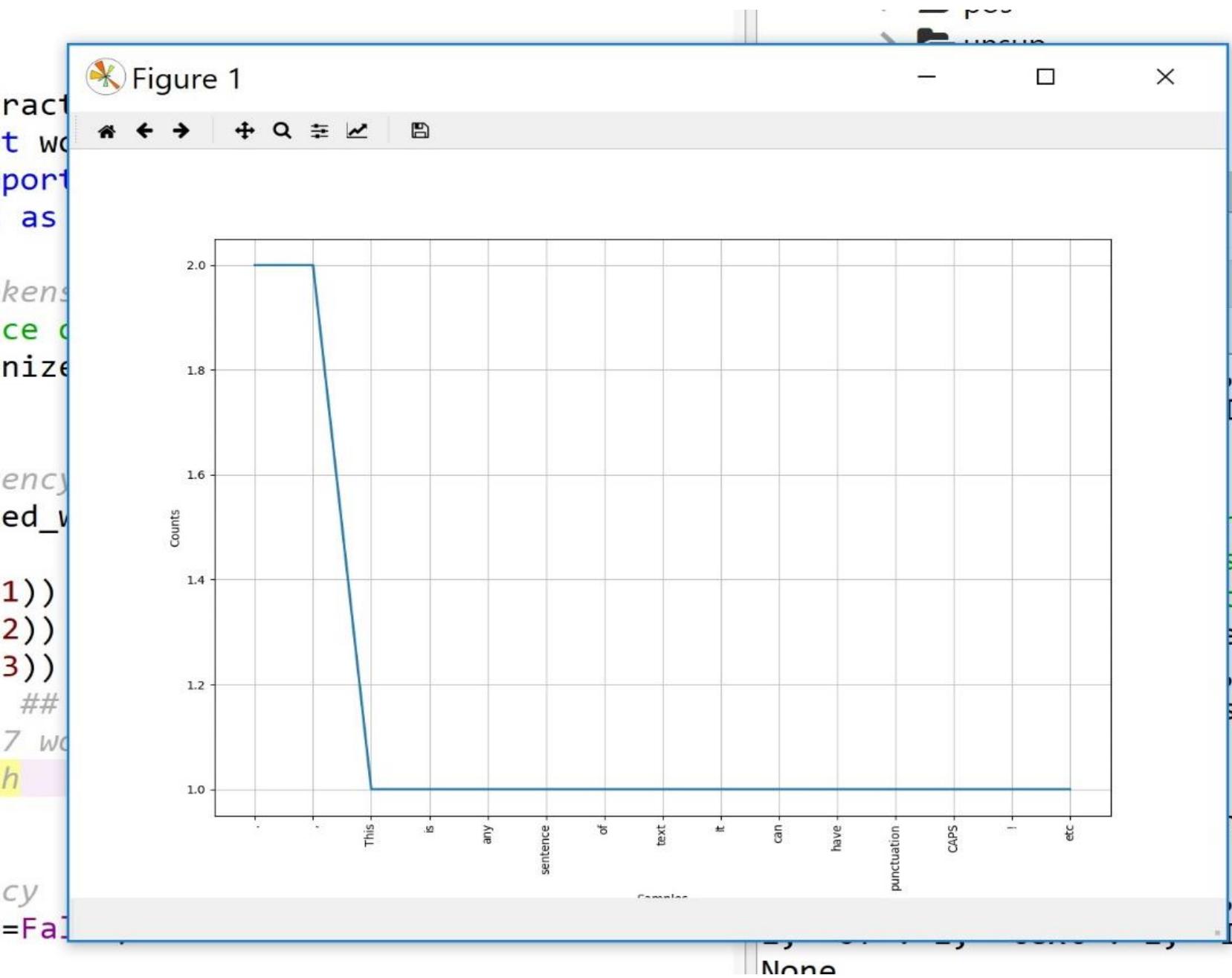
Tokens, Frequency, and Plot

```
7 import nltk
8 import pandas
9 from sklearn.feature_extraction.text import CountVectorizer
10 from nltk.tokenize import word_tokenize
11 from nltk.probability import FreqDist
12 import matplotlib.pyplot as plt
13 ## Tokenization
14 ## Breaking text into tokens - in this case - words
15 text="This is any sentence of text. It can have punctuation, C
16 tokenized_word=word_tokenize(text)
17 print(tokenized_word)
18
19 ## Looking at word frequency
20 fdist = FreqDist(tokenized_word)
21 print(fdist)
22 print(fdist.most_common(1)) ## most common left to right
23 print(fdist.most_common(2)) ## two most common
24 print(fdist.most_common(3)) ## three most common
25 print(fdist.freq("is")) ## how frequent
26 ## "is" occurs once in 17 words. 1/17 = .058
27 fdist.N() # freq of each
28
29 # Visualize word frequency
30 fdist.plot(30,cumulative=False)
31 plt.show()
```

