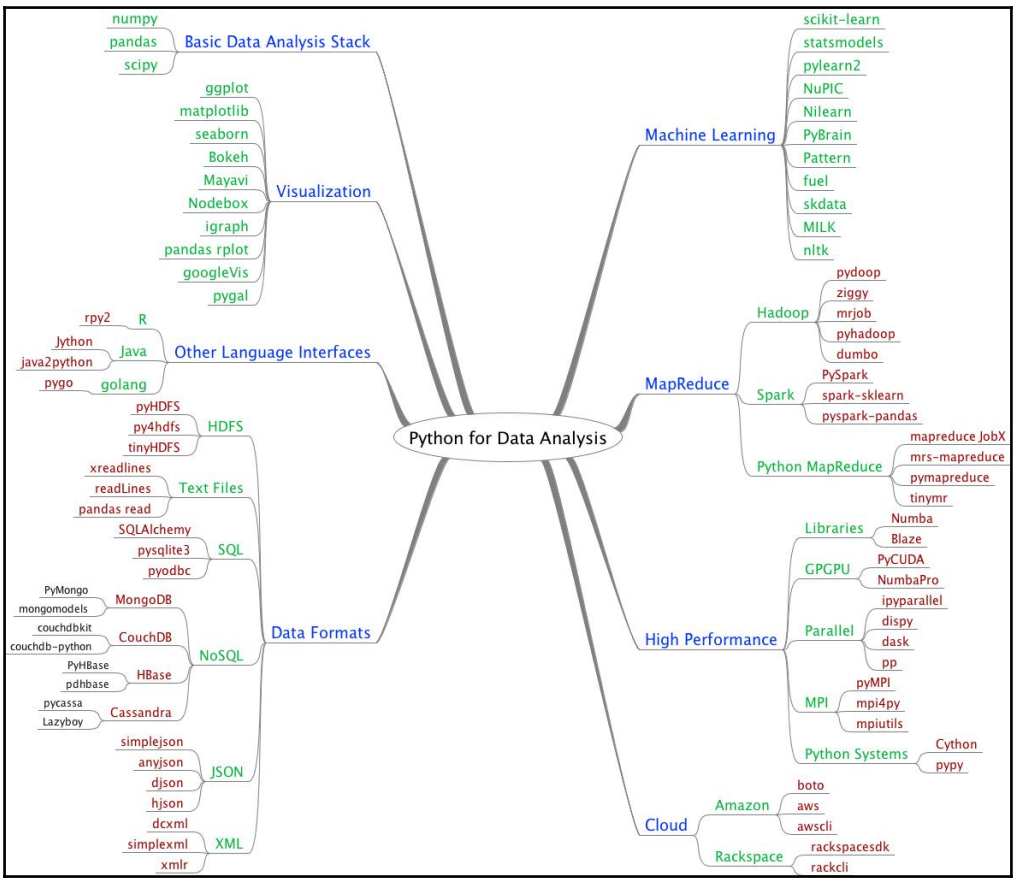
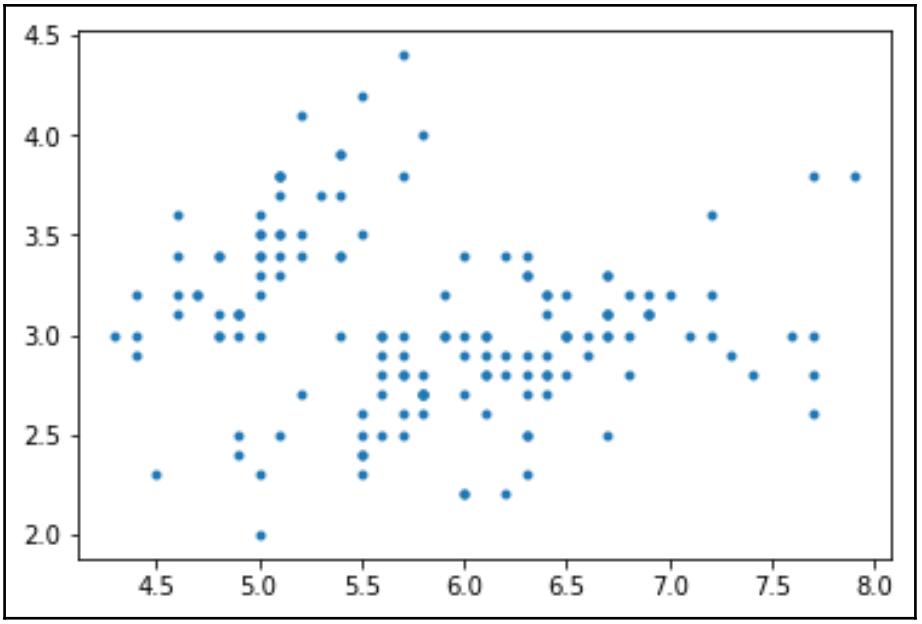
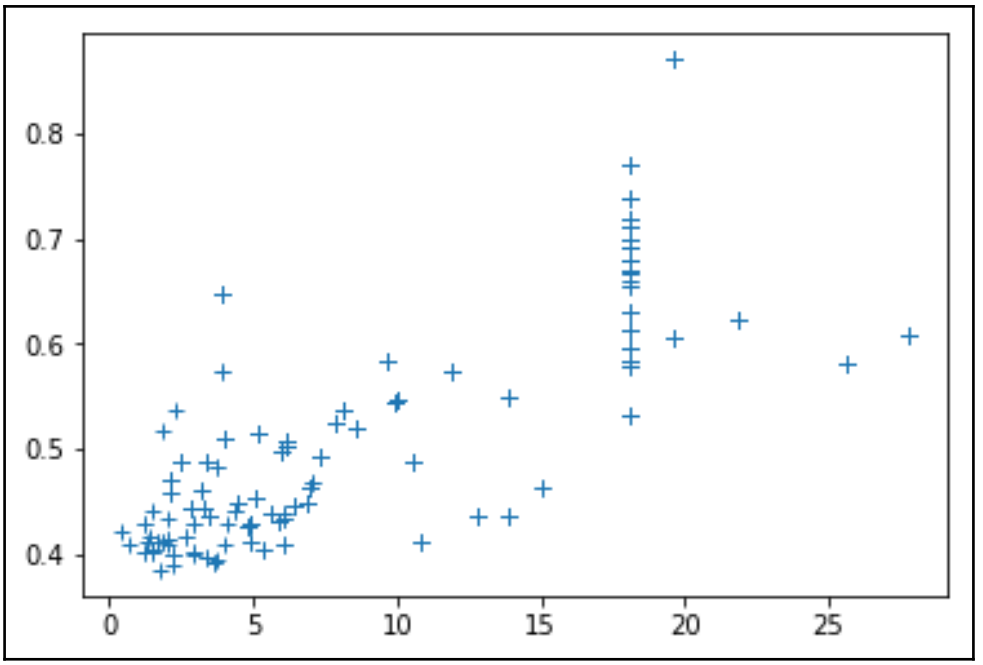
1

Chapter 1: Getting Started with Python Libraries

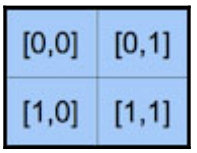


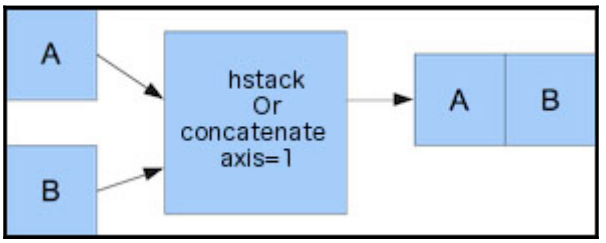
```
In [1]: import numpy

In [2]: help(numpy.ar
  numpy.arange        numpy.arctan        numpy.argpartition numpy.array2string
  numpy.arccos        numpy.arctan2       numpy.argsort      numpy.array_equal
  numpy.arccosh       numpy.arctanh       numpy.argwhere     numpy.array_equiv  >
  numpy.arcsin        numpy.argmax        numpy.around       numpy.array_repr
  numpy.arcsinh       numpy.argmin        numpy.array        numpy.array_split
```

