Data Processing Using Python

Data statistics, mining and application

ZHANG Li/Dazhuang
Nanjing University
Department of Computer Science and Technology
Department of University Basic Computer Teaching

Data Processing Using Python

1

BASIC DATA CHARACTERISTICS ANALYSIS OF DATA EXPLORATION

Data Exploration

Data Exploration

check data errors

understand data distribution

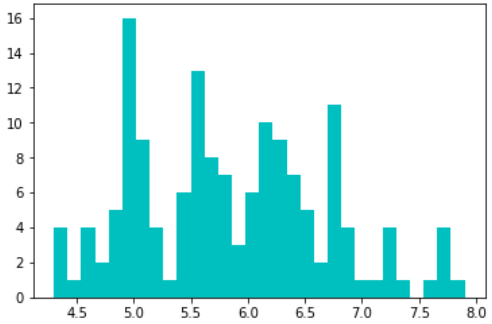
characteristics and inherent regularities

Detect Data Errors

- **Missing values**
- **Outliers**
- **Inconsistent data**
  - 149 2 -> 149 2.0
  - 1569936600 -> 2019-10-01

Basic Data Characteristics Analysis Method

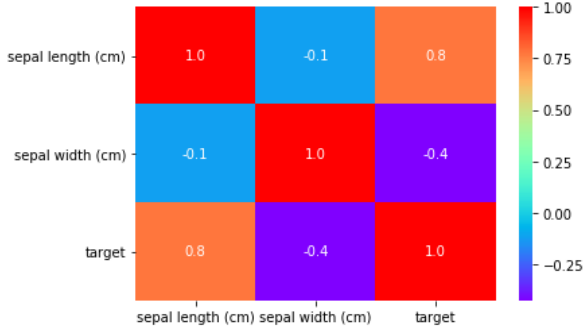
Distribution Analysis



Statistical Analysis

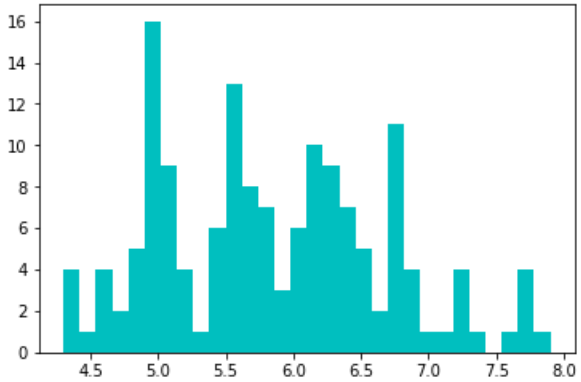
```
iris_df.iloc[:,0].describe()
count    150.000000
mean       5.843333
std        0.828066
min        4.300000
25%        5.100000
50%        5.800000
75%        6.400000
max        7.900000
Name: sepal length (cm), dtype: float64
```

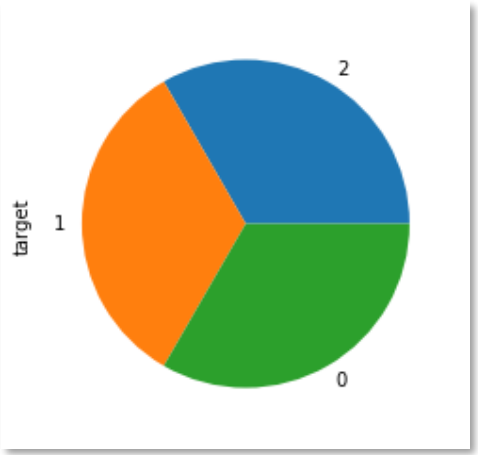
Correlation Analysis



Distribution Analysis

- Distribution Analysis
  - quantitative data
  - qualitative data





Quantitative Data Distribution Analysis

histogram

Source

```
>>> plt.hist(iris_df.iloc[:,0], 5, color = 'c')
```

normal distribution test

Source

```
>>> scipy.stats.normaltest(iris_df.iloc[:,0])
```

Qualitative Data Distribution Analysis

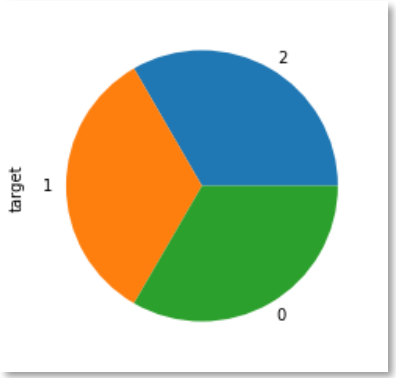
Source

```
>>> iris_df.target.value_counts()
```

```
2    50
1    50
0    50
Name: target, dtype: int64
```

Source

```
>>> iris_df.target.value_counts().plot(kind = 'pie')
```



Statistical Analysis

- Statistical Analysis
  - Central tendency analysis

    mean value, median
  - Dispersion tendency analysis

    standard deviation, IQR(the difference between the upper quartile and the lower quartile)

Statistical Analysis

- Statistical Analysis
  - Central tendency analysis:

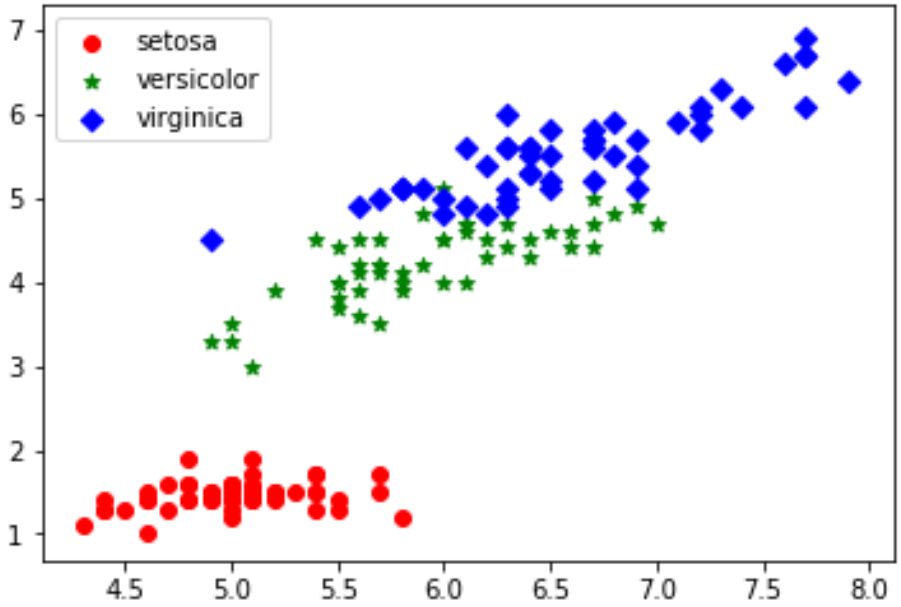
    mean(), median()
  - Dispersion tendency analysis

    std(), quantile()

```
iris_df.iloc[:,0].describe()
count    150.000000
mean       5.843333
std        0.828066
min        4.300000
25%        5.100000
50%        5.800000
75%        6.400000
max        7.900000
Name: sepal length (cm), dtype: float64
```