The screenshot shows a Jupyter Notebook interface. The file explorer sidebar on the right lists a directory structure with a 'pos' folder, an 'unsup' folder containing 'labeledBow.feat', 'unsupBow.feat', and 'urls_pos.txt', and a 'File Folder' entry. The IPython console below the sidebar shows the execution of code cells. Cell 24 runs a script to analyze text tokens and frequencies.

```
In [24]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/Example NLTK sklearn senti and processing.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining')
['This', 'is', 'any', 'sentence', 'of', 'text', '.', 'It', 'can',
 'have', 'punctuation', ',', 'CAPS', '!', ',', 'etc', '.']
<FreqDist with 15 samples and 17 outcomes>
[('.', 2)]
[('.', 2), (',', 2)]
[('.', 2), (',', 2), ('This', 1)]
0.058823529411764705
```

plot



Stopwords

from nltk.corpus import stopwords

```
13from nltk.corpus import stopwords
14## Tokenization
15## Breaking text into tokens - in this case - words
16text="This is any sentence of text. It can have punctuation,
17tokenized_word=word_tokenize(text)
18print(tokenized_word)
19## Looking at word frequency
20fdist = FreqDist(tokenized_word)
21print(fdistribution)
22print(fdistribution.most_common(1)) ## most common left to right
23print(fdistribution.most_common(2)) ## two most common
24print(fdistribution.most_common(3)) ## three most common
25print(fdistribution.freq("is")) ## how frequent
26## "is" occurs once in 17 words. 1/17 = .058
27fdistribution.N() # freq of each
28# Visualize word frequency
29fdistribution.plot(30,cumulative=False)
30plt.show()
31## Stopwords - English
32stop_words=set(stopwords.words("english"))
33print(stop_words)
34## Removing Stopwords
35filtered_text=[] ## Create a new empty list
36for w in tokenized_word:
37    print(w)
38    if w not in stop_words:
39        filtered_text.append(w)
40print("Tokenized text:",tokenized_word)
41print("Filtered text:",filtered_text)
```



```
This
is
any
sentence
of
text
.
It
can
have
punctuation
,
CAPS
!
,
etc
.

Tokenized text: ['This', 'is', 'any', 'sentence', 'of', 'text',
'.', 'It', 'can', 'have', 'punctuation', ',', 'CAPS', '!', ',',
'etc', '.']

Filtered text: ['This', 'sentence', 'text', '.', 'It',
'punctuation', ',', 'CAPS', '!', ',', 'etc', '.']
```

Stemming

```
43
44# Stemming
45from nltk.stem import PorterStemmer
46from nltk.tokenize import sent_tokenize, word_tokenize
47
48ps = PorterStemmer()    ## method from nltk
49
50stemmed_words=[]    ## make new empty list
51for w in filtered_text:
52    stemmed_words.append(ps.stem(w))
53
54print("Filtered:",filtered_text)
55print("Stemmed:",stemmed_words)
```

```
Filtered: ['This', 'sentence', 'text', '.', 'It', 'punctuation',
',', 'CAPS', '!', ',', 'etc', '.']
Stemmed: ['thi', 'sentenc', 'text', '.', 'It', 'punctuat', ',',
'cap', '!', ',', 'etc', '.']
```

Lemmatization vs. Stemming

```
65#-----
66#Lemmatization reduces words to their base word
67#-----
68# Lemmatization is usually more sophisticated than
69#stemming. Stemmer works on an individual word without
70# knowledge of the context. For example, The word "better"
71# has "good" as its lemma. This thing will miss by
72# stemming because it requires a dictionary look-up.
73from nltk.stem.wordnet import WordNetLemmatizer
74lem = WordNetLemmatizer() ## method we are using
75from nltk.stem.porter import PorterStemmer
76stem = PorterStemmer()
77word = "flying"
78print("Lemmatized Word:",lem.lemmatize(word,"v"))
79print("Stemmed Word:",stem.stem(word))
```

```
Lemmatized Word: fly
Stemmed Word: fli
```

Part-of-Speech(POS) tagging

```
80
81## -----
82## Part-of-Speech(POS) tagging
83## -----
84# identify the grammatical group
85# Whether it is a NOUN, PRONOUN,
86# ADJECTIVE, VERB, ADVERBS
87# -----
88sent = "Three wonderful things in life are hiking, moutains,
89Mytokens=nltk.word_tokenize(sent)
90print(Mytokens)
91MyTAGS = nltk.pos_tag(Mytokens)
92print(MyTAGS)
93
94
95
96
```

```
['Three', 'wonderful', 'things', 'in', 'life', 'are', 'hiking',
',', 'moutains', ',', 'and', 'COFFEE', '!']
[('Three', 'CD'), ('wonderful', 'JJ'), ('things', 'NNS'), ('in',
'IN'), ('life', 'NN'), ('are', 'VBP'), ('hiking', 'VBG'), (',',
','), ('moutains', 'NNS'), (',', ','), ('and', 'CC'), ('COFFEE',
'NNP'), ('!', '.')]
```

Formatting TEXT for Analysis

- 1) Text is messy!
- 2) What can it look like?
 - 1) JSON
 - 2) HTML
 - 3) Documents
 - 4) FB
 - 5) Twitter
 - 6) Other...
- 3) What are the variables or features of a document (or website or Facebook post, or Tweet, or novel, or Amazon ranking comment...)?

What is vectorization?

- 1) Grab the text (and all the garbage that comes with it).
- 2) Clean it so that it is just text – all lowercase.
- 3) If it is labeled – keep the label.
- 4) Read it into a Corpus object
- 5) Build a DF or Matrix where the words are the variables.
- 6) It is also possible that n-grams (such as bigrams) can be the variables. More on this later....
- 7) All of these steps require *a lot* of cleaning and prep.

