Python Libraries for Data Analysis and Machine Learning

Xu Weiwen & Yang Haoran

2022-4-1

Overview

- Environment Preparation for Python
- Python Libraries for Data Scientists
- Data Processing & Visualization Using Python
- Python for Basic Machine Learning Models

Environment Preparation for Python

We introduce

- **Anaconda** (https://www.anaconda.com/)
- **Jupyter Notebook** (https://jupyter.org/)

for Python environment.

Other alternatives:

- Text Editor + Command line
- IDE (Integrated Development Environment): PyCharm, Vscode, …

What is Anaconda?

The open-source Anaconda is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 19 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 7,500+ Python/R data science packages
- Analyze data with scalability and performance with Dask, **NumPy**, **pandas**, and Numba
- Visualize results with **Matplotlib**, Bokeh, Datashader, and Holoviews
- Develop and train machine learning and deep learning models with **scikit-learn**, TensorFlow, and Theano

Anaconda Installation

Please follow the instruction here to install the Anaconda (for Python 3.7)

https://www.anaconda.com/distribution/#download-section

It provides different versions to suit different OS. Please select the one you are using.

Just install according to the default setting, and the environment variables will be automatically configured after installation.

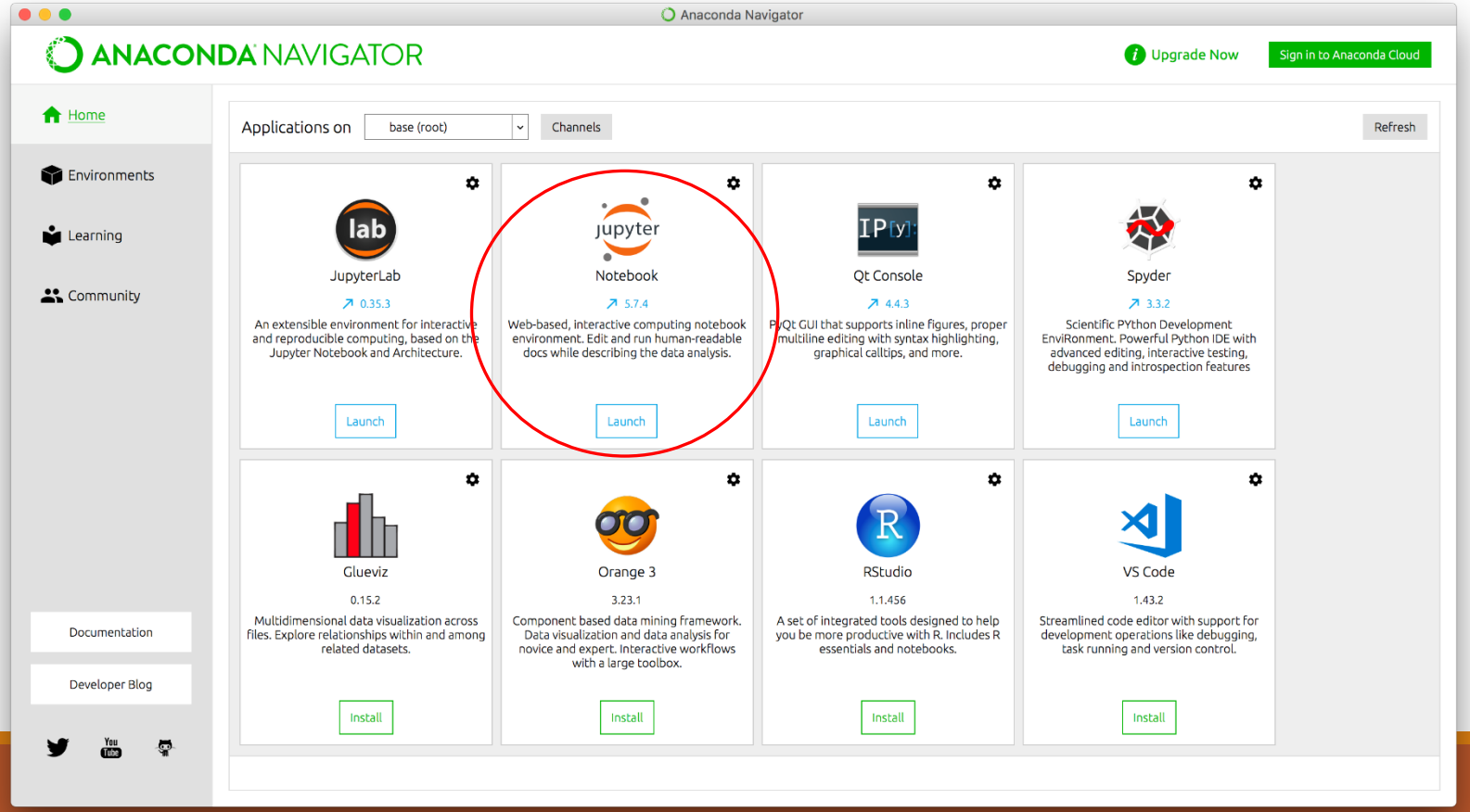
What is Jupyter Notebook?

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Jupyter Notebook is included in the Anaconda.

Basic Operation on Jupyter Notebook

After installing the Anaconda, open Anaconda-Navigator as below, and you can find the Jupyter Notebook on the Anaconda. Then click Launch.



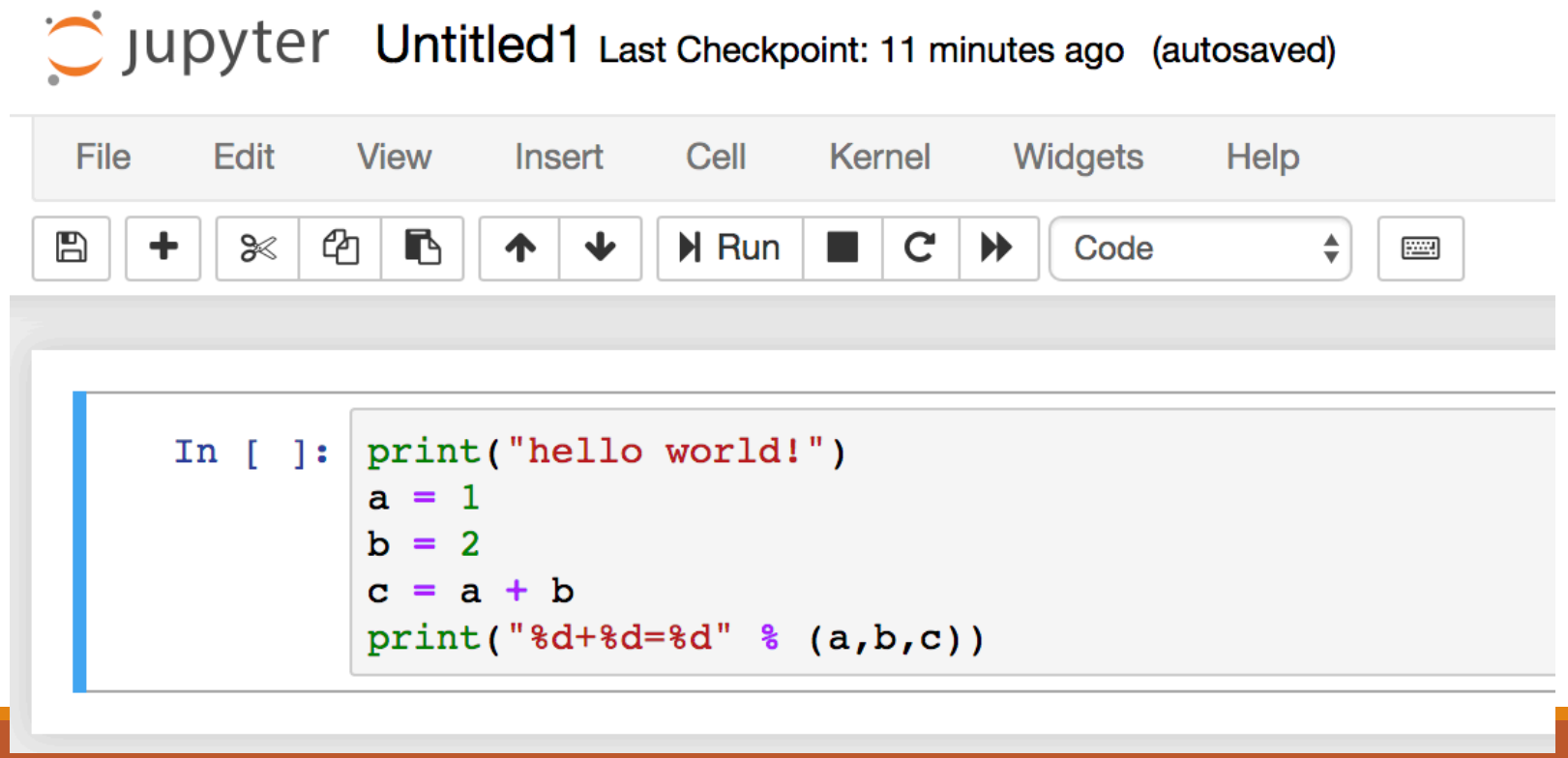
Basic Operation on Jupyter Notebook

Jupyter Notebook is presented as a website. Select the path, then under the button “New”, choose “Python 3” to open a new python file.