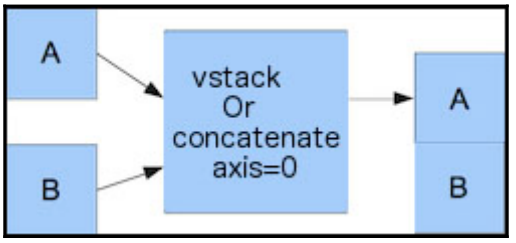


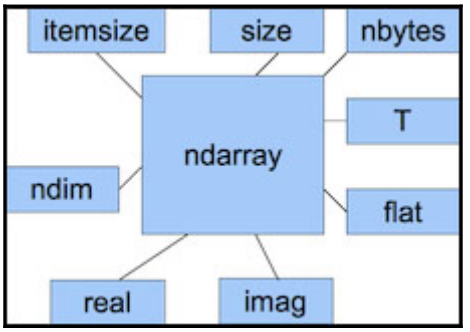

0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25

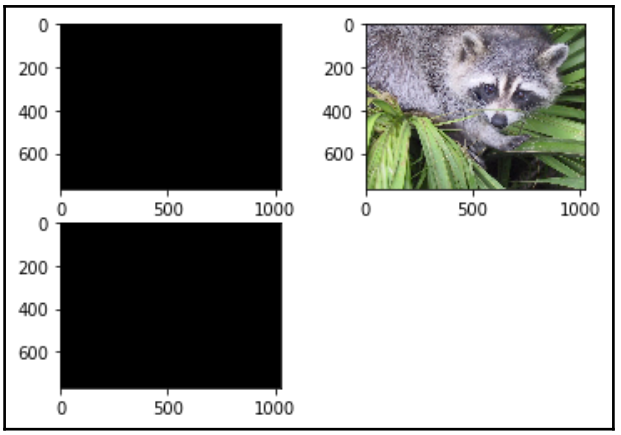
Chapter 2: NumPy Arrays

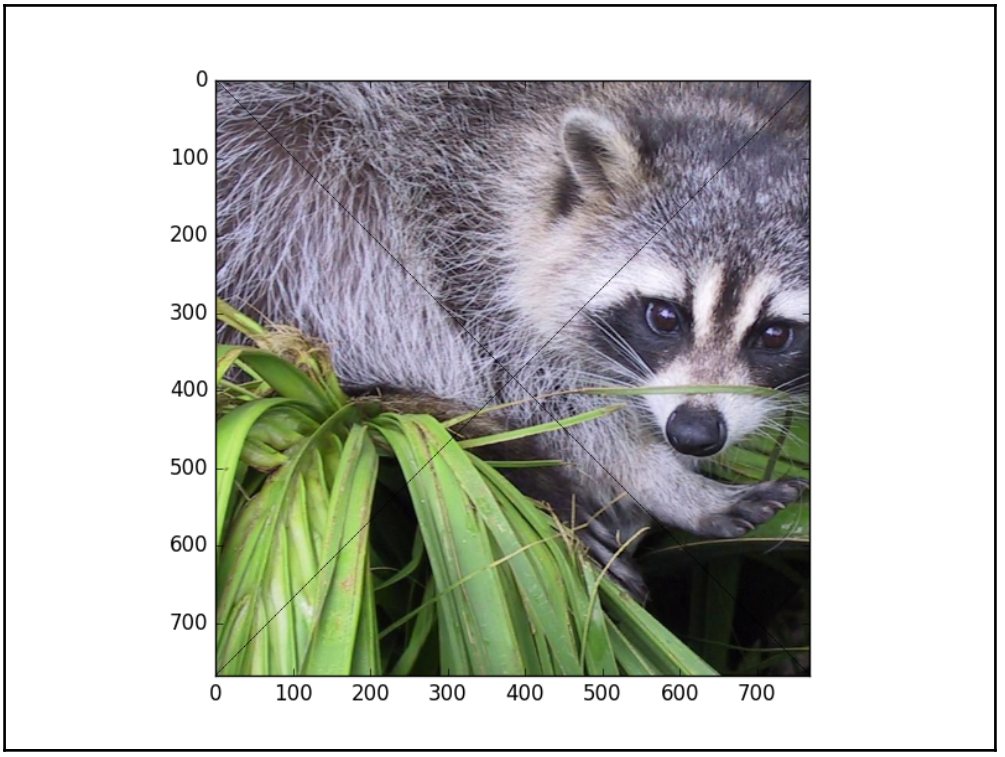


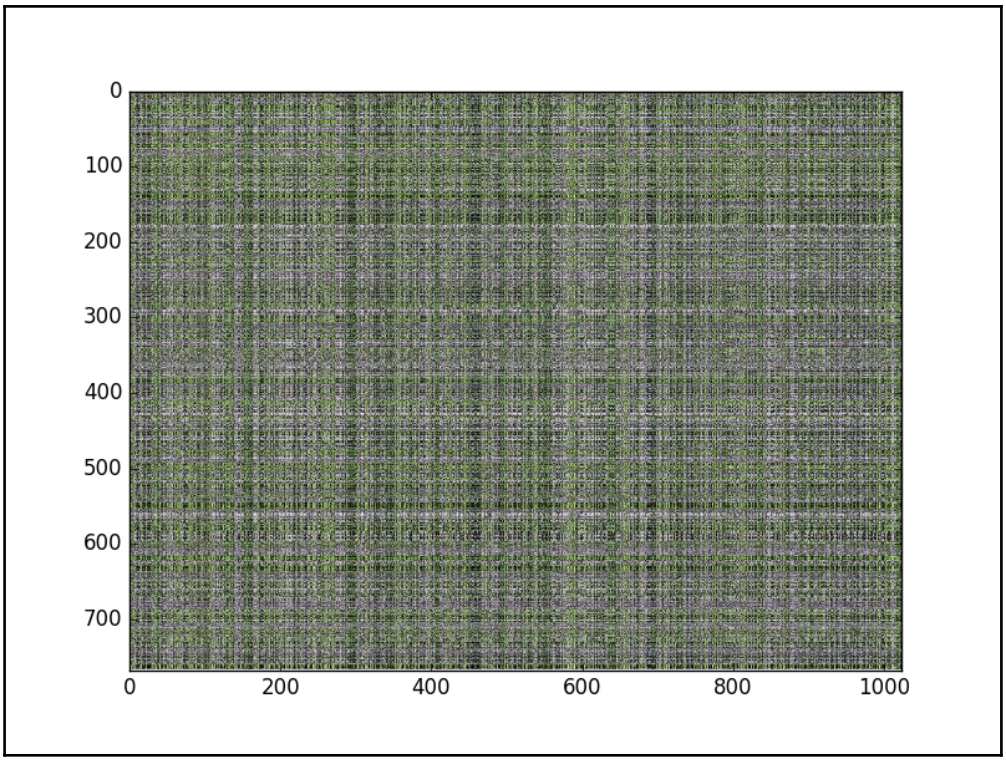


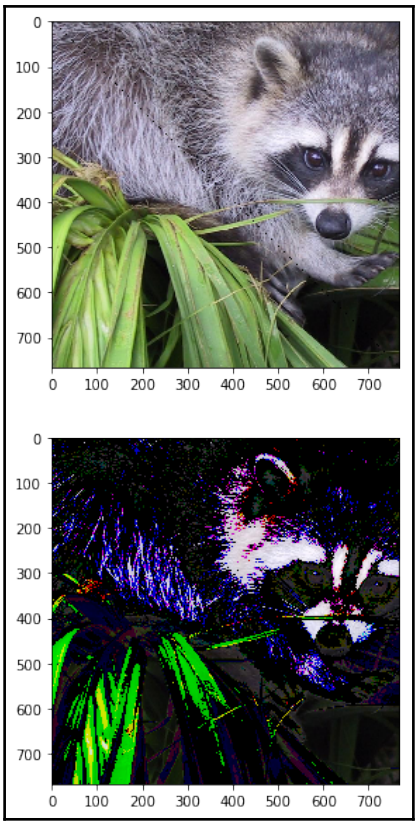


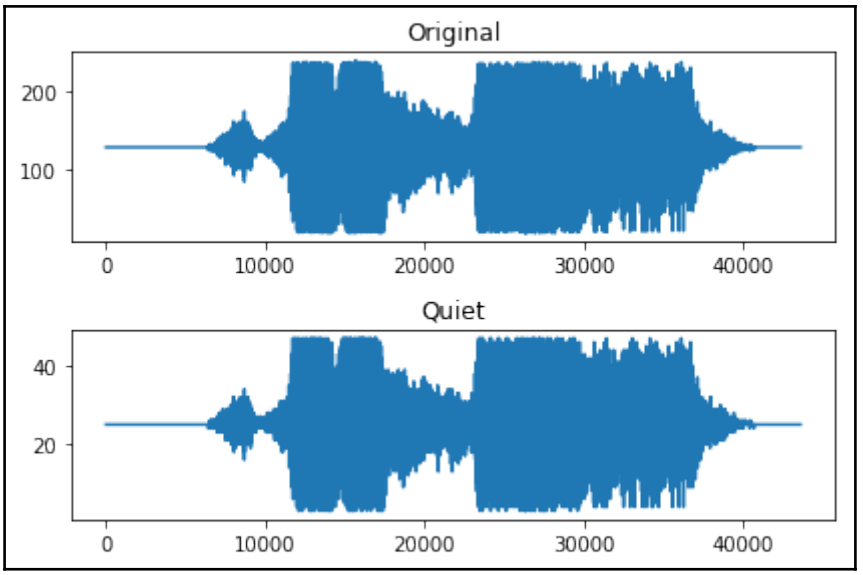



0
200
400
600
0
500
1000
0
200
400
600
0
500
1000
0
200
400
600
0
500
1000


0
100
200
300
400
500
600
700
0
100
200
300
400
500
600
700


0
100
200
300
400
500
600
700
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
0 100 200 300 400 500 600 700
0
100
200
300
400
500
600
700
0 100 200 300 400 500 600 700