Examples

```
## Note: the following examples will use at least these libraries
import nltk
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
## For Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
import os
```

Example 1: Positive and Negative Folders of Text files

The screenshot shows the Spyder Python 3.6 IDE interface. The top menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, and Help. The toolbar below has various icons for file operations. The main area shows an Editor window with the path C:\Users\profa\Documents\Python Scripts\TextMining\DATA\POS\cv000_29590.txt open. The code in the editor discusses a movie plot, mentioning Jack the Ripper and various actors like Alan Moore, Eddie Campbell, Robbie Coltrane, and Johnny Depp. To the right is a File explorer window displaying a directory structure. A red box highlights the 'cv000_29590.txt' file in the editor and its corresponding entry in the file explorer. The file explorer also shows other text files and folders like 'NEG', 'POS', 'SmallTextDocs', and 'deception_dataConverted_final.csv'. The bottom section of the IDE shows an IPython console with some initial code execution results.

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\profa\Documents\Python Scripts\TextMining\DATA\POS\cv000_29590.txt

```
films adapted from comic books have had plenty of success , whether they
for starters , it was created by alan moore ( and eddie campbell ) , who
to say moore and campbell thoroughly researched the subject of jack the
the book ( or " graphic novel , " if you will ) is over 500 pages long &
in other words , don't dismiss this film because of its source .
if you can get past the whole comic book thing , you might find another
getting the hughes brothers to direct this seems almost as ludicrous as
the ghetto in question is , of course , whitechapel in 1888 london's east
it's a filthy , sooty place where the whores ( called " unfortunates " )
when the first stiff turns up , copper peter godley ( robbie coltrane ,
abberline , a widower , has prophetic dreams he unsuccessfully tries to
upon arriving in whitechapel , he befriends an unfortunate named mary ke
i don't think anyone needs to be briefed on jack the ripper , so i won't
in the comic , they don't bother cloaking the identity of the ripper ,
it's funny to watch the locals blindly point the finger of blame at jews
and from hell's ending had me whistling the stonecutters song from the s
don't worry - it'll all make sense when you see it .
now onto from hell's appearance : it's certainly dark and bleak enough ,
the print i saw wasn't completely finished ( both color and music had no
oscar winner martin childs' ( shakespeare in love ) production design to
even the acting in from hell is solid , with the dreamy depp turning in
ians holm ( joe gould's secret ) and richardson ( 102 dalmatians ) log i
i cringed the first time she opened her mouth , imagining her attempt at
the film , however , is all good .
```

File explorer

Name	Size	Type	Date
urls_pos.txt	598 KB	txt File	4/12
urls_unsup.txt	2.3 MB	txt File	4/12
NEG		File Folder	12/2
neg_movie2		File Folder	12/2
POS		File Folder	12/2
cv000_29590.txt	4 KB	txt File	2/15
cv001_18431.txt	4 KB	txt File	2/15
cv002_15918.txt	2 KB	txt File	2/15
cv003_11664.txt	5 KB	txt File	2/15
cv004_11636.txt	3 KB	txt File	2/15
pos_movie2		File Folder	12/2
SmallTextDocs		File Folder	12/2
deception_dataConverted_final.csv	0 bytes	csv File	12/2

File explorer Variable explorer Help

IPython console

```
Console 1/A
```

Unnamed: 54
Unnamed: 55
Name: 0, dtype: object

Using SMALL Datasets to Practice: Our Data

The screenshot shows a development environment with two main windows: Spyder (Python 3.6) and Notepad.

Spyder (Python 3.6) Window:

- Toolbar:** File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, Help.
- Path:** C:\Users\profa\Documents\Python Scripts\TextMining
- Editor:** C:\Users\profa\Documents\Python Scripts\TextMining\DATA\SmallTextDocs\Dog.txt
- Content:** .py, nltkExample1.py, Example NLTK sklearn senti and processing.py, Dog.txt
- Dog.txt Content:** Dogs are fun. I like dogs. Having many dogs can be fun too.
- File explorer:** Shows a file structure:
 - Name
 - labeledBow.feat
 - urls_neg.txt
 - urls_pos.txt
 - > Movie_train
 - > NEG
 - > neg_movie2
 - > POS
 - > pos_movie2
 - > SmallTextDocs
 - Dog.txt
 - Hike.txt

Notepad Window:

- Title:** Hike - Notepad
- Toolbar:** File, Edit, Format, View, Help
- Content:** Hiking, or going on long mountain hikes is a great way to relax. Hike with your dog!

A red line connects the "Dog.txt" file in the Spyder file explorer to the "Dog.txt" file in the Notepad window, and another red line connects the "Hike.txt" file in the Spyder file explorer to the "Hike - Notepad" window.

```
118 import sklearn
119 from sklearn.feature_extraction.text import CountVectorizer
120 MyVectorizer1=CountVectorizer(
121     input='content', ## can be set as 'content', 'file', or 'filename'
122     #If set as 'filename', the **sequence passed as an argument to fit**
123     #is expected to be a list of filenames
124     #https://scikit-Learn.org/stable/modules/generated/
125     ##sklearn.feature_extraction.text.CountVectorizer.html#
126     ##examples-using-sklearn-feature-extraction-text-countvectorizer
127     encoding='latin-1',
128     decode_error='ignore', #{'strict', 'ignore', 'replace'}
129     strip_accents=None, # {'ascii', 'unicode', None}
130     lowercase=True,
131     preprocessor=None,
132     tokenizer=None,
133     #stop_words='english', #string {'english'}, list, or None (default)
134     stop_words=None,
135     token_pattern='(\\w+\\W+\\w+)', #Regular expression denoting what constitutes
136     ngram_range=(1, 1),
137     analyzer='word',
138     max_df=1.0, # ignore terms w document freq strictly > threshold
139     min_df=1,
140     max_features=None,
141     vocabulary=None,
142     binary=False, #If True, all non zero counts are set to 1
143     #dtype=<class 'numpy.int64'>
144 )
145 #print(MyVectorizer1)
146 MyVect2=CountVectorizer()
147 MyVect3=CountVectorizer(input='filename')
148
```

Count Vectorizers

In this next example, I will use MyVect3. Notice that input='filename'. I need this as I am reading in files from folders.