Basic Operation on Jupyter Notebook

Type the code into the input box on Jupyter.

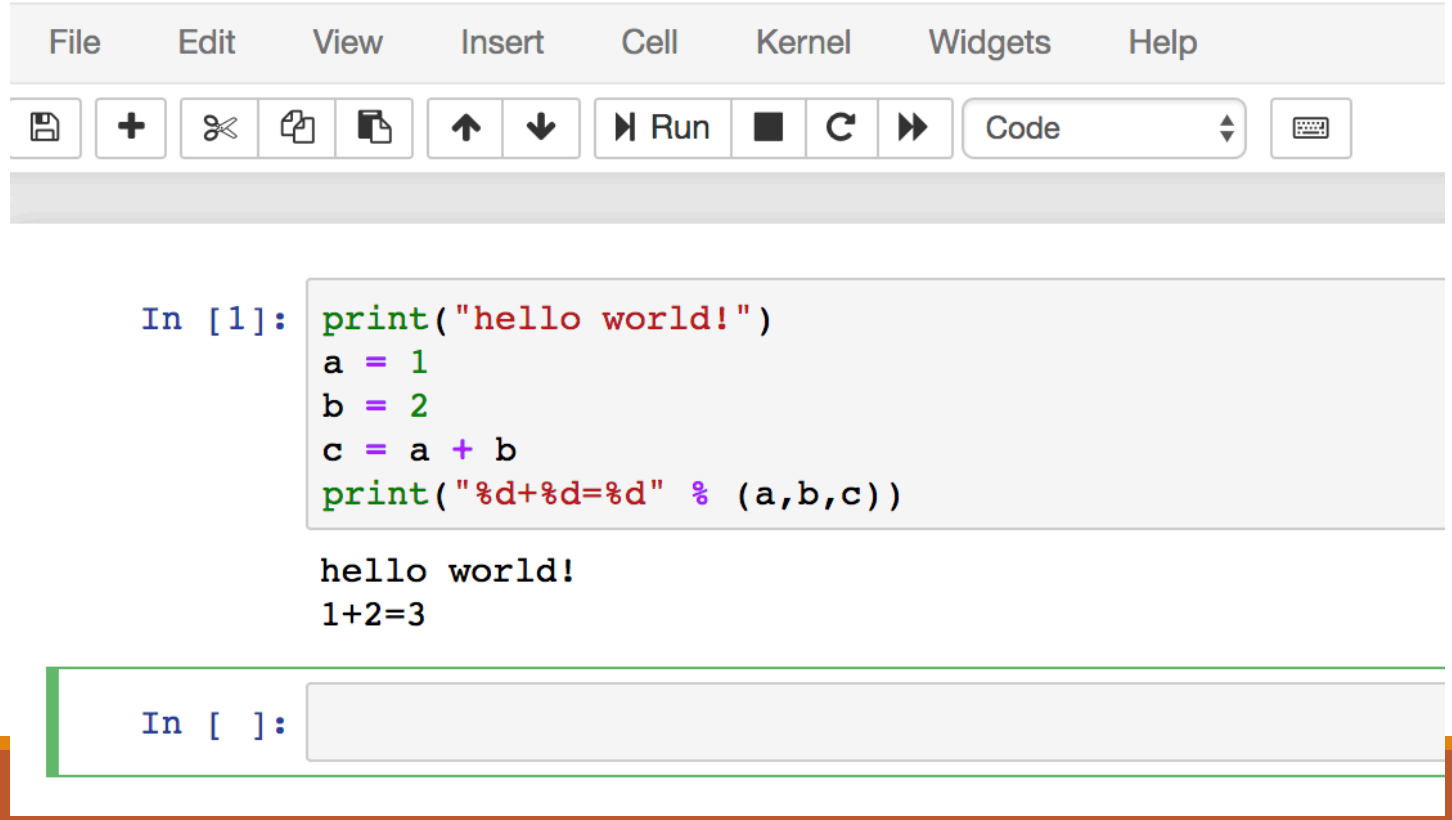
Get started learning Python: https://www.learnpython.org/



Basic Operation on Jupyter Notebook

Click “Run”.

The output will be shown in the blank area right below the input box.

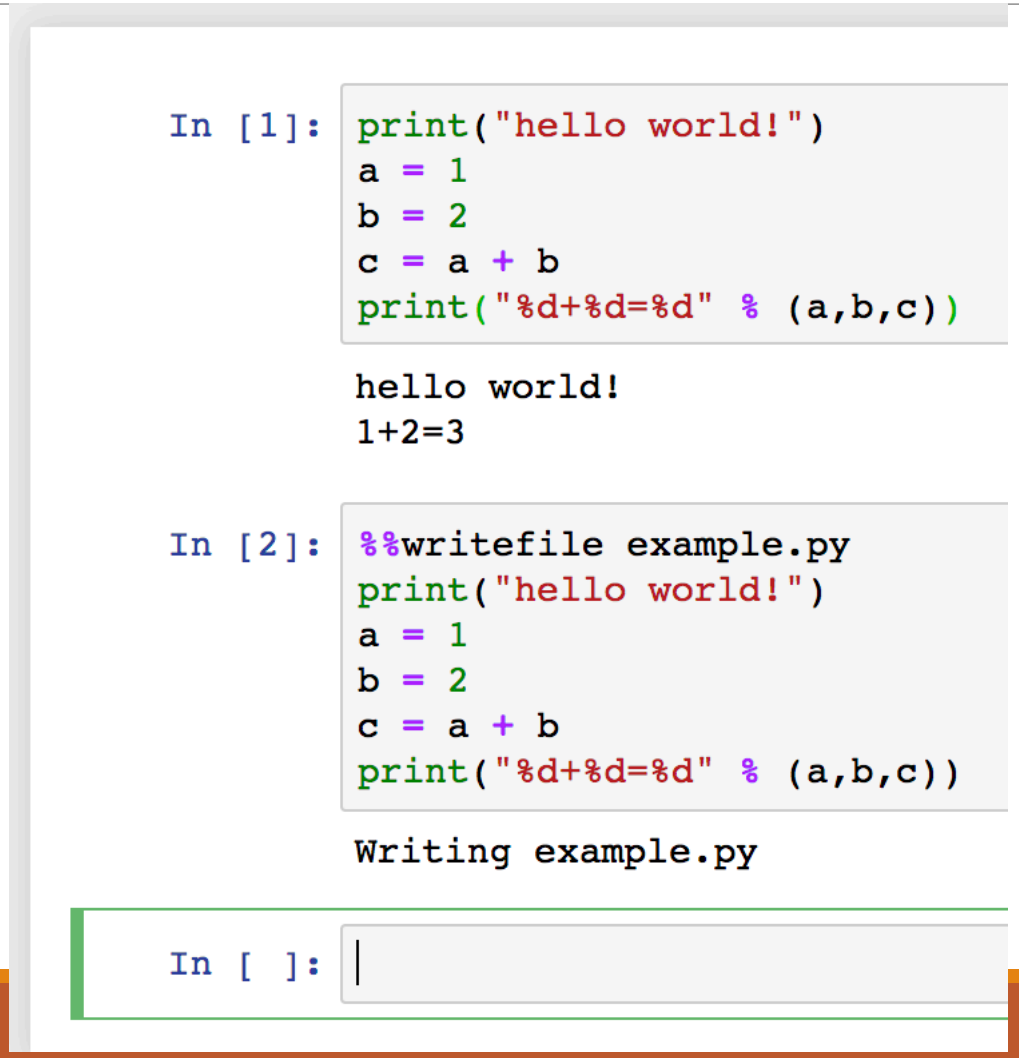


Basic Operation on Jupyter Notebook

Jupyter Notebook will help you save your code automatically in ".ipynb" format.