Correlation Analysis

- Common way
  - single graph
  - graph matrix
  - correlation coefficient



correlation coefficient - Pearson

$$r_{xy} = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{\left(\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2}\right)\left(\sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}\right)}$$

Constraints:

1. Linear relationship between two variables.
2. All variables are continuous.
3. The variables are all in normal distribution, and the binary distribution is also in normal distribution.
4. Two variables are independent.

correlation coefficient - Pearson

r:

positive correlation: r > 0

negative correlation: r < 0

unrelated: |r| = 0

complete linear correlation: |r| = 1

	sepal length (cm)	sepal width (cm)	target
sepal length (cm)	1.000000	-0.117570	0.782561
sepal width (cm)	-0.117570	1.000000	-0.426658
target	0.782561	-0.426658	1.000000

Data Processing Using Python

2

DATA STATISTICS AND ANALYSIS BASED ON PANDAS

Data Display

	code	name	price
0	MMM	3M	155.82
1	AXP	American Express	114.41
2	AAPL	Apple	227.01
3	BA	Boeing	375.70
4	CAT	Caterpillar	121.04
5	CVX	Chevron	113.85
6	CSCO	Cisco	47.52
7	KO	Coca-Cola	54.54
8	DIS	Disney	130.27
9	DOW	Dow Chemical	45.34
10	XOM	Exxon Mobil	68.97
11	GS	Goldman Sachs	200.80
12	HD	Home Depot	227.93
13	IBM	IBM	142.99
14	INTC	Intel	50.92
15	JNJ	Johnson & Johnson	133.66
16	JPM	JPMorgan Chase	114.62
17	MCD	McDonald's	211.69
18	MRK	Merck	85.00
19	MSFT	Microsoft	138.12
20	NKE	Nike	93.07
21	PFE	Pfizer	35.93
22	PG	Procter & Gamble	124.00
23	TRV	Travelers Companies Inc	144.96
24	UTX	United Technologies	133.21
25	UNH	UnitedHealth	219.80
26	VZ	Verizon	59.90
27	V	Visa	175.98
28	WMT	Wal-Mart	118.16
29	WBA	Walgreen	52.97

djidf

	close	high	low	open	volume
2018-10-19	106.730003	107.550003	104.059998	104.059998	5726300
2018-10-22	104.510002	106.959999	104.449997	106.610001	5003100
2018-10-23	104.379997	104.519997	101.839996	102.410004	4223800
2018-10-24	101.839996	104.949997	101.510002	104.430000	4056700
2018-10-25	103.599998	104.169998	101.800003	102.480003	3378900
2018-10-26	101.250000	102.660004	100.139999	102.540001	5395700
2018-10-29	101.190002	103.250000	100.040001	102.470001	4238700
2018-10-30	102.080002	102.389999	100.410004	101.599998	3778200
2018-10-31	102.730003	103.709999	102.550003	103.059998	4511300
2018-11-01	104.040001	104.269997	103.019997	103.260002	2786800
2018-11-02	103.709999	105.050003	102.889999	104.930000	4322200
2018-11-05	105.209999	105.400002	103.800003	104.040001	2697700
2018-11-06	104.980003	105.660004	104.370003	104.980003	2856000
2018-11-07	107.309998	107.480003	104.900002	105.730003	3606900
2018-11-08	108.500000	108.629997	107.029999	107.029999	2896700
2018-11-09	108.279999	109.330002	107.349998	108.379997	4444000
2018-11-12	106.489998	108.440002	106.300003	108.160004	3154600
2018-11-13	107.860001	108.199997	106.470001	106.650002	3021800
2018-11-14	107.769997	109.330002	106.889999	108.610001	4978100
2018-11-15	109.599998	109.699997	106.339996	106.680000	3742600

quotesdf

Basic Filtering and Statistics

1. Compute the average of last trade price for all DJI constituents.
2. Select the name of all companies whose last trade price are greater than or equal to 300.

Source

```
>>> djidf.price.mean()
133.148
>>> djidf[djidf.price >= 300].name
3    Boeing
Name: name, dtype: object
```

Basic Statistics and Filtering

- Seek the company information that the latest trade price in the DJI constituent stocks is more than or equal to 300 or less than or equal to 50.
- Count the opening days of AXP in September 2019.

Source

```
>>> djidf[(djidf.price >= 300) | (djidf.price <= 50)]
     code              name      price
3      BA            Boeing   372.31
6    CSCO             Cisco    47.01
9     DOW    Dow Chemical     48.13
21    PFE             Pfizer    36.65
>>> t = quotesdf[(quotesdf.index >= '2019-09-01')  &  (quotesdf.index <= '2019-09-30')]
>>> len(t)
20
```

Basic Statistics and Filtering

- Compute the total number of rise and fall days in form.

Source

```
>>> len(quotesdf[quotesdf.close > quotesdf.open])
130
>>> len(quotesdf)-130
122
```

- Compute the number of rise and fall days according to close price of every adjacent pair of days.

Source

```
>>> status = np.sign(np.diff(quotesdf.close))
>>> status
array([ -1.,  1., -1., ..., 1.,  -1.,  1.])
>>> len(status[status==1])
131
>>> len(status[status==-1])
118
```

Basic Statistics and Filtering

- Sort DJI constituent stocks according to the last trade price, and choose the first three companies.