Using the OS to read in the list of files

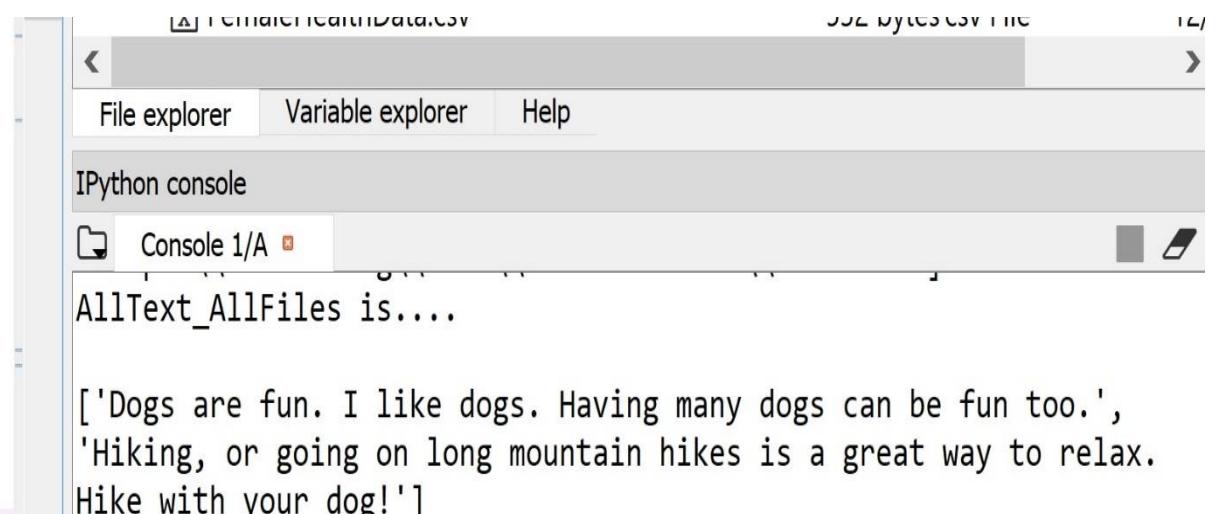
The screenshot shows a Jupyter Notebook environment with three main panes:

- Code Editor (Left):** Displays Python code for listing files in a directory. A red arrow points from the code editor to the "File explorer" pane.
- File Explorer (Top Right):** Shows a file tree with a folder named "SmallTextDocs" containing "Dog.txt" and "Hike.txt". A red box highlights this folder. Another red arrow points from the code editor to this pane.
- IPython Console (Bottom Right):** Displays the output of the executed code. It includes:
 - CORPUS EXAMPLES
 - calling os...
 - ['Dog.txt', 'Hike.txt']
 - ['Dog.txt', 'Hike.txt']
 - C:\Users\profa\Documents\Python Scripts\TextMining\DATA\SmallTextDocs\Dog.txt
 - C:\Users\profa\Documents\Python Scripts\TextMining\DATA\SmallTextDocs\Hike.txt
 - DONE...
 - full list...
 - ['C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\SmallTextDocs\\\\Dog.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\SmallTextDocs\\\\Hike.txt']A red arrow points from the code editor to the console output.

```
152#import os
153all_file_names = []
154path="C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\SmallTextDocs"
155print("calling os...")
156print(os.listdir(path))
157FileNameList=os.listdir(path)
158print(FileNameList)
159ListOfCompleteFiles=[]
160for name in os.listdir(path):
161    print(path+ "\\" + name)
162    next=path+ "\\" + name
163    ListOfCompleteFiles.append(next)
164print("DONE...")
165print("full list...")
166print(ListOfCompleteFiles)
167
168
169
170
171
172
173
174
175
```

Read in the Text

```
168AllText_AllFiles=[]
169for file in ListOfCompleteFiles:
170    FILE=open(file)
171    # print(FILE.read())
172    content=FILE.read()
173    AllText_AllFiles.append(content)
174    FILE.close()
175print("AllText_AllFiles is....\n")
176print(AllText_AllFiles)
177
```



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Displays the file name "F:\Data\TextData.csv" and the cell number "JUL 8 2018 11:41 AM 14".
- Toolbar:** Includes "File explorer", "Variable explorer", and "Help" buttons.
- Console Tab:** Labeled "Console 1/A".
- Output:** Shows the execution of the code and its output:
 - The variable "AllText_AllFiles" is defined as a list.
 - The list contains three strings:
 - 'Dogs are fun. I like dogs. Having many dogs can be fun too.'
 - 'Hiking, or going on long mountain hikes is a great way to relax.'
 - 'Hike with your dog!'