If you want to save the code as ".py" format.

Here, we just use ".ipynb" format.



Python Libraries for Data Scientists

Python toolboxes/libraries for data processing:

- NumPy
- SciPy
- Pandas

Visualization libraries

- matplotlib
- Seaborn

Machine learning & deep learning

- Scikit-learn
- Tensorflow/Pytorch/Theano

and many more …

Python Libraries for Data Scientists

NumPy:

- introduces objects for multidimensional arrays and matrices, as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects
- provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance
- many other python libraries are built on NumPy

Link: http://www.numpy.org/

Python Libraries for Data Scientists

SciPy:

- collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics and more
- part of SciPy Stack
- built on NumPy

Link: https://www.scipy.org/scipylib/

Python Libraries for Data Scientists

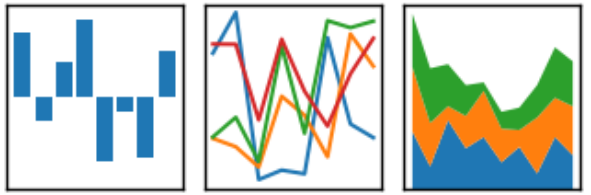
Pandas:

- adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R)
- provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- allows handling missing data

Link: http://pandas.pydata.org/

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Python Libraries for Data Scientists

matplotlib:

- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB
- line plots, scatter plots, barcharts, histograms, pie charts etc.
- relatively low-level; some effort needed to create advanced visualization

Link: https://matplotlib.org/

Python Libraries for Data Scientists

Seaborn:

- based on matplotlib
- provides high level interface for drawing attractive statistical graphics
- Similar (in style) to the popular ggplot2 library in R

Link: https://seaborn.pydata.org/

Python Libraries for Data Scientists

SciKit-Learn:

- provides machine learning algorithms: classification, regression, clustering, model validation etc.

- built on NumPy, SciPy and matplotlib

Link: http://scikit-learn.org/

Loading Python Libraries

```
In [1]: #Import Python Libraries
        import numpy as np
        import scipy as sp
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

Press `Shift+Enter` to execute the *jupyter* cell, or just click "Run".

Reading data using pandas

```
In [4]: df = pd.read_csv("http://www1.se.cuhk.edu.hk/~eclt5810/lecture/weka_tutorial/bank.csv")
```

There is a number of pandas commands to read other data formats:

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None, na_values=['NA'])
pd.read_stata('myfile.dta')
pd.read_sas('myfile.sas7bdat')
pd.read_hdf('myfile.h5','df')
```

Exploring data frames

In [5]:
```
#List first 5 records
df.head()
```

Out[5]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no

✓ Try to read the first 10, 20, 50 records
✓ Try to view the last few records

Data Frame data types

Pandas Type	Native Python Type	Description
object	string	The most general dtype. Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters. 64 refers to the memory allocated to hold this character.
float64	float	Numeric characters with decimals. If a column contains numbers and NaNs, pandas will default to float64, in case your missing value has a decimal.
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	Values meant to hold time data. Look into these for time series experiments.

Data Frame data types

```
In [7]: #Check a particular column type
        df['age'].dtype

Out[7]: dtype('int64')

In [8]: #Check types for all the columns
        df.dtypes
```

```
Out[8]: age            int64
        job           object
        marital       object
        education     object
        default       object
        balance        int64
        housing       object
        loan          object
        contact       object
        day            int64
        month         object
        duration       int64
        campaign       int64
        pdays          int64
        previous       int64
        poutcome      object
        y             object
        dtype: object
```

Data Frames attributes

Python objects have *attributes* and *methods*.

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

Data Frames attributes

```
In [11]: #Find how many records this data frame has
         df.shape

Out[11]: (4521, 17)

In [12]: #How many elements are there?
         df.size

Out[12]: 76857

In [13]: #What are the column names?
         df.columns

Out[13]: Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
                'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
                'previous', 'poutcome', 'y'],
               dtype='object')
```

Data Frames methods

Unlike attributes, python methods have *parenthesis.*
All attributes and methods can be listed with a *dir()* function: `dir(df)`

df.method()	description
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Data Frames methods

In [14]:
```
#Give the summary for the numeric columns in the dataset
df.describe()
```

Out[14]:

	age	balance	day	duration	campaign	pdays	previous
count	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000	4521.000000
mean	41.170095	1422.657819	15.915284	263.961292	2.793630	39.766645	0.542579
std	10.576211	3009.638142	8.247667	259.856633	3.109807	100.121124	1.693562
min	19.000000	-3313.000000	1.000000	4.000000	1.000000	-1.000000	0.000000
25%	33.000000	69.000000	9.000000	104.000000	1.000000	-1.000000	0.000000
50%	39.000000	444.000000	16.000000	185.000000	2.000000	-1.000000	0.000000
75%	49.000000	1480.000000	21.000000	329.000000	3.000000	-1.000000	0.000000
max	87.000000	71188.000000	31.000000	3025.000000	50.000000	871.000000	25.000000

Data Frames methods

```
In [15]: #Calculate standard deviation for all numeric columns
         df.std()
```

```
Out[15]: age           10.576211
         balance     3009.638142
         day            8.247667
         duration     259.856633
         campaign       3.109807
         pdays        100.121124
         previous       1.693562
         dtype: float64
```

```
In [17]: #What are the mean values for all numeric columns?
         df.mean()
```

```
Out[17]: age           41.170095
         balance     1422.657819
         day           15.915284
         duration     263.961292
         campaign       2.793630
         pdays         39.766645
         previous       0.542579
         dtype: float64
```

Selecting a column in a Data Frame

```
In [22]: #Subset the data frame using column name
         df['job'][:5]

Out[22]: 0     unemployed
         1       services
         2     management
         3     management
         4    blue-collar
         Name: job, dtype: object

In [23]: #Use the column name as an attribute
         df.job[:5]

Out[23]: 0     unemployed
         1       services
         2     management
         3     management
         4    blue-collar
         Name: job, dtype: object
```

Note: If we want to select a column with a name as the attribute in DataFrames we should use method 1.

E.G., Since there is an attribute – *rank* in DataFrame, if we want to select the column 'rank', we should use df['rank'], and cannot use method 2, i.e., df.rank, which will return the attribute *rank* of the data frame instead of the column "rank".

Data Frame: filtering

To subset the data we can apply Boolean indexing. This indexing is commonly known as a filter. For example if we want to subset the rows in which the age value is greater than 50:

```
In [31]: #Subset the rows in which the age value is greater than 50
         df_sub = df[ df['age'] > 50 ]
```

Any Boolean operator can be used to subset the data:

> greater; >= greater or equal;

< less; <= less or equal;

== equal; != not equal;

```
In [32]: #Select only those rows whose education level is primary
         df_primary = df[ df['education'] == 'primary' ]
```

Data Frames: Slicing

There are a number of ways to subset the Data Frame:

- one or more columns
- one or more rows
- a subset of rows and columns

Rows and columns can be selected by their position or label

Data Frames: Slicing

When selecting one column, it is possible to use single set of brackets, but the resulting object will be a Series (not a DataFrame):

```
In [ ]: #Select column age:
        df['age']
```

When we need to select more than one column and/or make the output to be a DataFrame, we should use double brackets:

```
In [ ]: #Select column age and job:
        df[['age','job']]
```

Data Frames: Selecting rows

If we need to select a range of rows, we can specify the range using ":"

```
In [ ]: #Select rows by their position:
        df[10:20]
```

Notice that the first row has a position 0, and the last value in the range is omitted:
So for 0:10 range the first 10 rows are returned with the positions starting with 0 and ending with 9

Graphics to explore the data

Seaborn package is built on matplotlib but provides high level interface for drawing attractive statistical graphics, similar to ggplot2 library in R. It specifically targets statistical data visualization

To show graphs within Python notebook include inline directive:

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

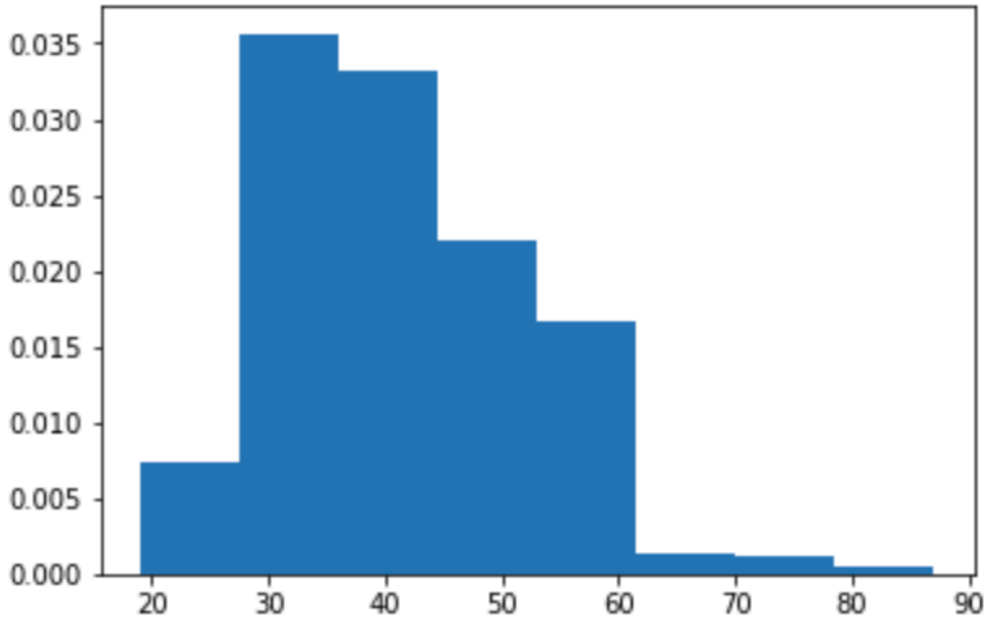
Graphics

	description
histplot	histogram
barplot	estimate of central tendency for a numeric variable
violinplot	similar to boxplot, also shows the probability density of the data
jointplot	Scatterplot
regplot	Regression plot
pairplot	Pairplot
boxplot	boxplot
swarmplot	categorical scatterplot
factorplot	General categorical plot

Draw Histogram Using Matplotlib

```
In [31]: #Use matplotlib to draw a histogram of a age data
         plt.hist(df['age'],bins=8, density=1)
```

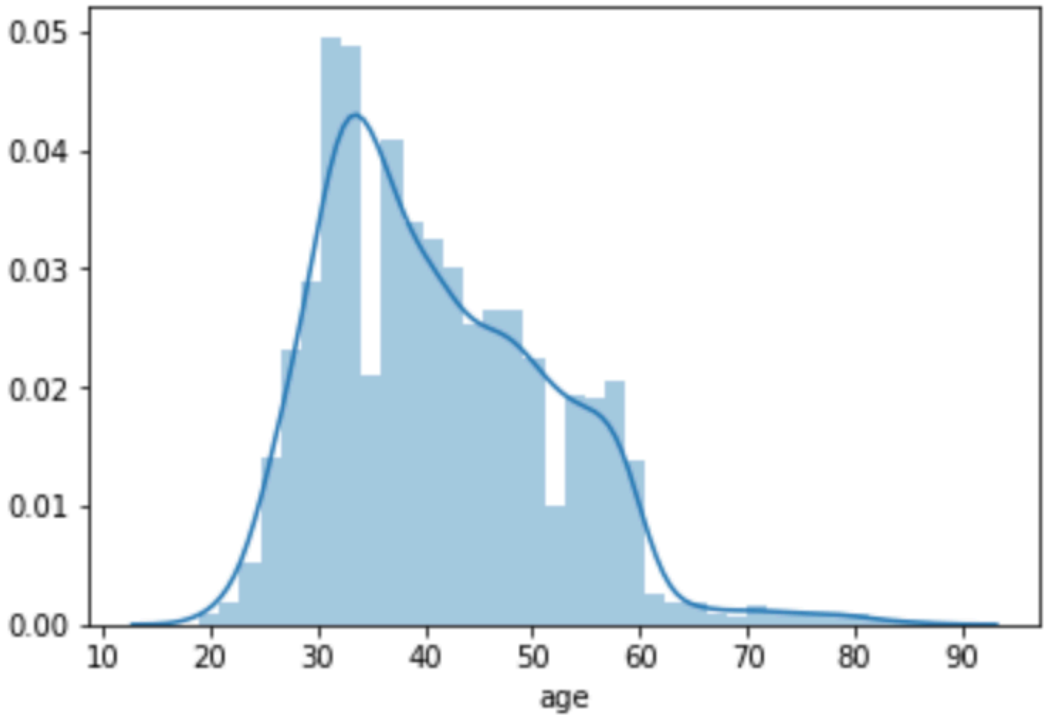
```
Out[31]: (array([0.0073383 , 0.03565062, 0.03320452, 0.02193684, 0.01662828,
                 0.00130112, 0.00111896, 0.0004684 ]),
          array([19. , 27.5, 36. , 44.5, 53. , 61.5, 70. , 78.5, 87. ]),
          <a list of 8 Patch objects>)
```



Draw Histogram Using Seaborn

```
In [33]: #Use seaborn package to draw a histogram
         sns.distplot(df[('age')])
```

```
Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x1a21b0cb00>
```



Python for Machine Learning

Machine learning: the problem setting:
In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), it is said to have several attributes or features.

We can separate learning problems in a few large categories:

- Supervised Learning (https://sklearn.org/supervised_learning.html#supervised-learning)
  - Classification
  - Regression
- Unsupervised Learning (https://sklearn.org/unsupervised_learning.html#unsupervised-learning)
  - Clustering

Python for Machine Learning

Training set and testing set:

Machine learning is about learning some properties of a data set and applying them to new data. This is why a common practice in machine learning to evaluate an algorithm is to split the data at hand into two sets, one that we call the **training set** on which we learn data properties and one that we call the **testing set** on which we test these properties.

scikit-learn comes with a few standard datasets, for instance the **iris** and **digits** datasets for **classification** and the **boston house prices** dataset for **regression**.

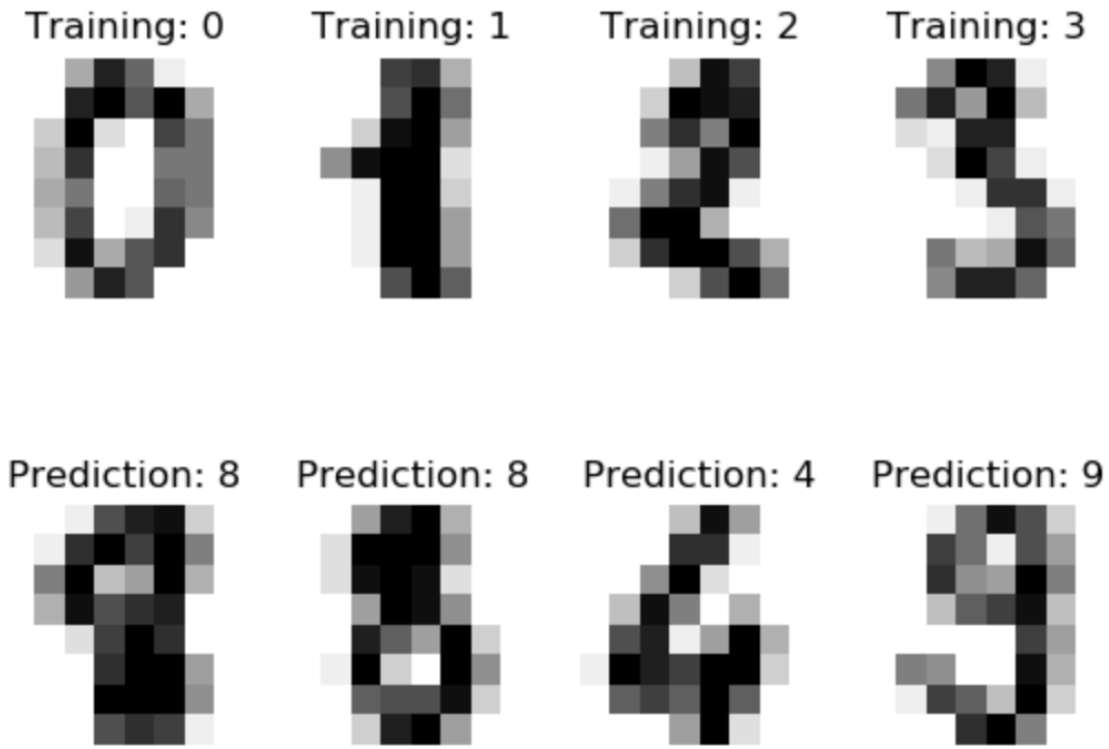
Loading an example dataset

```
In [61]: import sklearn
         from sklearn import datasets
         iris = datasets.load_iris()
         digits = datasets.load_digits()
```

A dataset is a dictionary-like object that holds all the data and some metadata about the data. This data is stored in the *.data* member, which is a *(n_samples, n_features)* array. In the case of supervised problem, one or more response variables are stored in the *.target* member.