Source

```
>>> tempdf = djidf.sort_values(by = 'price', ascending = False)
>>> tempdf
    code              name      price
3     BA             Boeing     372.31
25   UNH      UnitedHealth     238.50
...
6    CSCO             Cisco      47.01
21    PFE             Pfizer      36.65
>>> tempdf[:3].name
3              Boeing
25     UnitedHealth
12      Home Depot
Name: name, dtype: object
```

Grouping

- Calculate the number of opening days of AXP for each month in the past year.

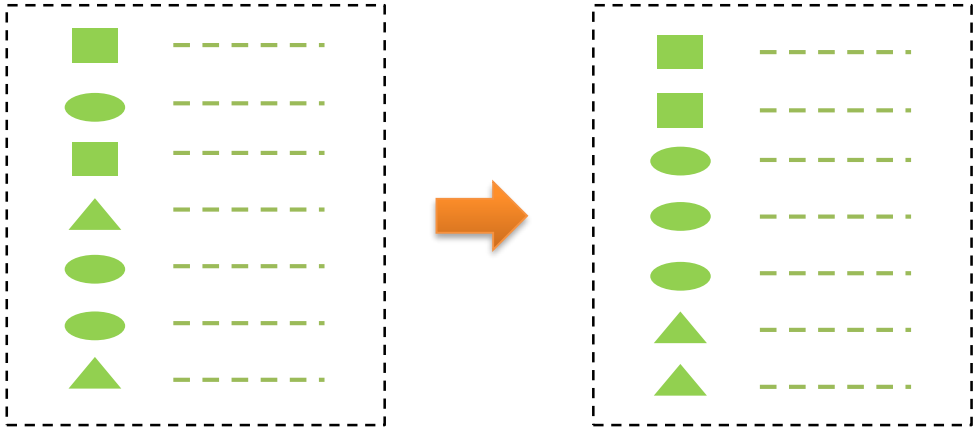
Source

```
>>> month = [item[5:7] for item in quotesdf.index]
>>> x = quotesdf.groupby(month).open.count()
```

Output:

```
01    21
02    19
03    21
04    21
05    22
06    20
07    22
08    22
09    20
10    23
11    21
12    19
Name: close, dtype: int64
```

Grouping



Order of Grouping

① Splitting

② Applying

③ Combining

groupby()与apply()

- Calculate the number of opening days of AXP for each month in the past year.

Source

```
>>> month = [item[5:7] for item in quotesdf.index]
>>> quotesdf.groupby(month).apply(len)
```

Output:

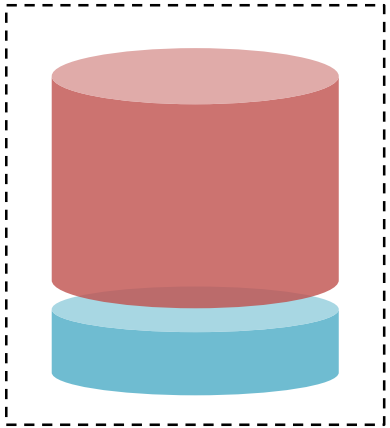
```
01    21
02    19
03    21
04    21
05    22
06    20
07    22
08    22
09    20
10    23
11    21
12    19
Name: close, dtype: int64
```

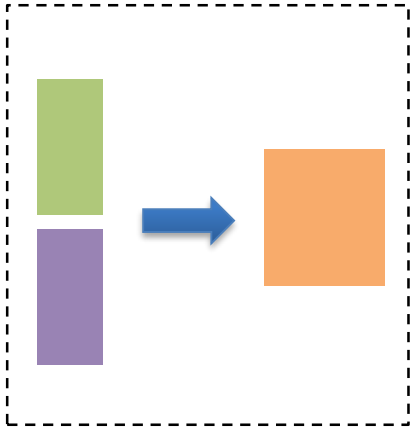
apply()

Source

```
>>> quotesdf.max()
>>> quotesdf.max(axis = 1)
>>> quotesdf.loc[:, ['close', 'open']].astype(int)
>>> quotesdf.apply(max)
>>> quotesdf.apply(max, axis = 1)
>>> quotesdf.loc[:, ['close', 'open']].apply(np.int32)
>>> quotesdf.loc[:, ['close', 'open']].apply(int)
>>> quotesdf.loc[:, ['close', 'open']].applymap(int)
>>> quotesdf.volume.apply(float)  # apply() also can be applied on every element of a Series object
>>> quotesdf.loc[:, ['close', 'open']]= quotesdf.loc[:, ['close', 'open']].apply(np.int32)
```

Merge





Form of Merge

- Append
  - add additional row into DataFrame
- Concat
  - Connect pandas objects
- Join
  - Connect SQL type

Append

- Merge the trade information between September 1st and 5th of year 2019 into the record of the last two days of June.

Source

```
>>> p = quotesdf['2019-06-01':'2019-06-30'][-2:]
>>> p
                close        high         low        open   volume
2019-06-27 123.940002  124.410004  123.559998  123.690002  1504300
2019-06-28 123.440002  124.550003  123.199997  124.290001  4338900
>>> q = quotesdf['2019-09-01':'2019-09-05']
>>> q
                close        high         low        open   volume
2019-09-03 117.599998  120.279999  117.519997  119.860001  3198700
2019-09-04 118.400002  118.730003  117.800003  118.419998  3746900
2019-09-05 120.669998  121.629997  119.500000  119.500000  5261100
>>> p.append(q)
                close        high         low        open   volume
2019-06-27 123.940002  124.410004  123.559998  123.690002  1504300
2019-06-28 123.440002  124.550003  123.199997  124.290001  4338900
2019-09-03 117.599998  120.279999  117.519997  119.860001  3198700
2019-09-04 118.400002  118.730003  117.800003  118.419998  3746900
2019-09-05 120.669998  121.629997  119.500000  119.500000  5261100
```

Concat

- Merge the first 3 and the last 3 records of AXP stock data in September 2019.