Original
200
100
0
10000
20000
30000
40000
Quiet
40
20
0
10000
20000
30000
40000

Chapter 3: The Pandas Primer

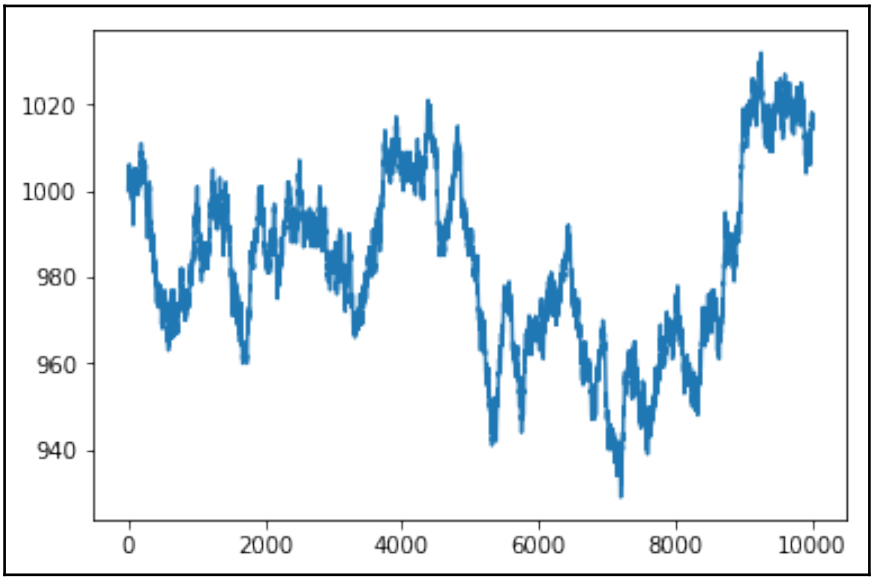
```
Data types Country                                                  object
CountryID                                                            int64
Continent                                                            int64
Adolescent fertility rate (%)                                      float64
Adult literacy rate (%)                                            float64
Gross national income per capita (PPP international $)             float64
Net primary school enrolment ratio female (%)                      float64
Net primary school enrolment ratio male (%)                        float64
Population (in thousands) total                                    float64
```

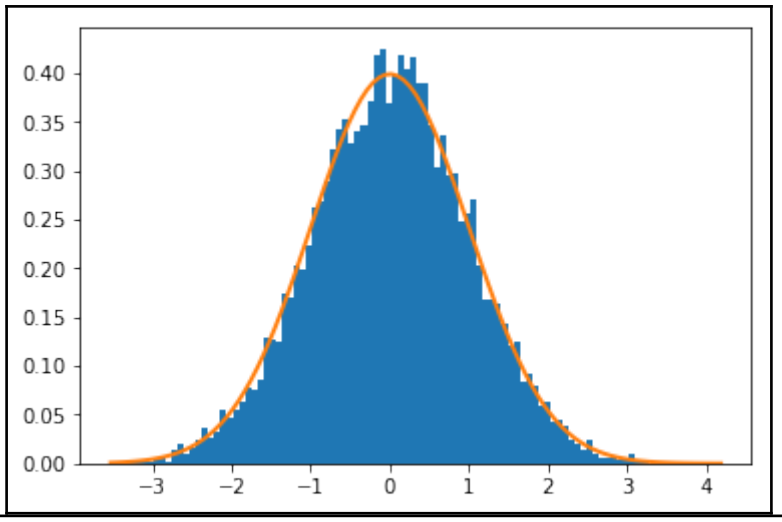
```
Describe        Yearly Mean Total Sunspot Number  Yearly Mean Standard Deviation  \
count                         316.000000                       198.000000
mean                           79.503481                         8.030303
std                            62.057114                         3.807299
min                             0.000000                         1.700000
25%                            25.050000                         4.725000
50%                            66.700000                         7.700000
75%                           116.400000                        10.475000
max                           269.300000                        19.100000

       Number of Observations  Definitive/Provisional Indicator
count              198.000000                             316.0
mean              1424.888889                               1.0
std               2394.898980                               0.0
min                150.000000                               1.0
25%                365.000000                               1.0
50%                365.000000                               1.0
75%                366.000000                               1.0
max               8903.000000                               1.0

Non NaN observations Yearly Mean Total Sunspot Number    316
Yearly Mean Standard Deviation      198
Number of Observations              198
Definitive/Provisional Indicator    316
dtype: int64

MAD Yearly Mean Total Sunspot Number      50.987620
Yearly Mean Standard Deviation         3.125375
Number of Observations              1777.463524
Definitive/Provisional Indicator       0.000000
dtype: float64

Median Yearly Mean Total Sunspot Number     66.7
Yearly Mean Standard Deviation         7.7
Number of Observations               365.0
Definitive/Provisional Indicator       1.0
dtype: float64
```

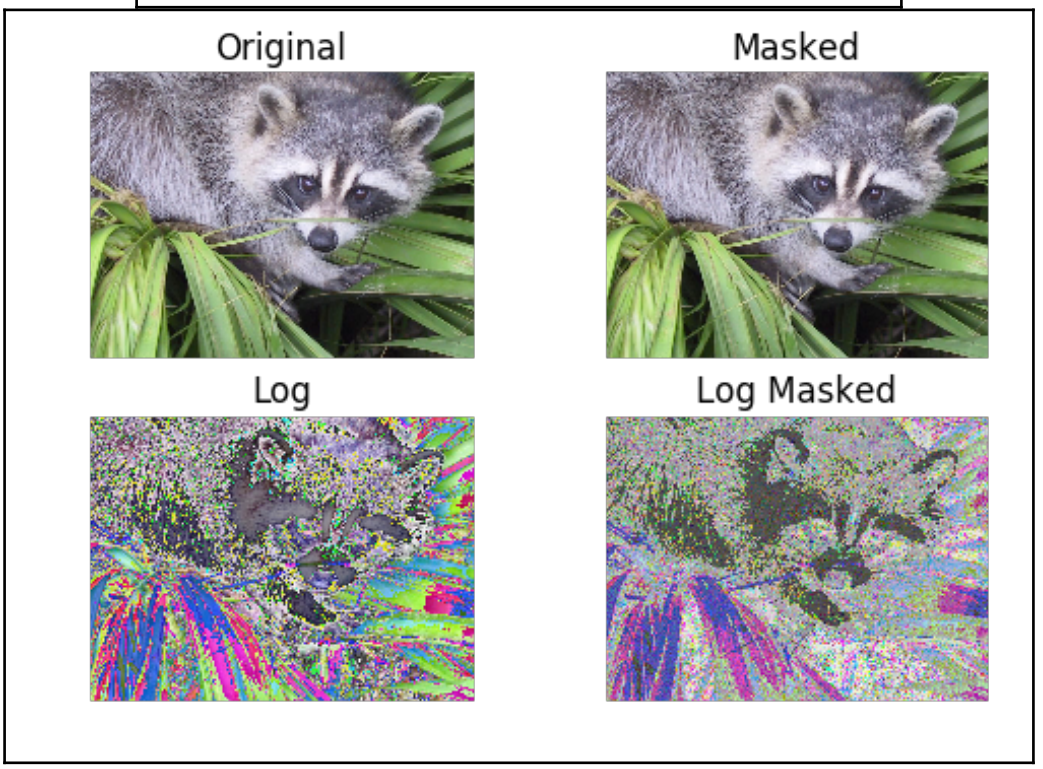
```
Min Yearly Mean Total Sunspot Number      0.0
Yearly Mean Standard Deviation        1.7
Number of Observations              150.0
Definitive/Provisional Indicator      1.0
dtype: float64

Max Yearly Mean Total Sunspot Number     269.3
Yearly Mean Standard Deviation        19.1
Number of Observations              8903.0
Definitive/Provisional Indicator      1.0
dtype: float64

Mode    Yearly Mean Total Sunspot Number  Yearly Mean Standard Deviation  \
0                              18.3                              9.2

   Number of Observations  Definitive/Provisional Indicator
0                   365.0                             1.0

Standard Deviation Yearly Mean Total Sunspot Number      62.057114
Yearly Mean Standard Deviation         3.807299
Number of Observations              2394.898980
Definitive/Provisional Indicator       0.000000
dtype: float64

Variance Yearly Mean Total Sunspot Number    3.851085e+03
Yearly Mean Standard Deviation      1.449552e+01
Number of Observations              5.735541e+06
Definitive/Provisional Indicator    0.000000e+00
dtype: float64

Skewness Yearly Mean Total Sunspot Number    0.799452
Yearly Mean Standard Deviation      0.555067
Number of Observations              1.876098
Definitive/Provisional Indicator    0.000000
dtype: float64

Kurtosis Yearly Mean Total Sunspot Number   -0.143733
Yearly Mean Standard Deviation     -0.244310
Number of Observations              1.783261
Definitive/Provisional Indicator    0.000000
dtype: float64
```