Fit Transforms...

```
175
176
177print("FIT TRANSFORM-----")  
178## FIT TRANSFORM USING a CountVect  
179print("The ListOfCompleteFiles is ...")  
180print(ListOfCompleteFiles)  
181X3=MyVect3.fit_transform(ListOfCompleteFiles)  
182print("The AllText_AllFiles is...")  
183print(AllText_AllFiles)  
184X2=MyVect2.fit_transform(AllText_AllFiles)  
185print(type(X2))  
186print(type(X3))  
187print(X2.get_shape())  
188ColumnNames2=MyVect2.get_feature_names()  
189ColumnNames3=MyVect3.get_feature_names()  
190print("The col name for 3 ", ColumnNames3)  
191
192# import pandas as pd
193#https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/
194## !!!!!!! The final DTM in Python!! (this took 20 hours :)  
195CorpusDF_A2=pd.DataFrame(X2.toarray(),columns=ColumnNames2)  
196print("The DF for 2 is...", CorpusDF_A2)  
197CorpusDF_A=pd.DataFrame(X3.toarray(),columns=ColumnNames3)  
198print("The DF for 3 is...", CorpusDF_A)
```

FIT TRANSFORM-----

The ListOfCompleteFiles is ...

['C:\\Users\\profa\\Documents\\Python Scripts\\TextMining\\DATA\\SmallTextDocs\\Dog.txt', 'C:\\Users\\profa\\Documents\\Python Scripts\\TextMining\\DATA\\SmallTextDocs\\Hike.txt']

The AllText_AllFiles is...

['Dogs are fun. I like dogs. Having many dogs can be fun too.', 'Hiking, or going on long mountain hikes is a great way to relax. Hike with your dog!']

<class 'scipy.sparse.csr.csr_matrix'>

<class 'scipy.sparse.csr.csr_matrix'>

(2, 25)

The col name for 3 ['are', 'be', 'can', 'dog', 'dogs', 'fun', 'going', 'great', 'having', 'hike', 'hikes', 'hiking', 'is', 'like', 'long', 'many', 'mountain', 'on', 'or', 'relax', 'to', 'too', 'way', 'with', 'your']

The DF for 2 is... are be can dog dogs fun ... relax to too way with your

	0	1	1	1	0	3	2	...	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	...	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

[2 rows x 25 columns]

The DF for 3 is... are be can dog dogs fun ... relax to too way with your

	0	1	1	1	0	3	2	...	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	...	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

[2 rows x 25 columns]

So far

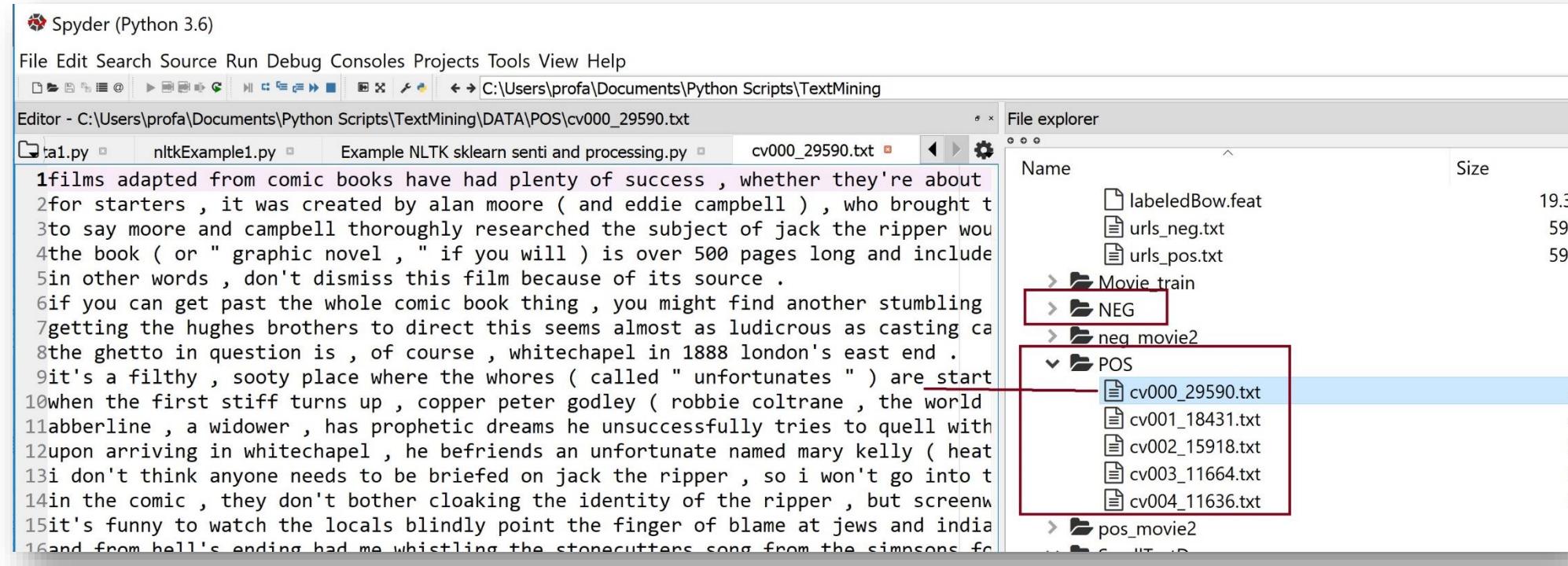
- 1) We have arrays of numbers and column names.
- 2) But – we do not yet have a proper dataframe....
- 3) We can confirm the counts – and should!
- 4) We also showed that we can read in text or files.
- 5) We also have dog and dogs seperately, etc...

```
The DF for 2 is...    are  be  can  dog  dogs  fun  ...  relax  to  too  way  with  your
0    1    1    1    0    3    2    ...      0    0    1    0    0    0
1    0    0    0    1    0    0    ...      1    1    0    1    1    1
```