Source

```
>>> tempdf = quotesdf[(quotesdf.index >= '2019-09-01') & (quotesdf.index <= '2019-09-30')]
>>> pieces = [tempdf[:3], tempdf[-3:]]
>>> pd.concat(pieces)
                 close      high       low      open  volume
2019-09-03  117.599998  120.279999  117.519997  119.860001  3198700
2019-09-04  118.400002  118.730003  117.800003  118.419998  3746900
2019-09-05  120.669998  121.629997  119.500000  119.500000  5261100
2019-09-26  118.910004  120.029999  118.839996  119.309998  3411200
2019-09-27  118.589996  119.620003  117.980003  119.129997  2909200
2019-09-30  118.279999  119.239998  118.139999  119.050003  2345800
```

Join

code	name
AXP	
KO	

volume	code	month
	AXP	
	AXP	
	KO	
	KO	

code	name	volume	month
AXP			
AXP			
KO			
KO			

Join

- Merge the monthly mean volume result of American Express and Coca-Cola with DJI constituents information.

code|name|volume|month

	code	name	price
0	MMM	3M	155.82
1	AXP	American Express	114.41
2	AAPL	Apple	227.01
3	BA	Boeing	375.70
4	CAT	Caterpillar	121.04
5	CVX	Chevron	113.85
6	CSCO	Cisco	47.52
7	KO	Coca-Cola	54.54
8	DIS	Disney	130.27
9	DOW	Dow Chemical	45.34
10	XOM	Exxon Mobil	68.97
11	GS	Goldman Sachs	200.80
12	HD	Home Depot	227.93
13	IBM	IBM	142.99
14	INTC	Intel	50.92
15	JNJ	Johnson & Johnson	133.66
16	JPM	JPMorgan Chase	114.62
17	MCD	McDonald's	211.69
18	MRK	Merck	85.00
19	MSFT	Microsoft	138.12
20	NKE	Nike	93.07
21	PFE	Pfizer	35.93
22	PG	Procter & Gamble	124.00
23	TRV	Travelers Companies Inc	144.96
24	UTX	United Technologies	133.21
25	UNH	UnitedHealth	219.80
26	VZ	Verizon	59.90
27	V	Visa	175.98
28	WMT	Wal-Mart	118.16
29	WBA	Walgreen	52.97

djidf

	volume	code	month
01	4.216338e+06	AXP	01
02	3.096963e+06	AXP	02
03	3.366676e+06	AXP	03
04	3.439100e+06	AXP	04
05	3.163909e+06	AXP	05
06	2.856915e+06	AXP	06
07	3.324177e+06	AXP	07
08	3.291932e+06	AXP	08
09	3.936210e+06	AXP	09
10	3.572539e+06	AXP	10
11	3.437376e+06	AXP	11
12	5.076395e+06	AXP	12
01	1.384537e+07	KO	01
02	2.045106e+07	KO	02
03	1.733149e+07	KO	03
04	1.133406e+07	KO	04
05	1.173954e+07	KO	05
06	1.242934e+07	KO	06
07	1.171300e+07	KO	07
08	1.235928e+07	KO	08
09	1.082578e+07	KO	09
10	1.389021e+07	KO	10
11	1.333130e+07	KO	11
12	1.621624e+07	KO	12

AKdf

Join

Source

```
>>> pd.merge(djidf.drop(['price'], axis = 1), AKdf, on = 'code')
```

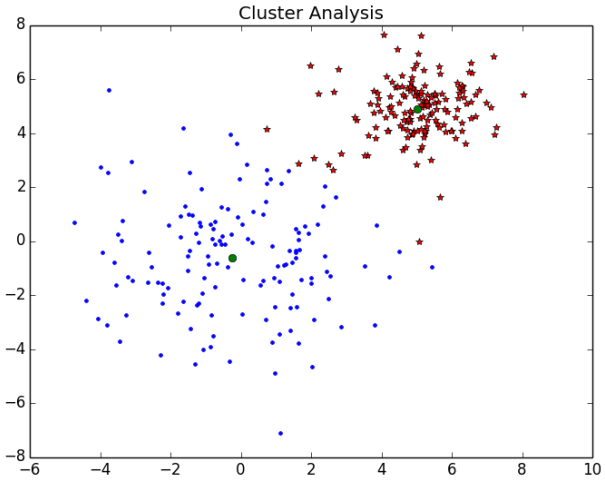
	code	name	volume	month
0	AXP	American Express	4.216338e+06	01
1	AXP	American Express	3.096963e+06	02
2	AXP	American Express	3.366676e+06	03
…				
11	AXP	American Express	5.076395e+06	12
12	KO	Coca-Cola	1.384537e+07	01
13	KO	Coca-Cola	2.045106e+07	02
…				
21	KO	Coca-Cola	1.389021e+07	10
22	KO	Coca-Cola	1.333130e+07	11
23	KO	Coca-Cola	1.621624e+07	12

	code	name	volume	month
0	AXP	American Express	4.216338e+06	01
1	AXP	American Express	3.096963e+06	02
2	AXP	American Express	3.366676e+06	03
3	AXP	American Express	3.439100e+06	04
4	AXP	American Express	3.163909e+06	05
5	AXP	American Express	2.856915e+06	06
6	AXP	American Express	3.324177e+06	07
7	AXP	American Express	3.291932e+06	08
8	AXP	American Express	3.936210e+06	09
9	AXP	American Express	3.572539e+06	10
10	AXP	American Express	3.437376e+06	11
11	AXP	American Express	5.076395e+06	12
12	KO	Coca-Cola	1.384537e+07	01
13	KO	Coca-Cola	2.045106e+07	02
14	KO	Coca-Cola	1.733149e+07	03
15	KO	Coca-Cola	1.133406e+07	04
16	KO	Coca-Cola	1.173954e+07	05
17	KO	Coca-Cola	1.242934e+07	06
18	KO	Coca-Cola	1.171300e+07	07
19	KO	Coca-Cola	1.235928e+07	08
20	KO	Coca-Cola	1.082578e+07	09
21	KO	Coca-Cola	1.389021e+07	10
22	KO	Coca-Cola	1.333130e+07	11
23	KO	Coca-Cola	1.621624e+07	12