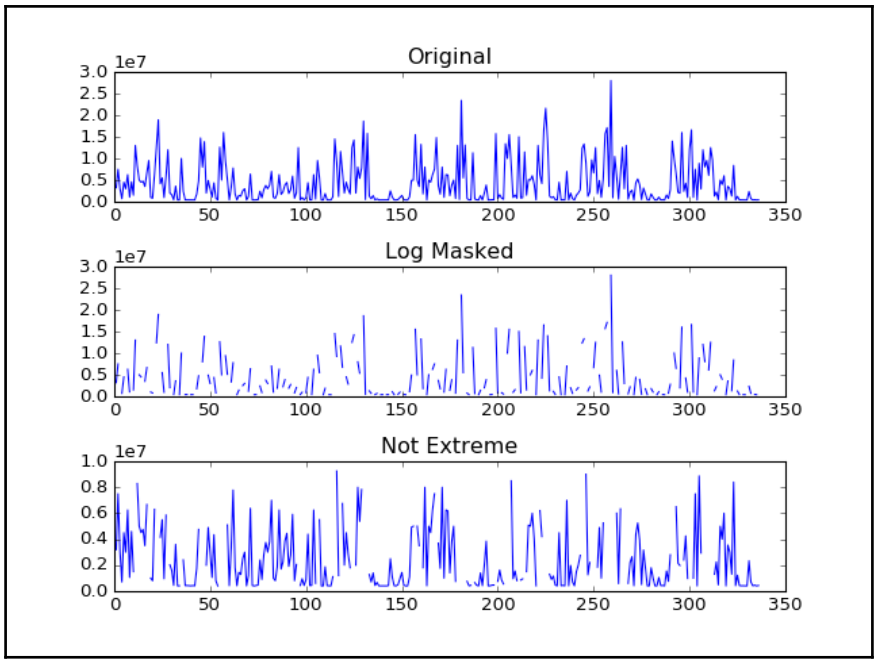
Chapter 4: Statistics and Linear Algebra

```
A
[[ 1 -2  1]
 [ 0  2 -8]
 [-4  5  9]]
b
[ 0  8 -9]
```

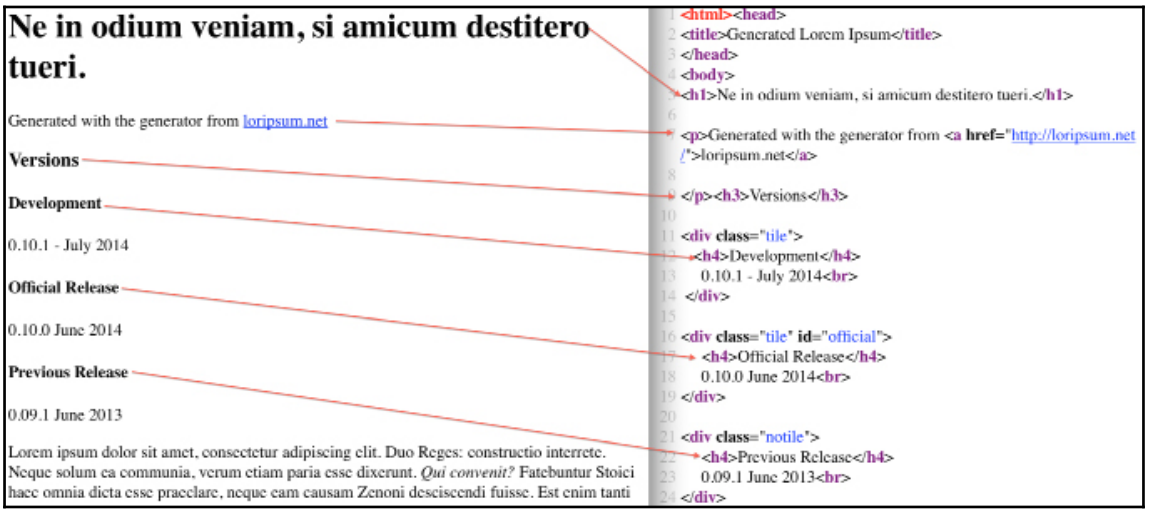



0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−3
−2
−1
0
1
2
3
4

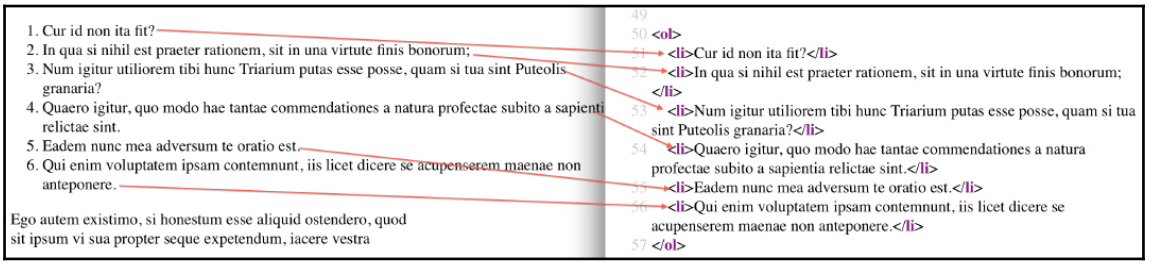

Original
Masked
Log
Log Masked


Original
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300
350
Log Masked
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
250
300
350
Not Extreme
1e7
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350

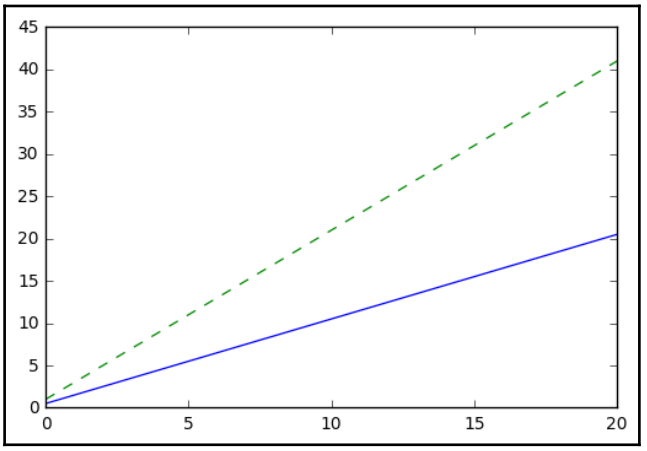
Chapter 5: Retrieving, Processing and Storing Data