Loading an example dataset - *digits*

An example showing how the scikit-learn can be used to recognize images of hand-written digits.



Loading an example dataset - *digits*

For instance, in the case of the digits dataset, *digits.data* gives access to the features that can be used to classify the digits samples:

```
In [62]: print(digits.data)
```

```
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]
```

and *digits.target* gives the ground truth for the digit dataset, that is the number corresponding to each digit image that we are trying to learn:

```
In [63]: print(digits.target)
```

```
[0 1 2 ... 8 9 8]
```

Learning and predicting

In the case of the *digits* dataset, the task is to predict, given an image, which digit it represents. We are given samples of each of the 10 possible classes (the digits *zero* through *nine*) on which we *fit* a **classifier** to be able to *predict* the classes to which unseen samples belong.

In scikit-learn, a classifier for classification is a Python object that implements the methods fit(X, y) and predict(T).

An example of a classifier is the class sklearn.svm.SVC, which implements **support vector classification**. The classifier's constructor takes as arguments the model's parameters.

Learning and predicting

For now, we will consider the classifier as a black box:

```
In [41]: from sklearn import svm
         clf = svm.SVC(gamma=0.001, C=100.)
```

Choosing the parameters of the model

In this example, we set the value of gamma manually. To find good values for these parameters, we can use tools such as grid search and cross validation.

Learning and predicting

For the training set, we'll use all the images from our dataset, except for the last image, which we'll reserve for our predicting. We select the training set with the [:-1] Python syntax, which produces a new array that contains all but the last item from digits.data:

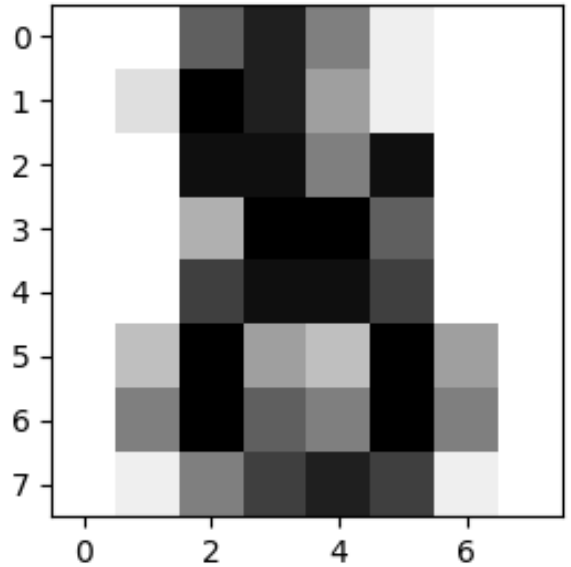
```
In [42]: clf.fit(digits.data[:-1], digits.target[:-1])

Out[42]: SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)
```

Learning and predicting

Now you can *predict* new values. In this case, you'll predict using the last image from digits.data. By predicting, you'll determine the image from the training set that best matches the last image.

```
In [43]: clf.predict(digits.data[-1:])
Out[43]: array([8])
```

The corresponding image is:



Text Processing (English/Chinese)

Introduction

In any machine learning task, cleaning or preprocessing the data is as important as model building if not more. And when it comes to unstructured data like text, this process is even more important.

Hilarious😂 !!!!

Want to know more. Checkout www.h2o.ai for additional information

thnks for readin the notebook

香港是一個國際化的大都市 \o/ :-)

９月１６日 (9月16日)

Objective

- To understand the various text preprocessing steps with code examples
- Some of the common text preprocessing / cleaning steps are:

English

- Lower casing
- Removal of Punctuations
- Removal of Frequent / Rare words
- Stemming / Lemmatization
- Removal of emojis / emoticons
- Removal of URLs / HTML tags

Chinese

- Conversion between Full / Half width
- Conversion between Traditional / Simplified words
- Chinese Word Segmentation

Lower Casing

- This is more helpful for text featurization techniques like frequency, tfidf as it helps to combine the same words together thereby reducing the duplication and get correct counts / tfidf values.
- This may not be helpful when we do tasks like Part of Speech tagging (where proper casing gives some information about Nouns and so on) and Sentiment Analysis (where upper casing refers to anger and so on)

Lower Casing

Let's use the "sample.csv" file as an example. It has many columns, while we are only interested in the "text" column.

We create a new column with all characters lower cased.

For string data structure, it has built-in function called .lower()

```
import pandas as pd
import re
import nltk
import string
pd.options.mode.chained_assignment = None
```

```
full_df = pd.read_csv("./sample.csv", nrows=5000)
df = full_df[["text"]]
df["text"] = df["text"].astype(str)
full_df.head()
```

	tweet_id	author_id	inbound	created_at	text	response_tweet_id	in_resp
0	119237	105834	True	Wed Oct 11 06:55:44 +0000 2017	@AppleSupport causing the reply to be disregar...	119236	
1	119238	ChaseSupport	False	Wed Oct 11 13:25:49 +0000 2017	@105835 Your business means a lot to us. Pleas...	NaN	
2	119239	105835	True	Wed Oct 11 13:00:09 +0000 2017	@76328 I really hope you all change but I'm su...	119238	
3	119240	VirginTrains	False	Tue Oct 10 15:16:08 +0000 2017	@105836 LiveChat is online at the moment - htt...	119241	
4	119241	105836	True	Tue Oct 10 15:17:21 +0000 2017	@VirginTrains see attached error message. I've...	119243	

```
df["text_lower"] = df["text"].str.lower()
df.head()
```

	text	text_lower
0	@AppleSupport causing the reply to be disregar...	@applesupport causing the reply to be disregar...
1	@105835 Your business means a lot to us. Pleas...	@105835 your business means a lot to us. pleas...
2	@76328 I really hope you all change but I'm su...	@76328 i really hope you all change but i'm su...
3	@105836 LiveChat is online at the moment - htt...	@105836 livechat is online at the moment - htt...
4	@VirginTrains see attached error message. I've...	@virgintrains see attached error message. i've...

```
text = 'I study in the Chinese University of Hong Kong.'
text.lower()
```

```
'i study in the chinese university of hong kong.'
```

Removal of Punctuations

One another common text preprocessing technique is to remove the punctuations from the text data. This is again a text standardization process that will help to treat 'hurray' and 'hurray!' in the same way.

We also need to carefully choose the list of punctuations to exclude depending on the use case. For example, the `string.punctuation` in python contains the following punctuation symbols

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

We can add or remove more punctuations as per our need.

Removal of Punctuations

We need to get a dictionary to store all kinds of punctuations, i.e. string.punctuation.

```
# drop the new column created in last cell
df.drop(["text_lower"], axis=1, inplace=True)

PUNCT_TO_REMOVE = string.punctuation
def remove_punctuation(text):
    """custom function to remove the punctuation"""
    return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))

df["text_wo_punct"] = df["text"].apply(lambda text: remove_punctuation(text))
df.head()
```

Once we meet a punctuation in the input text, we will remove it to empty string.

	text	text_wo_punct
0	@AppleSupport causing the reply to be disregar...	AppleSupport causing the reply to be disregard...
1	@105835 Your business means a lot to us. Pleas...	105835 Your business means a lot to us Please ...
2	@76328 I really hope you all change but I'm su...	76328 I really hope you all change but Im sure...
3	@105836 LiveChat is online at the moment - htt...	105836 LiveChat is online at the moment https...
4	@VirginTrains see attached error message. I've...	VirginTrains see attached error message Ive tr...