Data Processing Using Python

CLUSTER ANALYSIS

Cluster



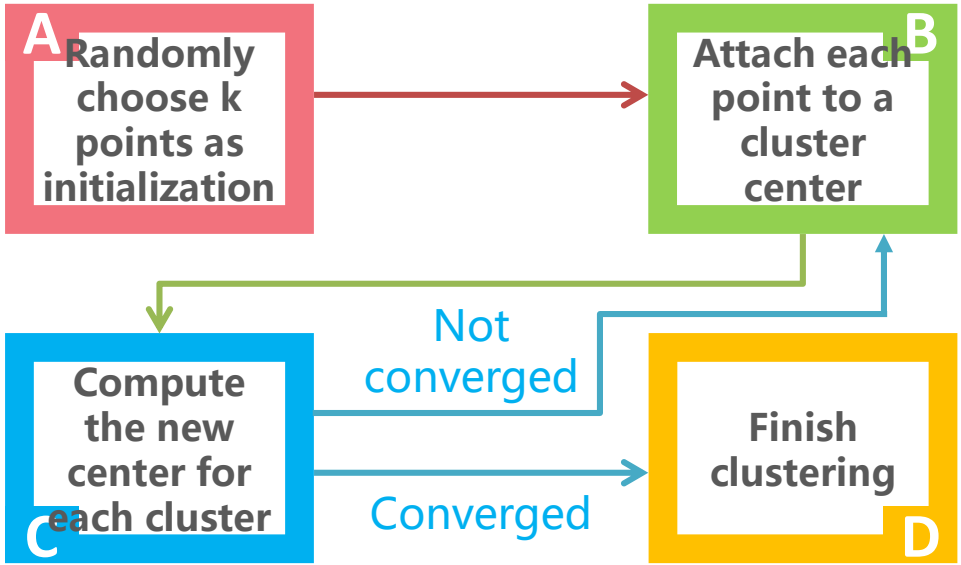
- **cluster analysis**

grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

  - Feature
    - Based on similarity
    - Have multiple cluster centers

K-MEANS

K-means algorithm uses k points in space as the centers to cluster all objects.



A Daily Example

	Math	English	Python	Music
ming	88	64	96	85
MING	92	99	95	94
peng	91	87	99	95
PENG	78	99	97	81
meng	88	78	98	84
MENG	100	95	100	92

Output:

[1 0 0 1 1 0]

File

```
# Filename: kmeansStu1.py
import numpy as np
from scipy.cluster.vq import vq, kmeans, whiten
list1 = [88.0, 74.0, 96.0, 85.0]
list2 = [92.0, 99.0, 95.0, 94.0]
list3 = [91.0, 87.0, 99.0, 95.0]
list4 = [78.0, 99.0, 97.0, 81.0]
list5 = [88.0, 78.0, 98.0, 84.0]
list6 = [100.0, 95.0, 100.0, 92.0]
data = np.array([list1,list2,list3,list4,list5,list6])
whiten = whiten(data)
centroids,_ = kmeans(whiten, 2)
result,_= vq(whiten, centroids)
print(result)
```

Solve with Specific Tools

File

```
# Filename: kmeansStu2.py
import numpy as np
from sklearn.cluster import KMeans
list1 = [88.0,74.0,96.0,85.0]
list2 = [92.0,99.0,95.0,94.0]
list3 = [91.0,87.0,99.0,95.0]
list4 = [78.0,99.0,97.0,81.0]
list5 = [88.0,78.0,98.0,84.0]
list6 = [100.0,95.0,100.0,92.0]
X = np.array([list1,list2,list3,list4,list5,list6])
kmeans = KMeans(n_clusters = 2).fit(X)
pred = kmeans.predict(X)
print(pred)
```

scikit learn

```
from sklearn import datasets
from sklearn import svm
clf = svm.SVC(gamma=0.001, C=100.)
digits = datasets.load_digits()
clf.fit(digits.data[:-1], digits.target[:-1])
clf.predict(digits.data[-1])
```

Output:

[0 1 1 1 0 1]

Another Example

- Cluster 10 DJI constituents according to close price trend of every adjacent pair of days.

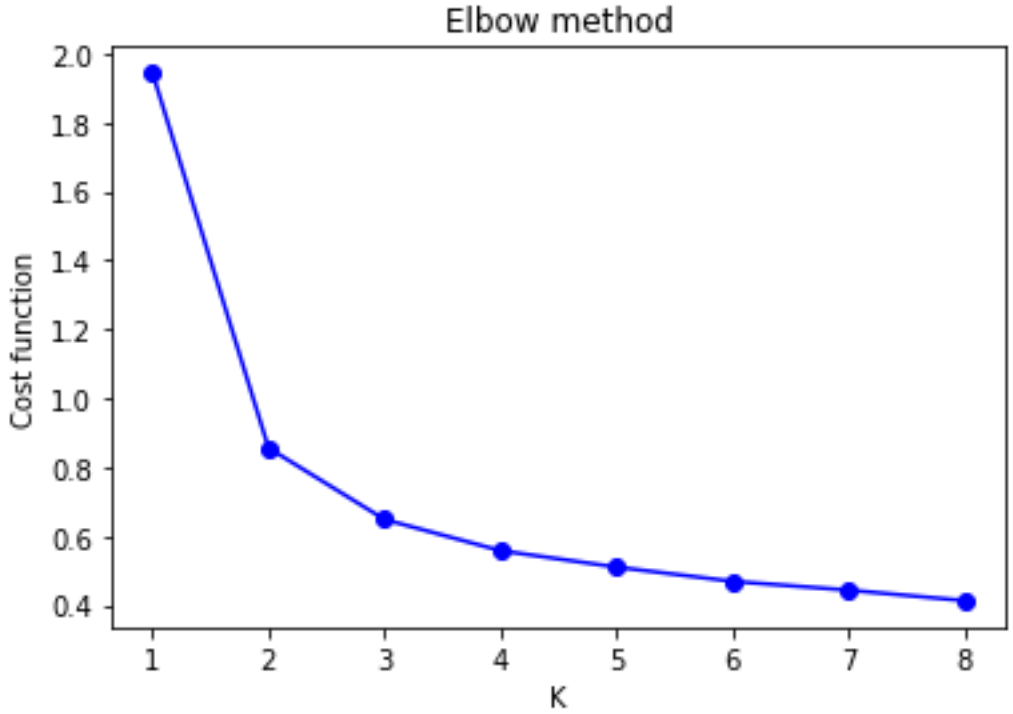
File ['MMM','AXP','AAPL','BA','CAT','CVX','CSCO','KO','DIS','DD']

```
# Filename: kmeansDJI.py
listDji = ['MMM','AXP','AAPL','BA','CAT','CVX','CSCO','KO','DIS','DD']
listTemp = [0] * len(listDji)
for i in range(len(listTemp)):
    listTemp[i] = create_df(listDji[i]).close      # a function for creating a DataFrame
status = [0] * len(listDji)
for i in range(len(status)):
    status[i] = np.sign(np.diff(listTemp[i]))
kmeans = KMeans(n_clusters = 3).fit(status)
pred = kmeans.predict(status)
print(pred)
```