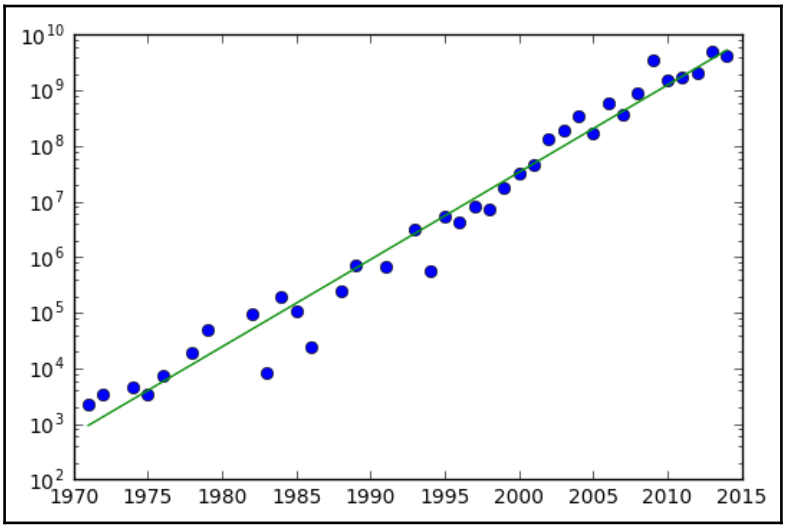


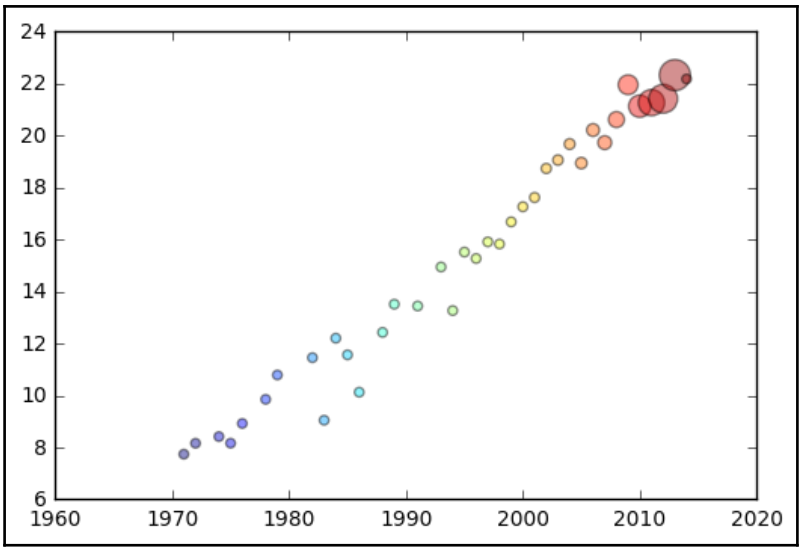
,

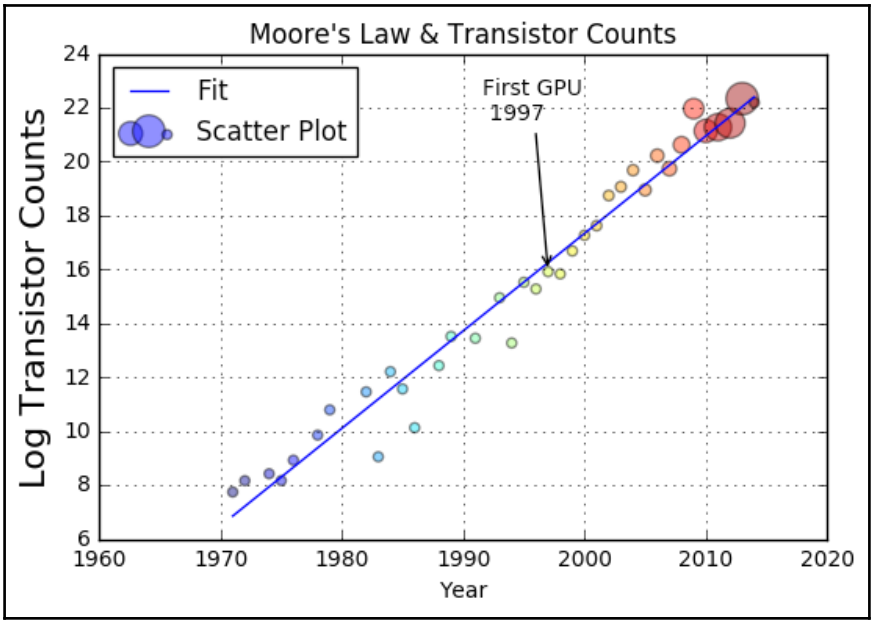


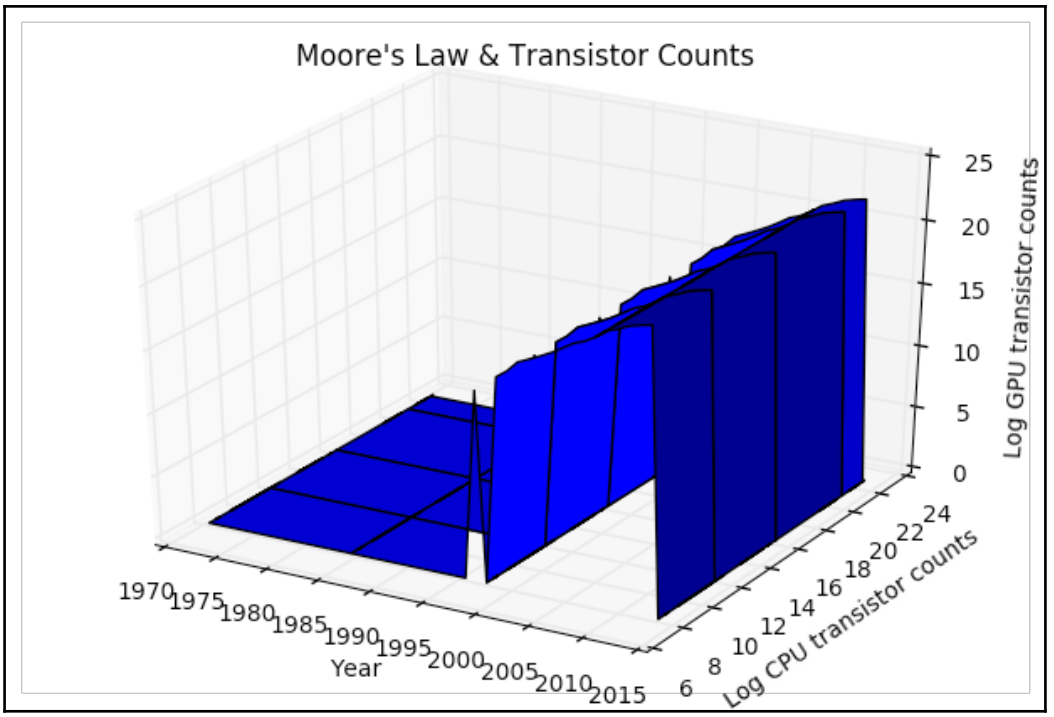
Chapter 6: Data Visualization

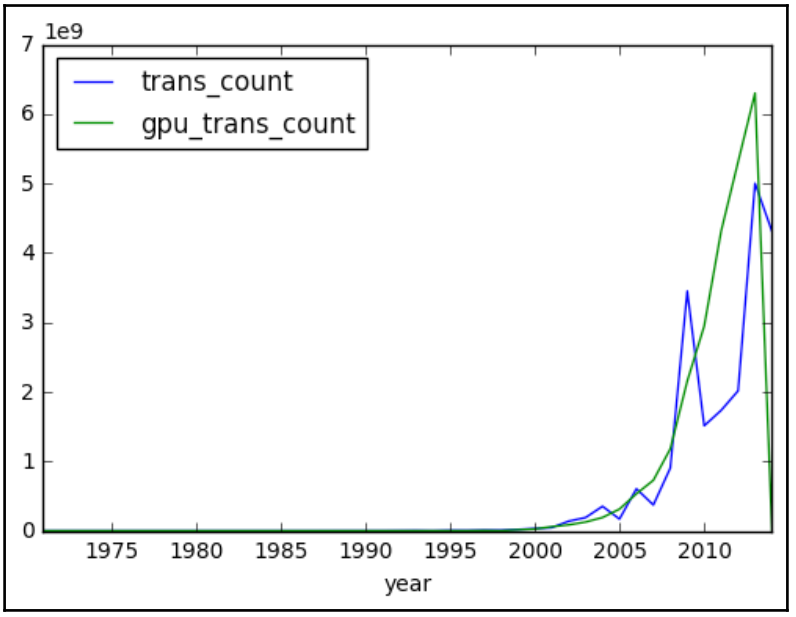


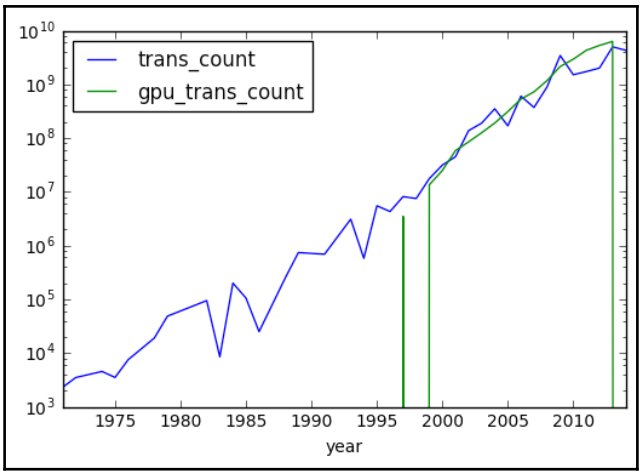


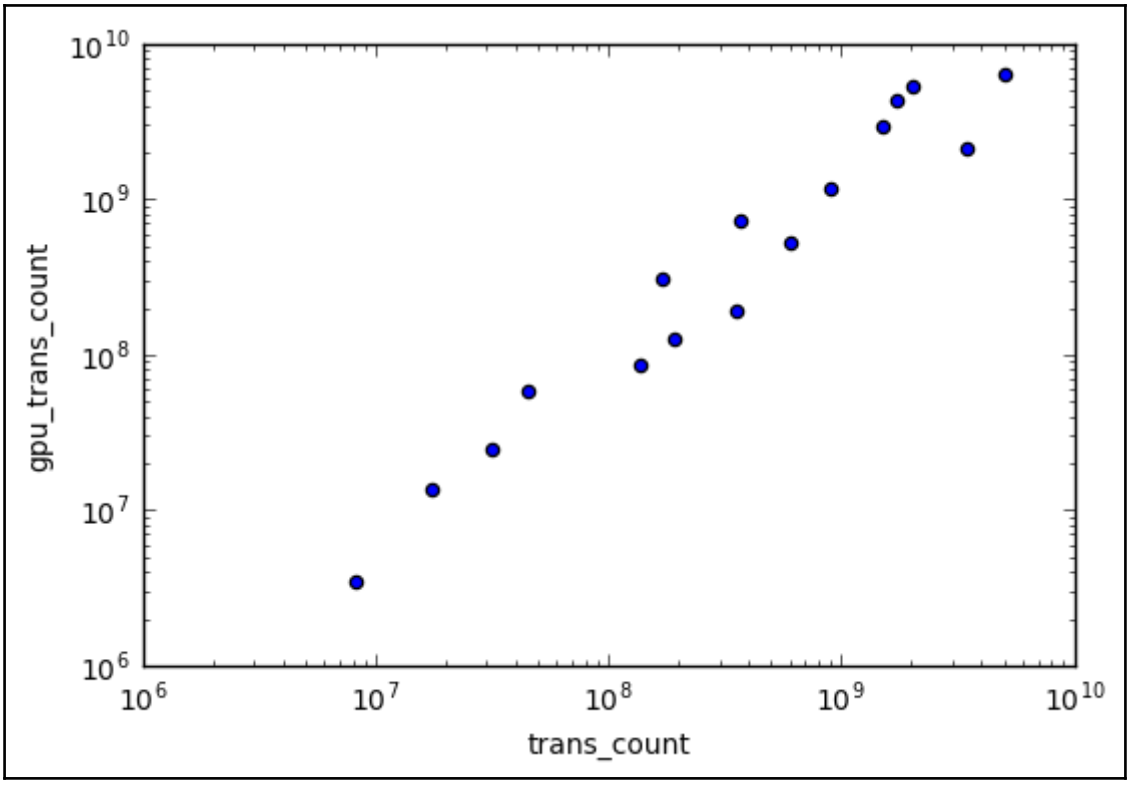

24
22
20
18
16
14
12
10
8
6
1960
1970
1980
1990
2000
2010
2020