Removal of Frequent / Rare words

- stopwords
  - Stopwords are commonly occuring words in a language like 'the', 'a' and so on. They can be removed from the text most of the times, as they don't provide valuable information for downstream analysis. In cases like Part of Speech tagging, we should not remove them as provide very valuable information about the POS.
  - These stopword lists are already compiled for different languages and we can safely use them. For example, the stopword list for english language from the nltk package can be seen below.

```
"i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, yo
u'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, wh
at, which, who, whom, this, that, that'll, these, those, am, is, are, was, wer
```

- Domain specific frequent words
  - But say, if we have a domain specific corpus, we might also have some frequent words which are of not so much importance to us. So we also need to remove other domain-specific frequent words
  - If we use something like tfidf, this is automatically taken care of.

Removal of Frequent / Rare words (Cont)

- Rare word
  - The volume of rare words are too small to affect the text analysis results. In addition, many rare words are just some noiseeeeeeeeeeeee.
  - But sometimes, rare words are important, because they are some unique words for a particular domain.

Removal of Frequent / Rare words

```
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
def remove_stopwords(text):
    """custom function to remove the stopwords"""
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])

df["text_wo_stop"] = df["text_wo_punct"].apply(lambda text: remove_stopwords(text))
df.head()
```

	text	text_wo_punct	text_wo_stop
0	@AppleSupport causing the reply to be disregar...	AppleSupport causing the reply to be disregard...	AppleSupport causing reply disregarded tapped ...
1	@105835 Your business means a lot to us. Pleas...	105835 Your business means a lot to us Please ...	105835 Your business means lot us Please DM na...
2	@76328 I really hope you all change but I'm su...	76328 I really hope you all change but Im sure...	76328 I really hope change Im sure wont Becaus...
3	@105836 LiveChat is online at the moment - htt...	105836 LiveChat is online at the moment https...	105836 LiveChat online moment httpstcoSY94VtU8...
4	@VirginTrains see attached error message. I've...	VirginTrains see attached error message Ive tr...	VirginTrains see attached error message Ive tr...

nltk is a package that defines some stop words for you.

Removal of Frequent / Rare words

```
from collections import Counter
cnt = Counter()
for text in df["text_wo_stop"].values:
    for word in text.split():
        cnt[word] += 1

cnt.most_common(10)
```

```
[('I', 34),
 ('us', 25),
 ('DM', 19),
 ('help', 17),
 ('httpstcoGDrqU22YpT', 12),
 ('AppleSupport', 11),
 ('Thanks', 11),
 ('phone', 9),
 ('Hi', 8),
 ('get', 8)]
```

First, count the frequency for each word in your corpus.

Remove the most/least frequent words according to a threshold “n" (e.g. 10).

```
FREQWORDS = set([w for (w, wc) in cnt.most_common(10)])
def remove_freqwords(text):
    """custom function to remove the frequent words"""
    return " ".join([word for word in str(text).split() if word not in FREQWORDS])

df["text_wo_stopfreq"] = df["text_wo_stop"].apply(lambda text: remove_freqwords(text))
df.head()
```

	text	text_wo_punct	text_wo_stop	text_wo_stopfreq
0	@AppleSupport causing the reply to be disregar...	AppleSupport causing the reply to be disregard...	AppleSupport causing reply disregarded tapped ...	causing reply disregarded tapped notification ...
1	@105835 Your business means a lot to us. Pleas...	105835 Your business means a lot to us Please ...	105835 Your business means lot us Please DM na...	105835 Your business means lot Please name zip...
2	@76328 I really hope you all change but I'm su...	76328 I really hope you all change but Im sure...	76328 I really hope change Im sure wont Becaus...	76328 really hope change Im sure wont Because ...
3	@105836 LiveChat is online at the moment - htt...	105836 LiveChat is online at the moment https...	105836 LiveChat online moment httpstcoSY94VtU8...	105836 LiveChat online moment httpstcoSY94VtU8...
4	@VirginTrains see attached error message. I've...	VirginTrains see attached error message Ive tr...	VirginTrains see attached error message Ive tr...	VirginTrains see attached error message Ive tr...

```
# Drop the two columns which are no more needed
df.drop(["text_wo_punct", "text_wo_stop"], axis=1, inplace=True)

n_rare_words = 10
RAREWORDS = set([w for (w, wc) in cnt.most_common()[:-n_rare_words-1:-1]])
def remove_rarewords(text):
    """custom function to remove the rare words"""
    return " ".join([word for word in str(text).split() if word not in RAREWORDS])

df["text_wo_stopfreqrare"] = df["text_wo_stopfreq"].apply(lambda text: remove_rarewords(text))
df.head()
```

	text	text_wo_stopfreq	text_wo_stopfreqrare
0	@AppleSupport causing the reply to be disregar...	causing reply disregarded tapped notification ...	causing reply disregarded tapped notification ...
1	@105835 Your business means a lot to us. Pleas...	105835 Your business means lot Please name zip...	105835 Your business means lot Please name zip...
2	@76328 I really hope you all change but I'm su...	76328 really hope change Im sure wont Because ...	76328 really hope change Im sure wont Because ...
3	@105836 LiveChat is online at the moment - htt...	105836 LiveChat online moment httpstcoSY94VtU8...	105836 LiveChat online moment httpstcoSY94VtU8...
4	@VirginTrains see attached error message. I've...	VirginTrains see attached error message Ive tr...	VirginTrains see attached error message Ive tr...

Stemming and lemmatization

- For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set.
- The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:
- am, are, is => be
- car, cars, car's, cars' => car
- The result of this mapping of text will be something like:
- the boy's cars are different colors => the boy car be differ color
- However, the two words differ in their flavor. Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma . If confronted with the token saw, stemming might return just s, whereas lemmatization would attempt to return either see or saw depending on whether the use of the token was as a verb or a noun. The two may also differ in that stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma. Linguistic processing for stemming or lemmatization is often done by an additional plug-in component to the indexing process, and a number of such components exist, both commercial and open-source.

Stemming and lemmatization

nltk also provides some quick stem function for you.

Therefore, you do not need to stem every word by yourself.

```
from nltk.stem.porter import PorterStemmer

# Drop the two columns
df.drop(["text_wo_stopfreq", "text_wo_stopfreqrare"], axis=1, inplace=True)

stemmer = PorterStemmer()
def stem_words(text):
    return " ".join([stemmer.stem(word) for word in text.split()])

df["text_stemmed"] = df["text"].apply(lambda text: stem_words(text))
df.head()
```

	text	text_stemmed
0	@AppleSupport causing the reply to be disregar...	@applesupport caus the repli to be disregard a...
1	@105835 Your business means a lot to us. Pleas...	@105835 your busi mean a lot to us. pleas dm y...
2	@76328 I really hope you all change but I'm su...	@76328 i realli hope you all chang but i'm sur...
3	@105836 LiveChat is online at the moment - htt...	@105836 livechat is onlin at the moment - http...
4	@VirginTrains see attached error message. I've...	@virgintrain see attach error message. i'v tri...

Stemming and lemmatization

Lemmatization requires additional corpus called “wordnet” in nltk.

As a result, Lemmatization is generally slower than stemming process. So depending on the speed requirement, we can choose to use either stemming or lemmatization.

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
def lemmatize_words(text):
    return " ".join([lemmatizer.lemmatize(word) for word in text.split()])

df["text_lemmatized"] = df["text"].apply(lambda text: lemmatize_words(text))
df.head()
```

```
[nltk_data] Downloading package wordnet to /Users/weiwen/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

	text	text_stemmed	text_lemmatized
0	@AppleSupport causing the reply to be disregar...	@applesupport caus the repli to be disregard a...	@AppleSupport causing the reply to be disregar...
1	@105835 Your business means a lot to us. Pleas...	@105835 your busi mean a lot to us. pleas dm y...	@105835 Your business mean a lot to us. Please...
2	@76328 I really hope you all change but I'm su...	@76328 i realli hope you all chang but i'm sur...	@76328 I really hope you all change but I'm su...
3	@105836 LiveChat is online at the moment - htt...	@105836 livechat is onlin at the moment - http...	@105836 LiveChat is online at the moment - htt...
4	@VirginTrains see attached error message. I've...	@virgintrain see attach error message. i'v tri...	@VirginTrains see attached error message. I've...

Stemming and lemmatization

Advanced lemmatization according to word's Part-of-Speech (POS) label.

```
print("Word is : stripes")
print("Lemma result for verb : ",lemmatizer.lemmatize("stripes", 'v'))
print("Lemma result for noun : ",lemmatizer.lemmatize("stripes", 'n'))
```

```
Word is : stripes
Lemma result for verb :  strip
Lemma result for noun :  stripe
```

Removal of Emojis / Emoticons

With more and more usage of social media platforms, there is an explosion in the usage of emojis and emoticons in our day to day life as well. Probably we might need to remove these emojis for some of our textual analysis.