[2 rows x 25 columns]

```
The DF for 3 is...    are  be  can  dog  dogs  fun  ...  relax  to  too  way  with  your
0    1    1    1    0    3    2    ...      0    0    1    0    0    0
1    0    0    0    1    0    0    ...      1    1    0    1    1    1
```

Next Step -



- 1) We will take what we have learned, and repeat this process.
- 2) The goal this time is to have a cleaner and easier to view DF
- 3) We want the labels and column names to part of the DF...
- 4) In this next example set, we will use a folder called POS that contains many text files all with positive sentiment and a folder called NEG with text files with negative sentiment.

Example2:

```
235## Step 1: Read in the POS files corpus into a DF1
236print("Building Vecotrizer....")
237MyVect4=CountVectorizer(input='filename',
238                         analyzer = 'word',
239                         stop_words='english',
240                         token_pattern='(?!u)[a-zA-Z]+'
241                         )
242path="C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS"
243
244##Create empty list
245POSListOfCompleteFiles=[]
246for name in os.listdir(path):
247    print(path + "\\\" + name)
248    next=path+ "\\\" + name
249    POSListOfCompleteFiles.append(next)
250
251print("POS full list...")
252print(POSListOfCompleteFiles)
253
254
255
256
257
```

The screenshot shows a Jupyter Notebook interface with two panes. The left pane displays Python code for reading a POS file corpus and creating a CountVectorizer. The right pane shows the execution environment with a file browser, variable explorer, and IPython console. The IPython console output shows the progress of building the vectorizer and lists the five POS files: cv000_29590.txt, cv001_18431.txt, cv002_15918.txt, cv003_11664.txt, and cv004_11636.txt.

```
[2 rows x 25 columns]
COMPLETE -----
Building Vecotrizer....
C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv000_29590.txt
C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv001_18431.txt
C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv002_15918.txt
C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv003_11664.txt
C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv004_11636.txt
POS full list...
['C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv000_29590.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv001_18431.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv002_15918.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv003_11664.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv004_11636.txt']
```

```

244##Create empty list
245POSListOfCompleteFiles=[]
246for name in os.listdir(path):
247    print(path + "\\" + name)
248    next=path+ "\\" + name
249    POSListOfCompleteFiles.append(next)
250
251print("POS full list...")
252print(POSListOfCompleteFiles)
253
254print("FIT with Vectorizer...")
255X4=MyVect4.fit_transform(POSListOfCompleteFiles)
256print(type(X4))
257print(X4.get_shape())
258POSColumnNames=MyVect4.get_feature_names()
259print("Column names: ", POSColumnNames[0:10])
260print("Building DF....")
261#import pandas as pd
262POS_CorpusDF_A=pd.DataFrame(X4.toarray(),columns=POSColumnNames)
263print(POS_CorpusDF_A)
264
265## This Looks good. Notice that when I built the Vectorizer
266## above, that I used [a-zA-Z] which means
267## letter ONLY - no numbers.
268
269
270
271

```

File explorer Variable explorer Help

IPython console

Console 1/A

POS full list...

```
[ 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv000_29590.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv001_18431.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv002_15918.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv003_11664.txt', 'C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\POS\\\\cv004_11636.txt']
```

FIT with Vectorizer...

```
<class 'scipy.sparse.csr.csr_matrix'>
(5, 1139)
```

Column names: ['abberline', 'ably', 'aboard', 'absinthe', 'absolutely', 'academy', 'accent', 'achiever', 'act', 'acting']

Building DF....

	abberline	ably	aboard	absinthe	...	years	yglesias	york	young
0	2	1	0	1	...	0	1	0	0
1	0	0	0	0	...	0	0	0	0
2	0	0	0	0	...	0	0	0	0
3	0	0	1	0	...	1	0	1	1
4	0	0	0	0	...	0	0	1	0

Adding the Label for Sentiment

268

269## Now, we need to add a column for P or N.

270## I will call it PosORNeg and because all of these

271## are positive, I will fill it with P

272## DataFrame.insert(loc, column, value, allow_duplicates=False)

273#Length=POS_CorpusDF_A.shape

274#print(Length[0]) ## num of rows

275#print(Length[1]) ## num of columns

276

277## Add column

278print("Adding new column....")