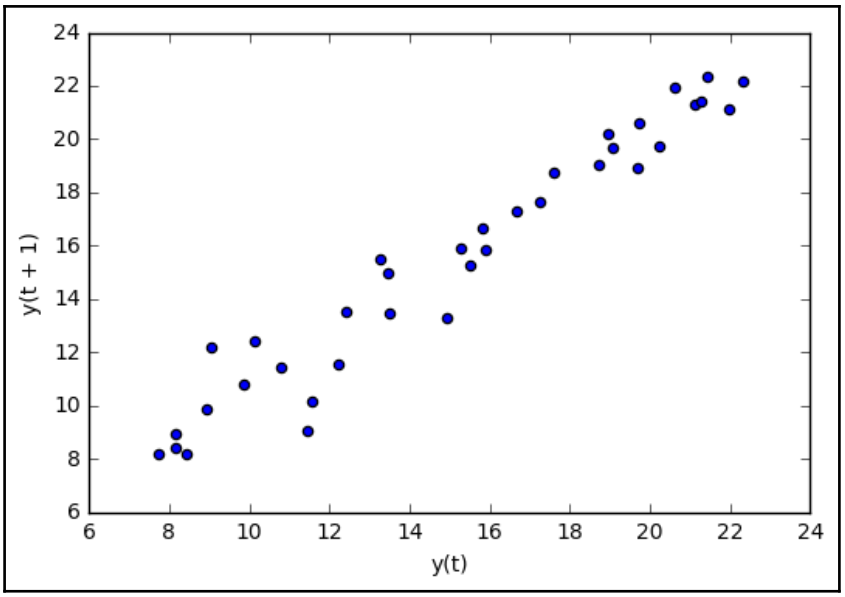

Moore's Law & Transistor Counts
Fit
Scatter Plot
First GPU
1997
Log Transistor Counts
Year
24
22
20
18
16
14
12
10
8
6
1960
1970
1980
1990
2000
2010
2020

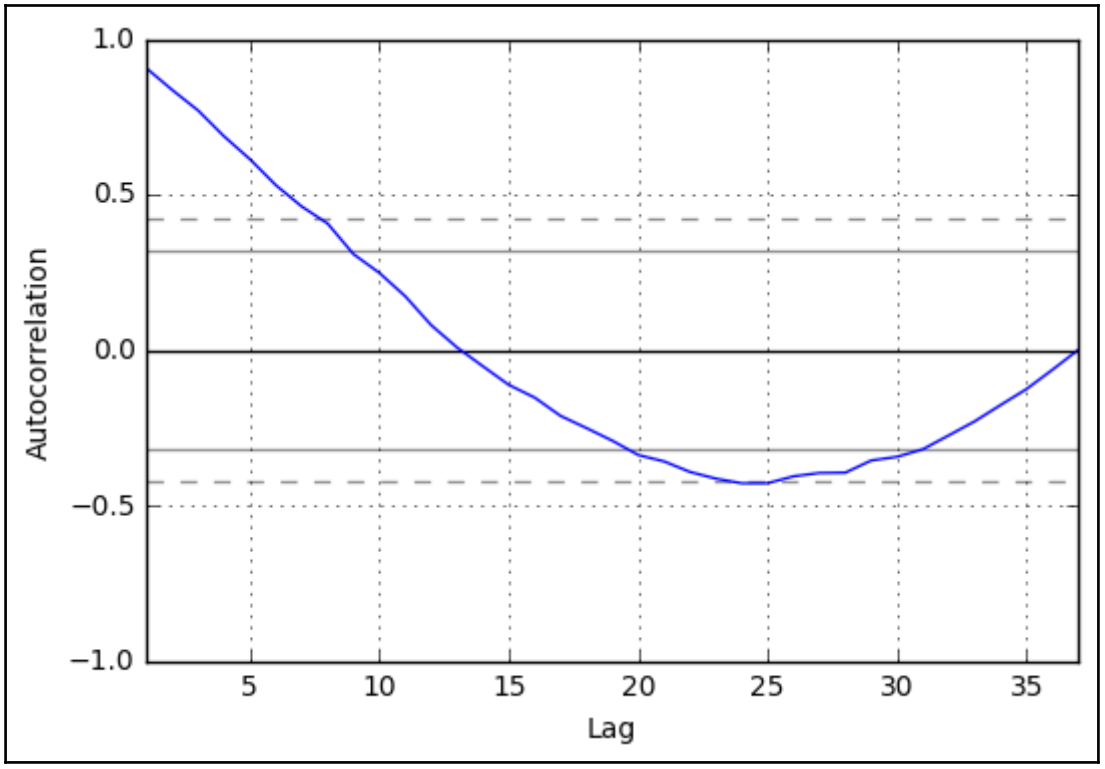

Moore's Law & Transistor Counts
Year
Log CPU transistor counts
Log GPU transistor counts
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
6
8
10
12
14
16
18
20
22
24
0
5
10
15
20
25