There is a minor difference between emojis and emoticons.

From Grammarist.com, emoticon is built from keyboard characters that when put together in a certain way represent a facial expression, an emoji is an actual image.

:-) is an emoticon

 is an emoji

Please note again that the removal of emojis / emoticons are not always preferred and decision should be made based on the use case at hand.

Removal of Emojis / Emoticons

```
# Reference : https://gist.github.com/slowkow/7a7f61f495e3dbb7e3d767f97bd7304b
def remove_emoji(string):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', string)

remove_emoji("game is on 🔥🔥")
```

```
'game is on '
```

In emojis removal, we use a useful python package “re” (regular expression).

“re” allows us to find all sub-strings match the query regular expression.

In emoticons removal, emoticons need to be manually defined.

```
# Reference: https://github.com/NeelShah18/emot/blob/master/emot/emo_unicode.py
EMOTICONS = {
    u":-D":"Laughing, big grin or laugh with glasses",
    u":D":"Laughing, big grin or laugh with glasses",
    u"8-D":"Laughing, big grin or laugh with glasses",
    u"8D":"Laughing, big grin or laugh with glasses",
    u"X-D":"Laughing, big grin or laugh with glasses",
    u"XD":"Laughing, big grin or laugh with glasses",
    u"=D":"Laughing, big grin or laugh with glasses",
    u"=3":"Laughing, big grin or laugh with glasses",
    u"B^D":"Laughing, big grin or laugh with glasses",}
def remove_emoticons(text):
    emoticon_pattern = re.compile(u'(' + u'|'.join(k for k in EMOTICONS) + u')')
    return emoticon_pattern.sub(r'', text)

remove_emoticons("Hello :-D; Hello 8D; Hello =3")
```

```
'Hello ; Hello ; Hello '
```

Removal of URLs / HTML tags

Next preprocessing step is to remove any URLs present in the data. For example, if we are doing a twitter analysis, then there is a good chance that the tweet will have some URL in it. Probably we might need to remove them for our further analysis.

One another common preprocessing technique that will come handy in multiple places is removal of html tags. This is especially useful, if we scrap the data from different websites. We might end up having html strings as part of our text.

Removal of URLs / HTML tags

We use regular expression to detect all urls in the input text. And then remove all of them.

```
def remove_urls(text):
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(r'', text)

text = "Driverless AI NLP blog post on https://www.h2o.ai/blog/detecting-sarcasm-is-difficult"
remove_urls(text)
```

```
'Driverless AI NLP blog post on '
```

An alternative is to replace url to a special token called '[url]' such that the sentence is still grammatically correct.

```
def remove_urls(text):
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(r'[url]', text)

text = "Driverless AI NLP blog post on https://www.h2o.ai/blog/detecting-sarcasm-is-difficult"
remove_urls(text)
```

```
'Driverless AI NLP blog post on [url]'
```

Removal of URLs / HTML tags

Similarly, we use regular expression to remove all HTML tags

```
def remove_html(text):
    html_pattern = re.compile('<.*?>')
    return html_pattern.sub(r'', text)

text = """<div>
<h1> H2O</h1>
<p> AutoML</p>
<a href="https://www.h2o.ai/products/h2o-driverless-ai/"> Driverless AI</a>
</div>"""

print(remove_html(text))
```

```
 H2O
 AutoML
 Driverless AI
```

Conversion between Full / Half width

In CJK (Chinese, Japanese and Korean) computing, graphic characters are traditionally classed into **fullwidth** (in Taiwan and Hong Kong: 全形; in CJK: 全角) and **halfwidth** (in Taiwan and Hong Kong: 半形; in CJK: 半角) characters. Unlike monospaced fonts, a halfwidth character occupies half the width of a fullwidth character, hence the name.

Halfwidth and Fullwidth Forms is also the name of a Unicode block U+FF00–FFEF, provided so that older encodings containing both halfwidth and fullwidth characters can have lossless translation to/from Unicode.

９月１６日

9月16日

Conversion between Full / Half width

ord() would return the ascII or unicode of a character.

```python
def q_to_b(q_str):
    """全角转半角"""
    b_str = ""
    for uchar in q_str:
        inside_code = ord(uchar)
        if inside_code == 12288:  # 全角空格直接转换
            inside_code = 32
        elif 65374 >= inside_code >= 65281:  # 全角字符（除空格）根据关系转化
            inside_code -= 65248
        b_str += chr(inside_code)
    return b_str


def b_to_q(b_str):
    """半角转全角"""
    q_str = ""
    for uchar in b_str:
        inside_code = ord(uchar)
        if inside_code == 32:  # 半角空格直接转化
            inside_code = 12288
        elif 126 >= inside_code >= 32:  # 半角字符（除空格）根据关系转化
            inside_code += 65248
        q_str += chr(inside_code)
    return q_str

q_to_b("９月１６日")
```

```
'9月16日'
```

Conversion between Traditional / Simplified words

Traditional Chinese characters remain in common use in Taiwan, Hong Kong and Macau, as well as in most overseas Chinese communities outside Southeast Asia.

Simplified Chinese is widely used in mainland China and Singapore.

We do not want an input text to have both traditional and simplified Chinese words, because it is hard for human readers to understand and it would also create a duplicated mention of the same word.

Conversion between Traditional / Simplified words

You will need a dictionary to record the traditional and simplified version of the words.

```
def tra2sim(text):
    tra2sim_dict = {
        '個':"个",
        "國":'国',
        "際":"际",
    }
    for char in text:
        if char in tra2sim_dict:
            text = text.replace(char, tra2sim_dict[char])
    return text

text = '香港是一個國際化的大都市'
tra2sim(text)
```

'香港是一个国际化的大都市'

Chinese Word Segmentation

It's well known that Chinese is an ideographic language and there is no word delimiter between words in written Chinese sentences. Chinese word segmentation is to segment a Chinese sentence into some segment, where each segment is a semantic unit.

Word segmentation becomes the very first task when processing Chinese text and in turn the accuracy of word segmentation is essential to the performance of the following procedures.

小明硕士毕业于中国科学院计算所，后在日本京都大学深造

小明/ 硕士/ 毕业/ 于/ 中国科学院/ 计算所/ ，/ 后/ 在/ 日本京都大学/ 深造

小明硕士/ 毕业/ 于/ 中国科/ 学院/ 计算/ 所/ ，/ 后/ 在/ 日本京都大学/ 深造

Chinese Word Segmentation

jieba（结巴）is the most famous Chinese word segmentation tool. Here we use jibe for demonstration purpose.

Other tools like thulac (made by Tsinghua), spacy (made by Stanford) are also very good.

jieba is not included in Anaconda, you have to manually install this library.

```
# encoding=utf-8
import jieba

seg_list = jieba.cut("我来到北京清华大学", cut_all=False)
print("/ ".join(seg_list))  # 精确模式

seg_list = jieba.cut("他来到了网易杭研大厦")  # 默认是精确模式
print("/ ".join(seg_list))

seg_list = jieba.cut("小明硕士毕业于中国科学院计算所，后在日本京都大学深造")  # 精确模式
print("/ ".join(seg_list))

seg_list = jieba.cut_for_search("小明硕士毕业于中国科学院计算所，后在日本京都大学深造")  # 搜索引擎模式
print("/ ".join(seg_list))
```

```
Building prefix dict from the default dictionary ...
Loading model from cache /var/folders/74/zmwvkvzx4v7d_hlr3fqg7x_00000gn/T/jieba.cache
Loading model cost 0.449 seconds.
Prefix dict has been built successfully.
```

```
我/ 来到/ 北京/ 清华大学
他/ 来到/ 了/ 网易/ 杭研/ 大厦
小明/ 硕士/ 毕业/ 于/ 中国科学院/ 计算所/ ，/ 后/ 在/ 日本京都大学/ 深造
小明/ 硕士/ 毕业/ 于/ 中国/ 科学/ 学院/ 科学院/ 中国科学院/ 计算/ 计算所/ ，/ 后/ 在/ 日本/ 京都/ 大学/ 日本京都大学/ 深造
```