Output:

[2 0 2 2 0 0 2 2 1 1]

Model Selection and Evaluation



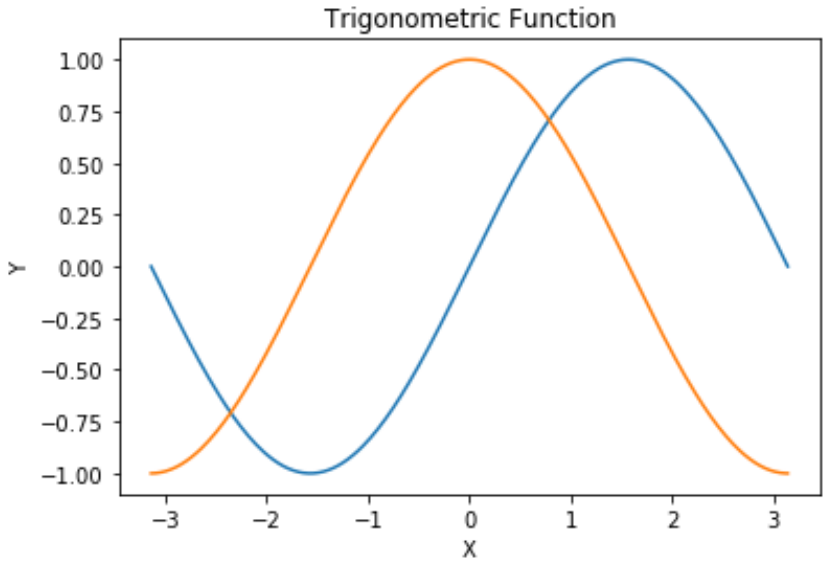
Data Processing Using Python

4 APPLICATIONS OF PYTHON INTO SCIENCE AND ENGINEERING FIELDS

Computation of Trigonometric Function

File

```
# Filename: mathA.py
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-np.pi, np.pi, 256)
s = np.sin(x)
c = np.cos(x)
plt.title('Trigonometric Function')
plt.xlabel('X')
plt.ylabel('Y')
plt.plot(x, s)
plt.plot(x, c)
```



Fast Fourier Transformation

Array：[1,1,…,1,-1,-1,…,1,1,1…,1]

File

```
# Filename: mathB.py
import scipy as sp
import matplotlib.pyplot as plt
listA = sp.ones(500)
listA[100:300] = -1
f = sp.fft(listA)
plt.plot(f)
```

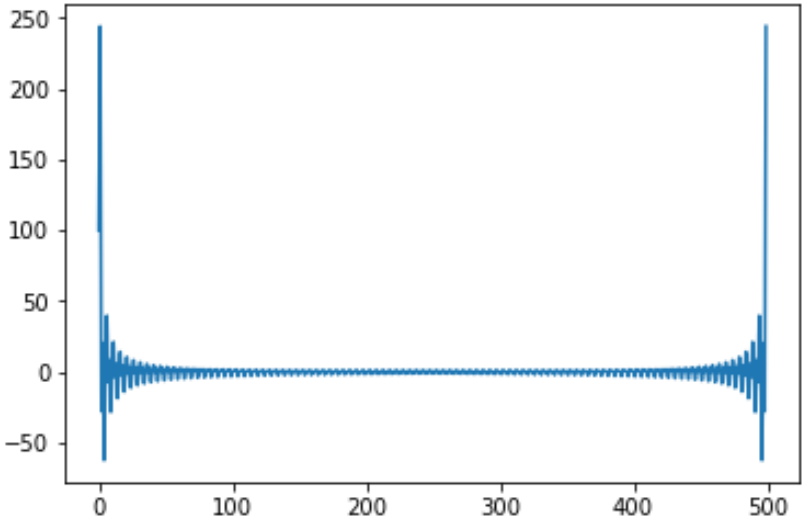


Image Processing

- **Useful Python Library**
  - Pillow(PIL)
  - OpenCV
  - Skimage

File

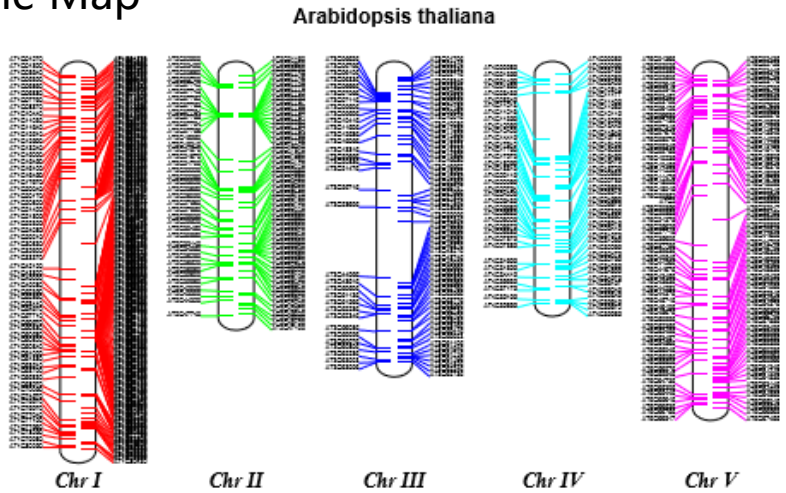
```
# Filename: pasteimg.py
from PIL import Image
im1 = Image.open('1.jpg')
print(im1.size, im1.format, im1.mode)
Image.open('1.jpg').save('2.png')
im2 = Image.open('2.png')
size = (288, 180)
im2.thumbnail(size)
out = im2.rotate(45)
im1.paste(out, (50,50))
```

Biopython

- Developed by Biopython, a group focusing on computational biology with Python
- Sequence, Alphabet and Chromosome Map