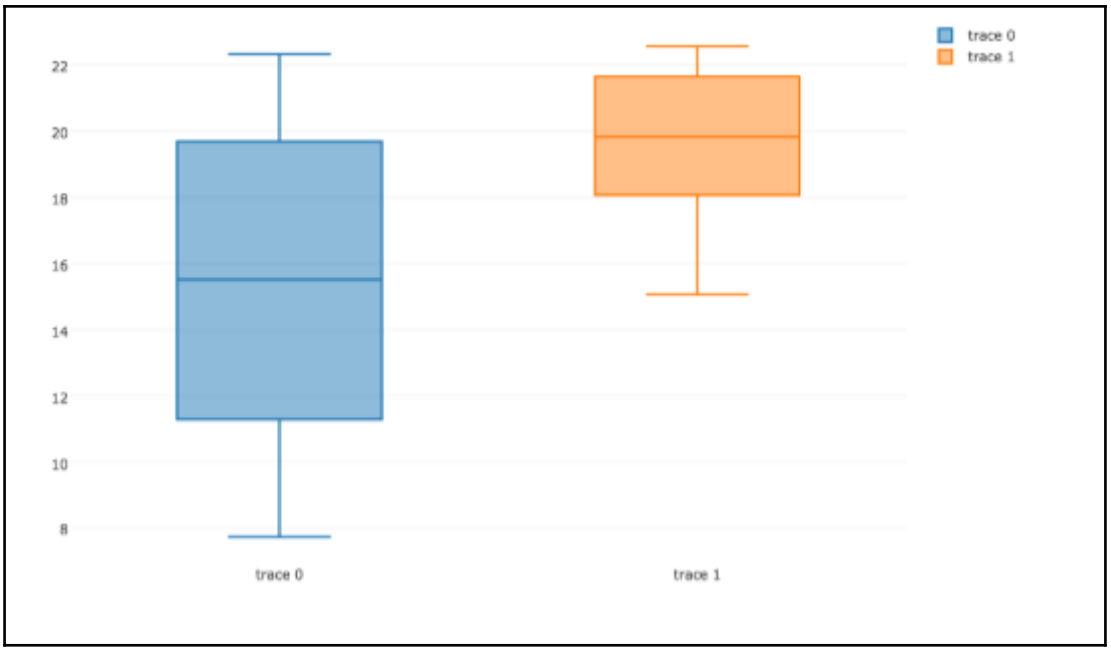

1e9
trans_count
gpu_trans_count
year
0
1
2
3
4
5
6
7
1975
1980
1985
1990
1995
2000
2005
2010