Text Classification with Python

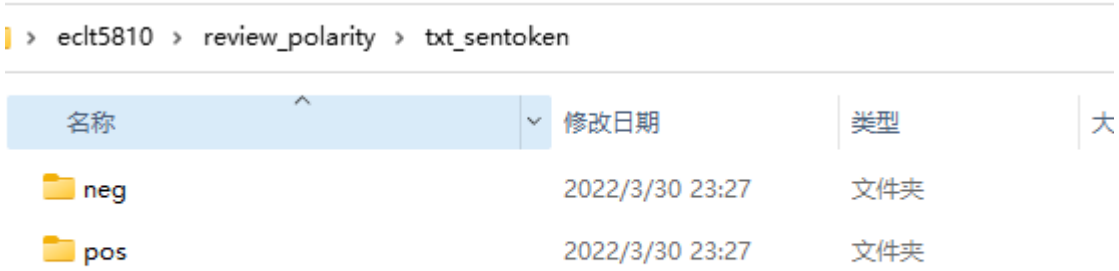
Yang Haoran
2022-4-1

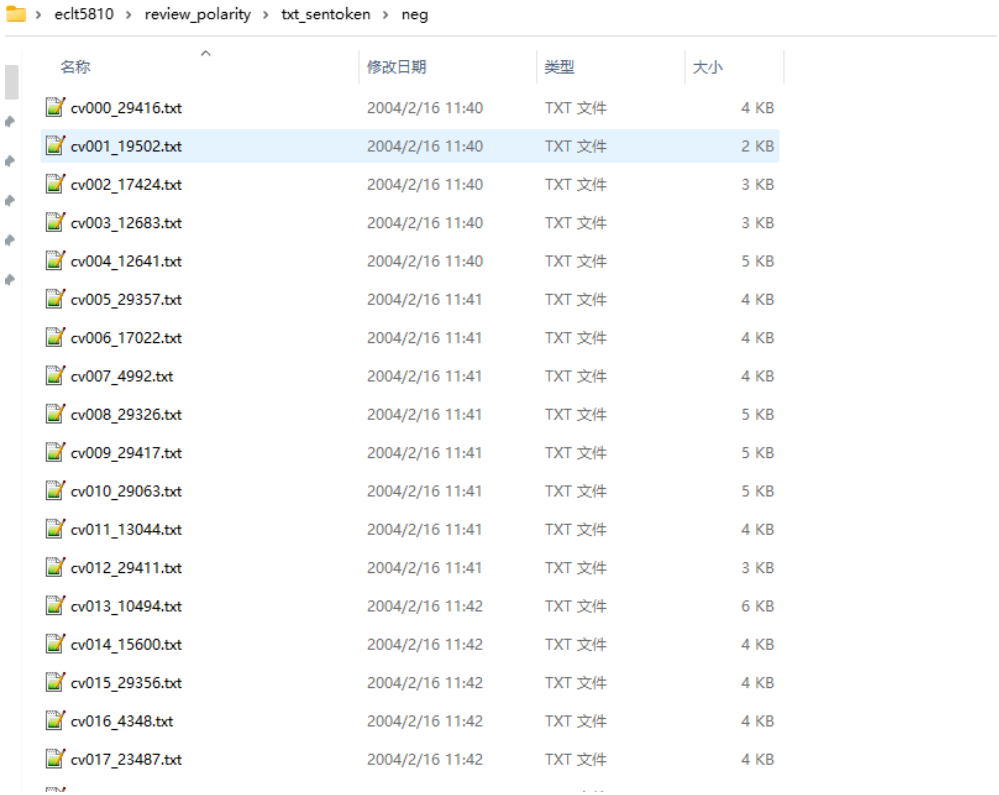
Overview

- Movie Review Sentiment Classification
  - train a classification model capable of predicting whether a given movie review is positive or negative.
- Dataset
  - Moview-review data collected by Cornell University
- Text Preprocessing
  - Mainly using NLTK and regular expression
  - Converting Text to Numbers
- Model training and Evaluation
  - Naïve bayes, logistic regression…
  - Confusion matrix…

Dataset

You can download the sentiment classification dataset from the below Link or course website
www.cs.cornell.edu/people/pabo/movie-review-data/review_polarity.tar.gz





Contains 1000 positive reviews and 1000 negative reviews.

Dataset

cv008_29326.txt

```
call it a road trip for the walking wounded .
stellan skarsg ? rd plays such a convincingly zombified drunken loser that it's difficult to spend nearly two hours of screen
yet this ever-reliable swedish actor adds depth and significance to the otherwise plodding and forgettable aberdeen , a sentim
playwright august strindberg built his career on families and relationships paralyzed by secrets , unable to express their lon
that's an accurate reflection of what aberdeen strives for , focusing on the pairing of an alcoholic father , tomas ( skarsg ?
they haven't spoken in years , and wouldn't even be making the long trip from norway to aberdeen , scotland by automobile if i
in a soap opera twist , mother has only a few days to live .
 ( only in the movies , right ? )
too blitzed to even step foot on a plane , tomas hits the open road with kaisa .
loathing each other all the while , they make periodic stops for tomas to puke on the dashboard or pass out -- whenever he isn
despite his sloshed viewpoint , tomas recognizes that the apple hasn't fallen very far from the tree .
kaisa gets nosebleeds from snorting coke , sabotages her personal relationships through indifference , and is unable to restra
ain't they a pair ?
unable to find true notes of unspoken familial empathy in the one-note and repetitively bitchy dialogue , screenwriters kristi
by the time they reach the hospital , it's time to unveil the secrets from a dark past that are not only simplistic devices th
this revelation exists purely for its own sake .
aberdeen doesn't know where else to go .
weak , unimaginative casting thwarts the pivotal role of kaisa .
if lena headey were a stronger actress , perhaps aberdeen could have been able to coast on the performances and moody , haunti
headey's too busy acting , using her face and furrowed brow to convey every last twitch of insouciance .
if she were paying any attention to skarsg ? rd , maybe she'd figure out that doing less can reveal so much more .
it's worthwhile to compare aberdeen to an earlier film released in 2001 , jonathan nossiter's captivating signs & wonders .
it's not just because skarsg ? rd and rampling played disturbed parental figures in both films ( they're not bound by ceremoni
the differences in the way their characters were presented is significant .
in aberdeen , rampling is a luminous diva , preening and static in her hospital bed .
despite skarsg ? rd's solid performance as tomas , his pathetic drunk is never given much of a chance to emote anything beside
there's genuine ferocity and sexually charged frisson during their understated confrontations in signs & wonders , allowing th
nossiter's film thoroughly explores this neurotic territory in addition to delving into the americanization of greece and the
if signs & wonders sometimes feels overloaded with ideas , at least it's willing to stretch beyond what we've come to expect f
aberdeen is never half so ambitious , content to sleepwalk through the rhythms and timing of other movies .
when did character driven stories stop paying attention to the complexities of real life ?
the depressing answer can be found in lawrence kasdan's trite but occasionally useful grand canyon , where steve martin's holl
even foreign films are taking that advice to heart .
```

Text Preprocessing

A **regular expression** is a sequence of characters that specifies a *search pattern* in text. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation.

You can also convert the original dataset to pandas' format, so that you can use pandas to process it.

```
for sen in range(0, len(X)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(X[sen]))

    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', '', document)

    # Converting to Lowercase
    document = document.lower()

    # Lemmatization
    document = document.split()

    document = [stemmer.lemmatize(word) for word in document]
    document = ' '.join(document)

    documents.append(document)
```

r+'pattern'

\s: match A whitespace character

\s+: match one or more ….

[a-zA-Z]: match ASCII characters in the range from A to Z and in the range from a to z

^[a-zA-Z]: begin with char in a-z or A-Z

Converting Text to Numbers

We use tf-idf to represent a document.

tf: term frequency

Idf: inverse document frequency

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \left(1 + \log\left(\mathrm{tf}_{t,d}\right)\right) \times \log\left({}^{N}/_{\mathrm{df}_t}\right)$$

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidfconverter = TfidfVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
X = tfidfconverter.fit_transform(documents).toarray()
```

max_features: maximum words we use. (Not using whole vocabulary since it may be very large)
we want to use 1500 most occurring words as features for training our classifier.
min_df: the minimum number of documents that should contain this feature. So we only include those words that occur in at least 5 documents.
max_df: we should include only those words that occur in a maximum of 70% of all the documents. Words that occur in almost every document are usually not suitable for classification because they do not provide any unique information about the document.

Training and Evaluation

Split the whole dataset to train and test set. The split fraction is 0.2.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)
```

```
y_pred = classifier.predict(X_test)
```

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```