279POS_CorpusDF_A["PosORNeg"]="P"

280print(POS_CorpusDF_A)

281

282## OK - now we have a Labeled DF

[5 rows x 1139 columns]

Adding new column....

	abberline	ably	aboard	absinthe	...	yglesias	york	young	PosORNeg
0	2	1	0	1	...	1	0	0	P
1	0	0	0	0	...	0	0	0	P
2	0	0	0	0	...	0	0	0	P
3	0	0	1	0	...	0	1	1	P
4	0	0	0	0	0	0	1	0	P

Do the Same Thing for the NEG Folder ...

```
289### Now - we will do the above for
290## the negative docs....
291pathN="C:\\\\Users\\\\profa\\\\Documents\\\\Python Scripts\\\\TextMining\\\\DATA\\\\NEG"
292##Create empty list
293NEGListOfCompleteFiles=[]
294for name in os.listdir(pathN):
295    print(pathN+ "\\" + name)
296    next=pathN+ "\\" + name
297    NEGListOfCompleteFiles.append(next)
298
299print("full list...")
300print(NEGListOfCompleteFiles)
301X5=MyVect4.fit_transform(NEGListOfCompleteFiles)
302print(type(X5))
303print(X5.get_shape())
304NEGColumnNames=MyVect4.get_feature_names()
305print(NEGColumnNames)
306
307#import pandas as pd
308NEG_CorpusDF_A=pd.DataFrame(X5.toarray(),columns=NEGColumnNames)
309print(NEG_CorpusDF_A)
310NEG_CorpusDF_A[ "PosORNeg"]="N"
311print(NEG_CorpusDF_A)
312
```

What's Next??

```
[5 rows x 1139 columns]
```

```
Adding new column....
```

	abberline	ably	aboard	absinthe	...	yglesias	york	young	PosORNeg
0	2	1	0	1	...	1	0	0	P
1	0	0	0	0	...	0	0	0	P
2	0	0	0	0	...	0	0	0	P
3	0	0	1	0	...	0	1	1	P
4	0	0	0	0	...	0	1	0	P

```
[5 rows x 1004 columns]
```

	able	abo	accent	accident	...	yellow	yes	young	PosORNeg
0	1	0	0	0	...	0	0	0	N
1	0	2	0	1	...	0	0	0	N
2	0	0	0	0	...	1	1	0	N
3	0	0	0	0	...	0	0	1	N
4	0	0	1	0	...	0	0	0	N

Next: Create a Complete DF with all data and shuffle – then build test and train sets.

1) MERGE

```
319## Create a new Large Pos and Neg DF
320## https://pandas.pydata.org/pandas-docs/stable/merging.h
321result = NEG_CorpusDF_A.append(POS_CorpusDF_A)
322print(result)
323## Replace the NaN with 0 because it actually
324## means none in this case
325result=result.fillna(0)
326print(result)
327
328
329
330
```

[5 rows x 1139 columns]

Adding new column....

	abberline	ably	aboard	absinthe	...	yglesias	york	young	PosORNeg
0	2	1	0	1	...	1	0	0	P
1	0	0	0	0	...	0	0	0	P
2	0	0	0	0	...	0	0	0	P
3	0	0	1	0	...	0	1	1	P
4	0	0	0	0	...	0	1	0	P

Shuffle

Why must we shuffle in this case?

```
343## SHUFFLE the DATAFRAME
344## df = df.sample(frac=1).reset_index(drop=True)
345## Here, specifying drop=True prevents .reset_index
346## from creating a column containing the old index entries.
347## the frac=1 means "resample" (shuffle) 100% of the data
348result=result.sample(frac=1).reset_index(drop=True)
349print(result)
350## This worked! You can see that the shape is the same
351## and that the label is no longer all N and then all P
352
353
354
355
356
357
358
359
360
```

File explorer variable explorer Help

IPython console

Console 1/A

[10 rows x 1893 columns]

	PosORNeg	abberline	able	ably	...	yes	yglesias	york	young
0	N	0.0	0.0	0.0	...	0.0	0.0	0.0	1
1	P	0.0	0.0	0.0	...	0.0	0.0	0.0	0
2	N	0.0	1.0	0.0	...	0.0	0.0	0.0	0
3	N	0.0	0.0	0.0	...	0.0	0.0	0.0	0
4	P	0.0	0.0	0.0	...	0.0	0.0	1.0	0
5	N	0.0	0.0	0.0	...	0.0	0.0	0.0	0
6	N	0.0	0.0	0.0	...	1.0	0.0	0.0	0
7	P	0.0	0.0	0.0	...	0.0	0.0	1.0	1
8	P	2.0	0.0	1.0	...	0.0	1.0	0.0	0
9	P	0.0	0.0	0.0	...	0.0	0.0	0.0	0

Test & Train Sets

```
345  
346  
347  
348  
349  
350  
351  
352  
353## From here, we can create (randomly) the test and train sets  
354# make results reproducible  
355import numpy as np  
356np.random.seed(140) ## or any number - does not matter  
357# sample without replacement  
358## I am choosing "6" here to make the training set  
359## of size 6. This will make the test set of size 4  
360## This can be any choice depending on YOU and your data  
361train_ix = np.random.choice(result.index, 6, replace=False)  
362df_training = result.iloc[train_ix]  
363df_test = result.drop(train_ix)  
364print("Training set...")  
365print(df_training)  
366print("Testing set...")  
367print(df_test)
```

Console 1/A

[10 rows x 1893 columns]

Training set...