Source

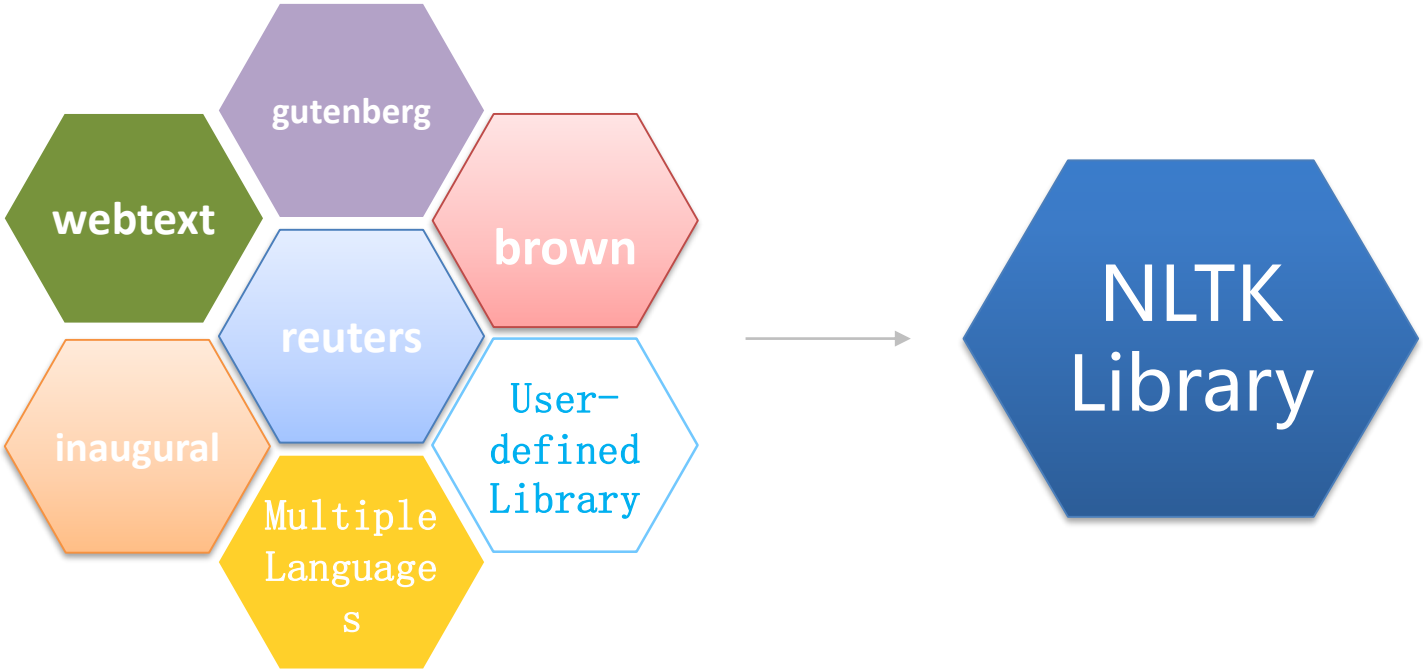
```
>>> from Bio.Seq import Seq
>>> my_seq = Seq("AGTACACTGGT")
>>> my_seq.alphabet
Alphabet()
>>> print(my_seq)
AGTACACTGGT
```



Data Processing Using Python

APPLICATIONS INTO HUMANITIES AND SOCIAL SCIENCES FIELDS

NLTK Library



Gutenberg Project

- Count all books currently included in Gutenberg Project

Source

```
>>> from nltk.corpus import gutenberg
>>> gutenberg.fileids()
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-
kjv.txt', 'blake-poems.txt', 'bryant-stories.txt', 'burgess-
busterbrown.txt', 'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-
brown.txt', 'chesterton-thursday.txt', 'edgeworth-parents.txt',
'melville-moby_dick.txt', 'milton-paradise.txt', 'shakespeare-caesar.txt',
'shakespeare-hamlet.txt', 'shakespeare-macbeth.txt', 'whitman-
leaves.txt']
```

Gutenberg Project

- Some simple calculation

Source

```
>>> from nltk.corpus import gutenberg
>>> allwords = gutenberg.words('shakespeare-hamlet.txt')
>>> len(allwords)
37360
>>> len(set(allwords))
5447
>>> allwords.count('Hamlet')
99
>>> A = set(allwords)
>>> longwords = [w for w in A if len(w) > 12]
>>> print(sorted(longwords))
```

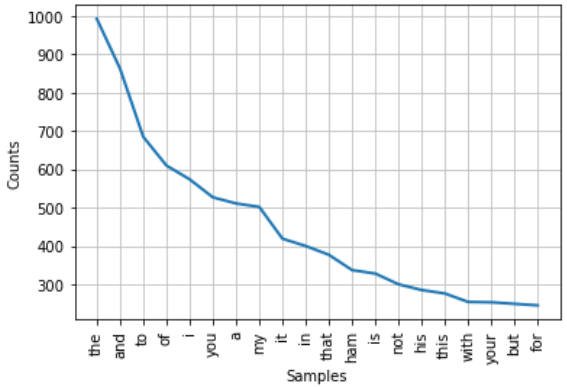
Output:

```
['Circumstances',
'Guildensterne',
'Incontinencie',
'Recognizances',
'Vnderstanding',
'determination',
'encompassement',
'entertainment',
'imperfections',
'indifferently',
'instrumentall',
'reconcilement',
'stubbornnesse',
'transformation',
'vnderstanding']
```

Gutenberg Project

File

```
# Filename: freqG20.py
from nltk.corpus import gutenberg
from nltk.probability import *
fd2 = FreqDist([sx.lower() for sx in allwords if sx.isalpha()]
print(fd2.B())
print(fd2.N())
fd2.tabulate(20)
fd2.plot(20)
fd2.plot(20, cumulative = True)
```



Output:

```
4699
30266
 the  and  to  of  i  you  a  my  it  in that  ham
is  not  his this with your  but  for
 993  863  685  610  574  527  511  502  419  400
377  337  328  300  285  276  254  253  249  245
```