	PosORNeg	abberline	able	ably	...	yes	yglesias	york	young
3	N	0.0	0.0	0.0	...	0.0	0.0	0.0	0
2	N	0.0	1.0	0.0	...	0.0	0.0	0.0	0
1	P	0.0	0.0	0.0	...	0.0	0.0	0.0	0
4	P	0.0	0.0	0.0	...	0.0	0.0	1.0	0
0	N	0.0	0.0	0.0	...	0.0	0.0	0.0	1
5	N	0.0	0.0	0.0	...	0.0	0.0	0.0	0

[6 rows x 1893 columns]

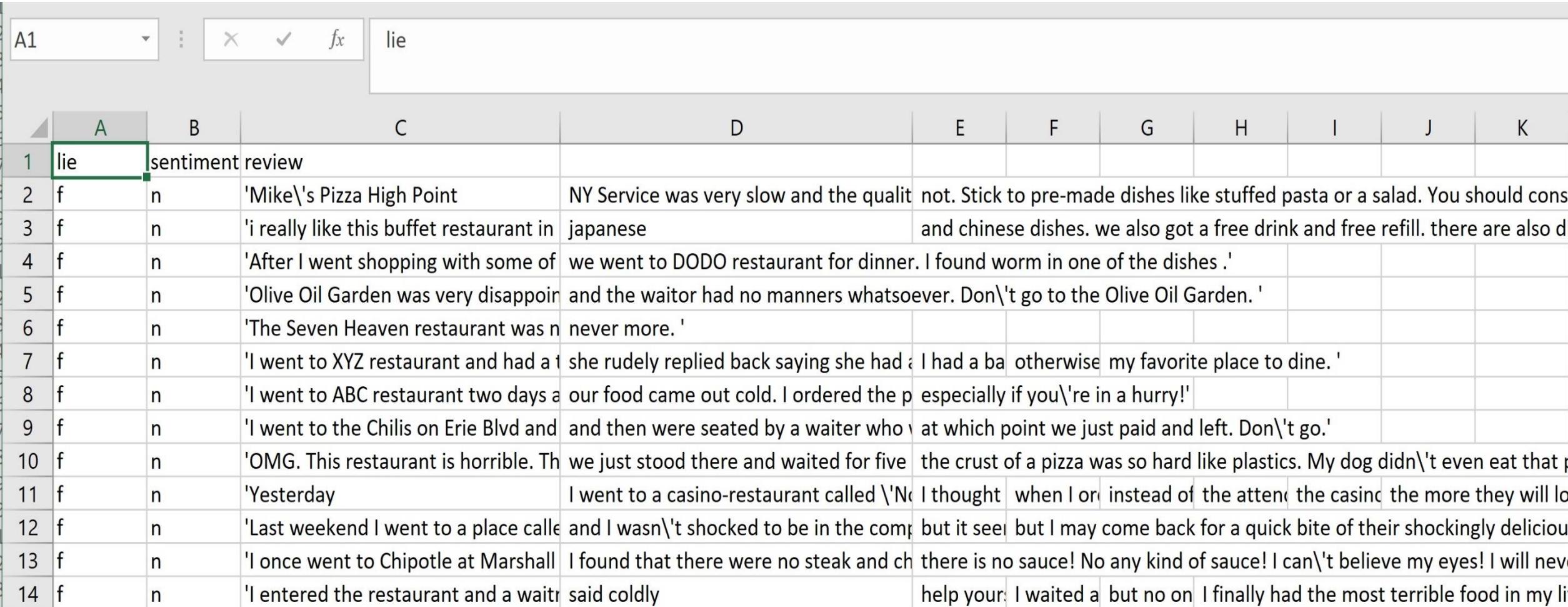
Testing set...

	PosORNeg	abberline	able	ably	...	yes	yglesias	york	young
6	N	0.0	0.0	0.0	...	1.0	0.0	0.0	0
7	P	0.0	0.0	0.0	...	0.0	0.0	1.0	1
8	P	2.0	0.0	1.0	...	0.0	1.0	0.0	0
9	P	0.0	0.0	0.0	...	0.0	0.0	0.0	0

[4 rows x 1893 columns]

Example 3: Reading in one .csv file...

More on this in the next ppt...



A	B	C	D	E	F	G	H	I	J	K
1	lie	sentiment review								
2	f	n	'Mike\'s Pizza High Point	NY Service was very slow and the qualit	not. Stick to pre-made dishes like stuffed pasta or a salad. You should cons					
3	f	n	'i really like this buffet restaurant in	japanese	and chinese dishes. we also got a free drink and free refill. there are also d					
4	f	n	'After I went shopping with some of	we went to DODO restaurant for dinner. I found worm in one of the dishes.'						
5	f	n	'Olive Oil Garden was very disappoint	and the waiter had no manners whatsoever. Don\'t go to the Olive Oil Garden.'						
6	f	n	'The Seven Heaven restaurant was n	never more.'						
7	f	n	'I went to XYZ restaurant and had a t	she rudely replied back saying she had a	I had a ba otherwise my favorite place to dine.'					
8	f	n	'I went to ABC restaurant two days a	our food came out cold. I ordered the p	especially if you're in a hurry!'					
9	f	n	'I went to the Chilis on Erie Blvd and	and then were seated by a waiter who	at which point we just paid and left. Don\'t go.'					
10	f	n	'OMG. This restaurant is horrible. Th	we just stood there and waited for five	the crust of a pizza was so hard like plastics. My dog didn't even eat that p					
11	f	n	'Yesterday	I went to a casino-restaurant called \'No	I thought when I or instead of the attend	the casin	the more they will lo			
12	f	n	'Last weekend I went to a place calle	and I wasn't shocked to be in the comp	but it see	but I may come back for a quick bite of their shockingly deliciou				
13	f	n	'I once went to Chipotle at Marshall	I found that there were no steak and ch	there is no sauce! No any kind of sauce! I can't believe my eyes! I will never					
14	f	n	'I entered the restaurant and a waitr	said coldly	help your	I waited a	but no on	I finally had the most terrible food in my li		