Inaugural Library

Source

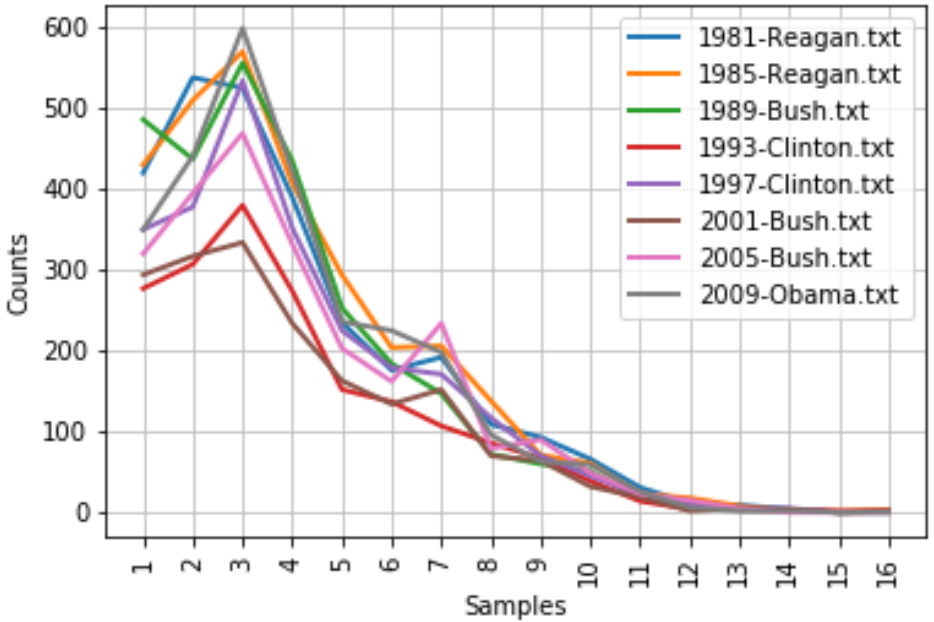
```
>>> from nltk.corpus import inaugural
>>> from nltk.probability import *
>>> fd3 = FreqDist([s for s in inaugural.words()])
>>> print(fd3.freq('freedom'))
0.00119394791917
```

File

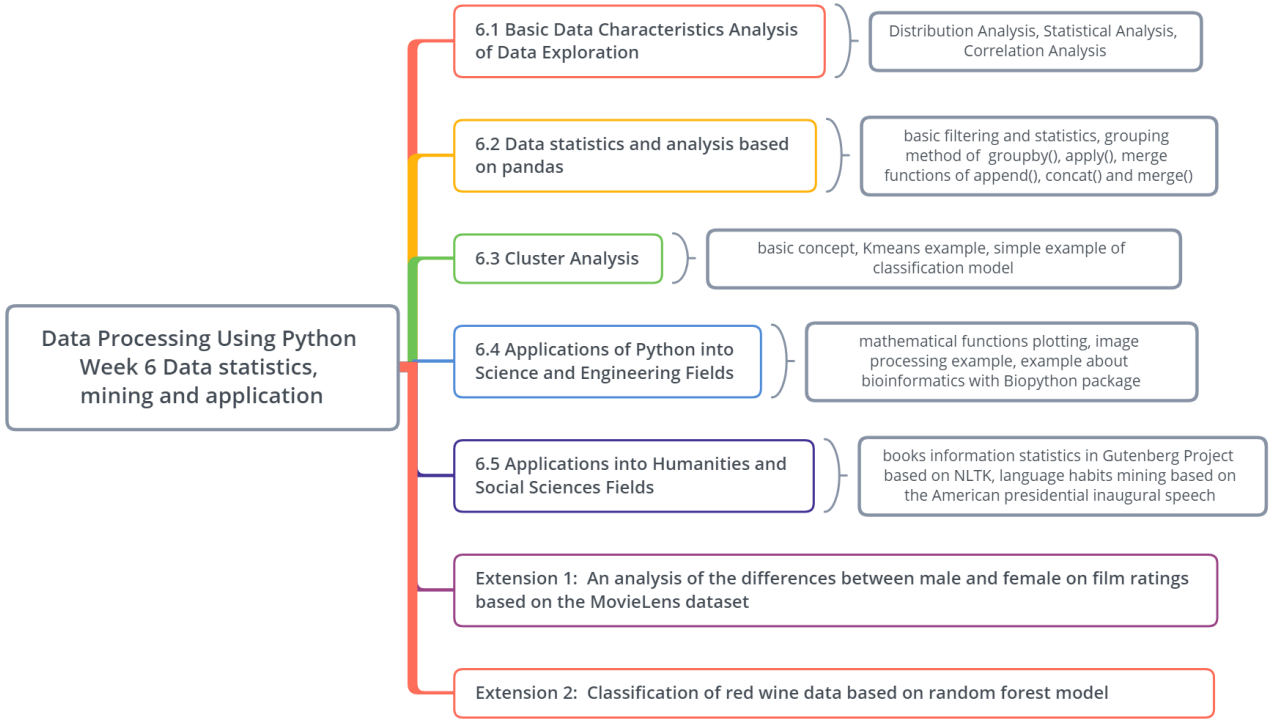
```
# Filename: inaugural.py
from nltk.corpus import inaugural
from nltk.probability import *
cfd = ConditionalFreqDist(
            (fileid, len(w))
            for fileid in inaugural.fileids()
            for w in inaugural.words(fileid)
            if fileid > '1980' and fileid < '2010')
print(cfd.items())
cfd.plot()
```

Inaugural Library

Output:
dict_items([('1981-Reagan.txt', FreqDist({2: 538, 3: 525, 1: 420, 4: 390, 5: 235, 7: 192, 6: 176, 8: 109, 9: 93, 10: 66, ...})), ... , ('2005-Bush.txt', FreqDist({3: 469, 2: 395, 4: 332, 1: 320, 7: 234, 5: 203, 6: 162, 9: 90, 8: 79, 10: 49, ...})), ('2009-Obama.txt', FreqDist({3: 599, 2: 441, 4: 422, 1: 350, 5: 236, 6: 225, 7: 198, 8: 96, 9: 63, 10: 59, ...}))])



Summary


Data Processing Using Python Week 6 Data statistics, mining and application
6.1 Basic Data Characteristics Analysis of Data Exploration
Distribution Analysis, Statistical Analysis, Correlation Analysis
6.2 Data statistics and analysis based on pandas
basic filtering and statistics, grouping method of groupby(), apply(), merge functions of append(), concat() and merge()
6.3 Cluster Analysis
basic concept, Kmeans example, simple example of classification model
6.4 Applications of Python into Science and Engineering Fields
mathematical functions plotting, image processing example, example about bioinformatics with Biopython package
6.5 Applications into Humanities and Social Sciences Fields
books information statistics in Gutenberg Project based on NLTK, language habits mining based on the American presidential inaugural speech
Extension 1: An analysis of the differences between male and female on film ratings based on the MovieLens dataset
Extension 2: Classification of red wine data based on random forest model