trans_count
gpu_trans_count
year

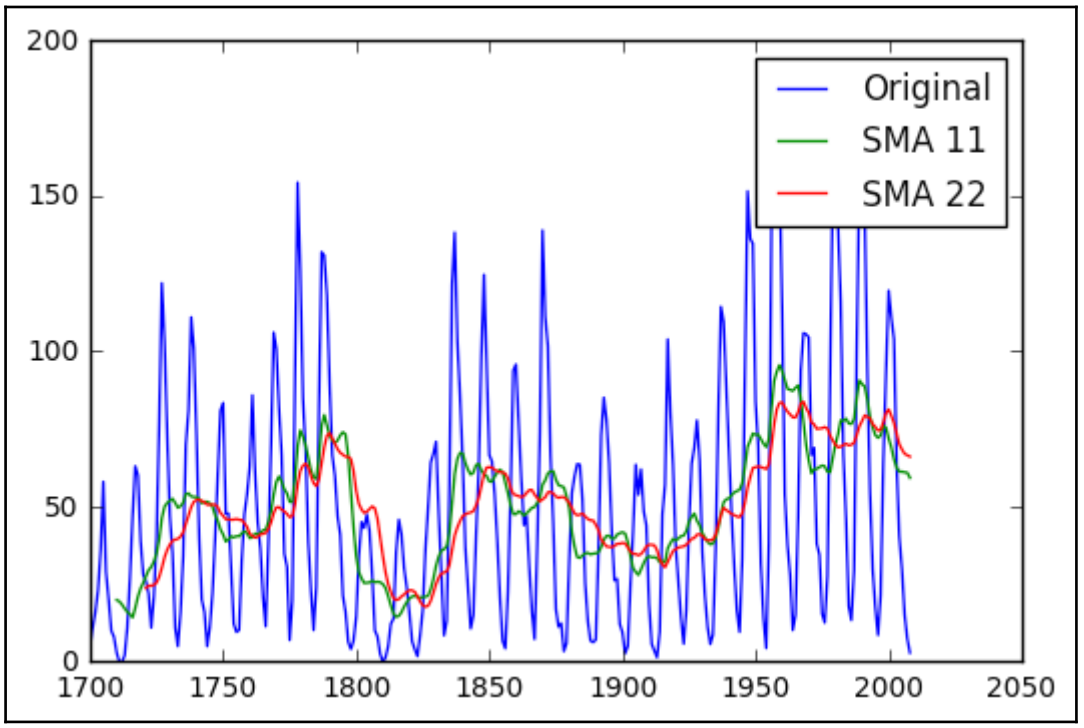

gpu_trans_count
trans_count


y(t + 1)
y(t)
24
22
20
18
16
14
12
10
8
6
6
8
10
12
14
16
18
20
22
24


Autocorrelation
Lag
1.0
0.5
0.0
−0.5
−1.0
5
10
15
20
25
30
35


trace 0
trace 1
22
20
18
16
14
12
10
8
trace 0
trace 1

Chapter 7: Signal Processing and Time Series

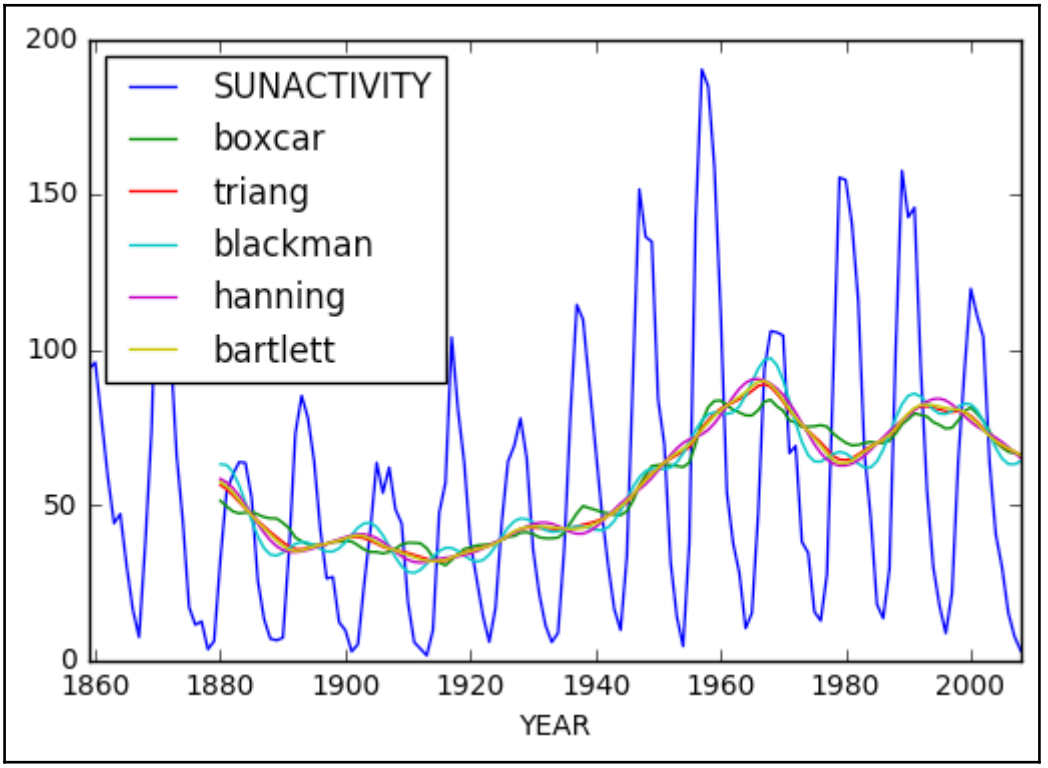


$$w(n) = 1 - \left| \frac{n - \frac{N-1}{2}}{\frac{L}{2}} \right|$$

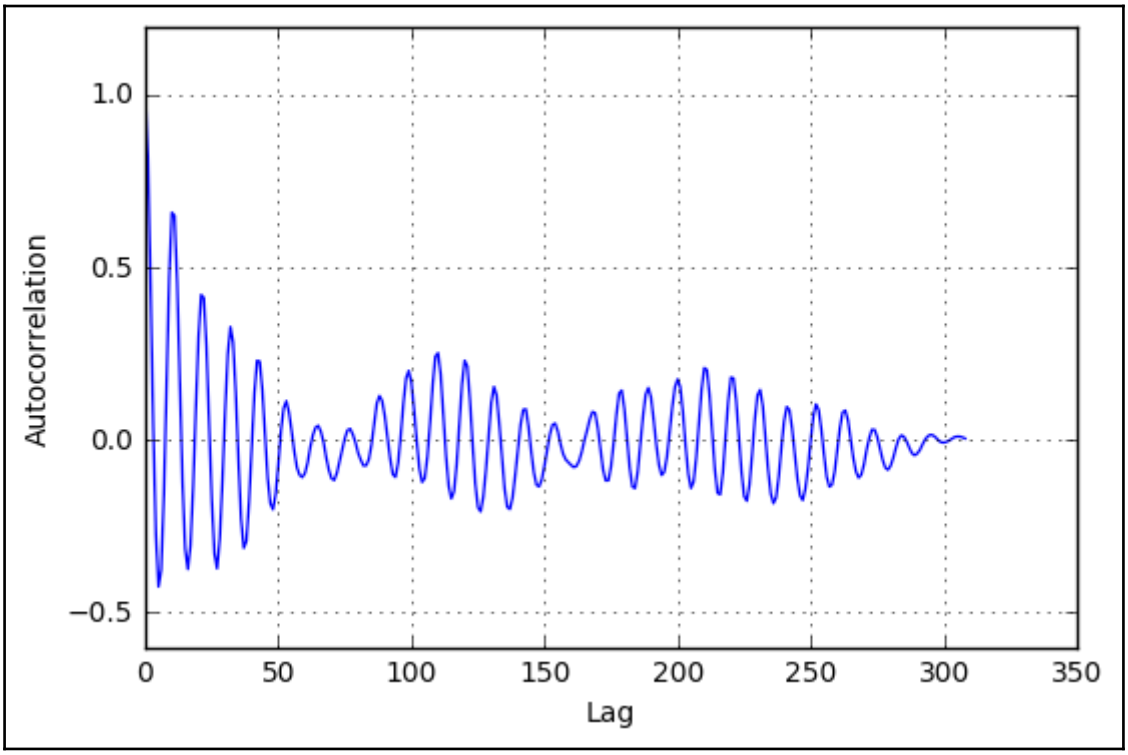
$$w(n) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right)$$

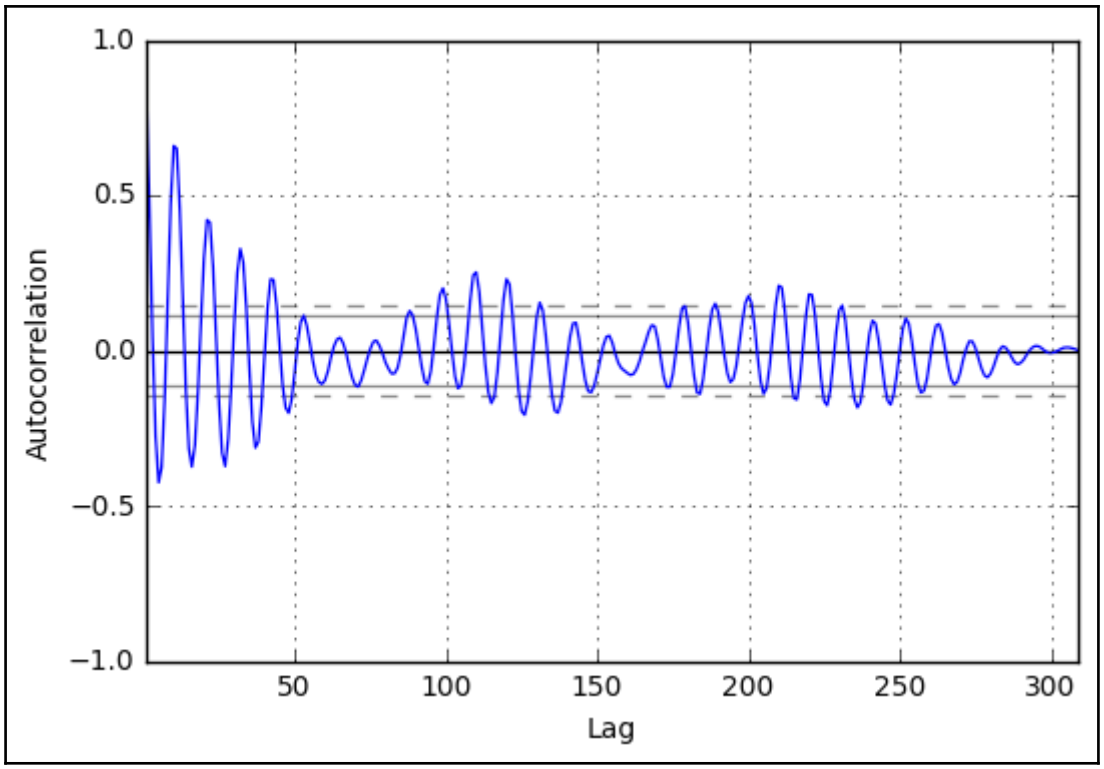
$$a_0 = \frac{1-\alpha}{2}; a_1 = \frac{1}{2}; a_2 = \frac{\alpha}{2}$$

$$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$$

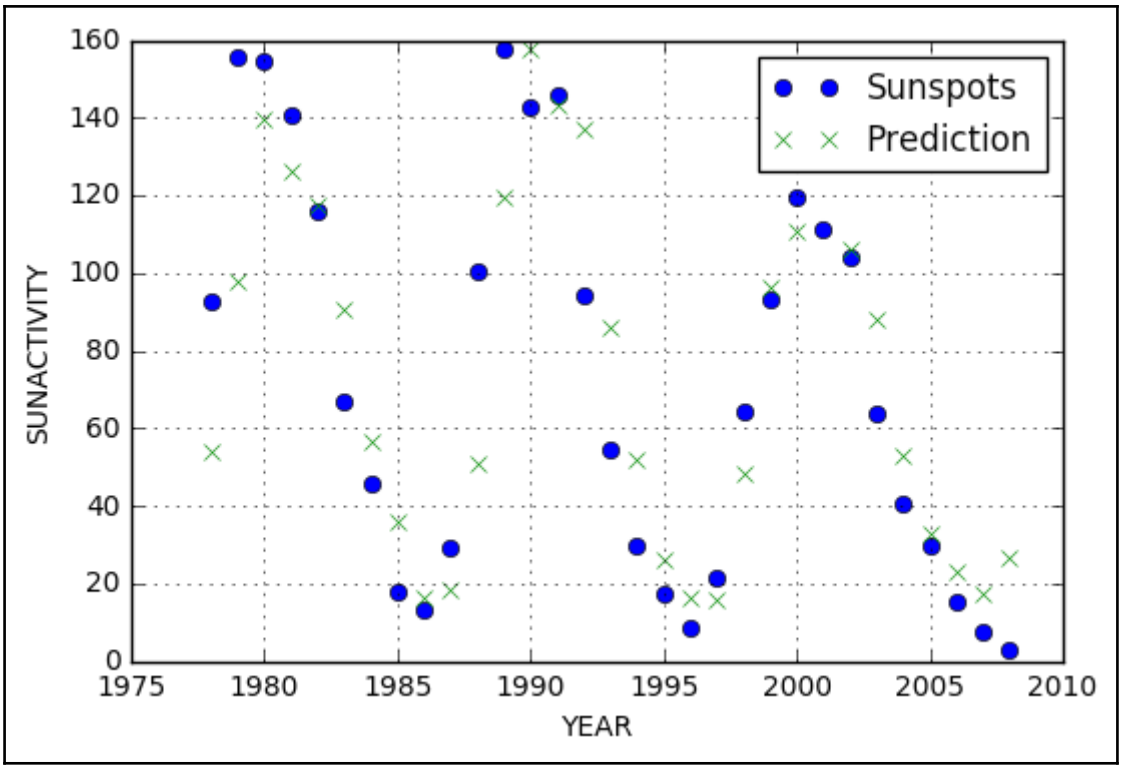


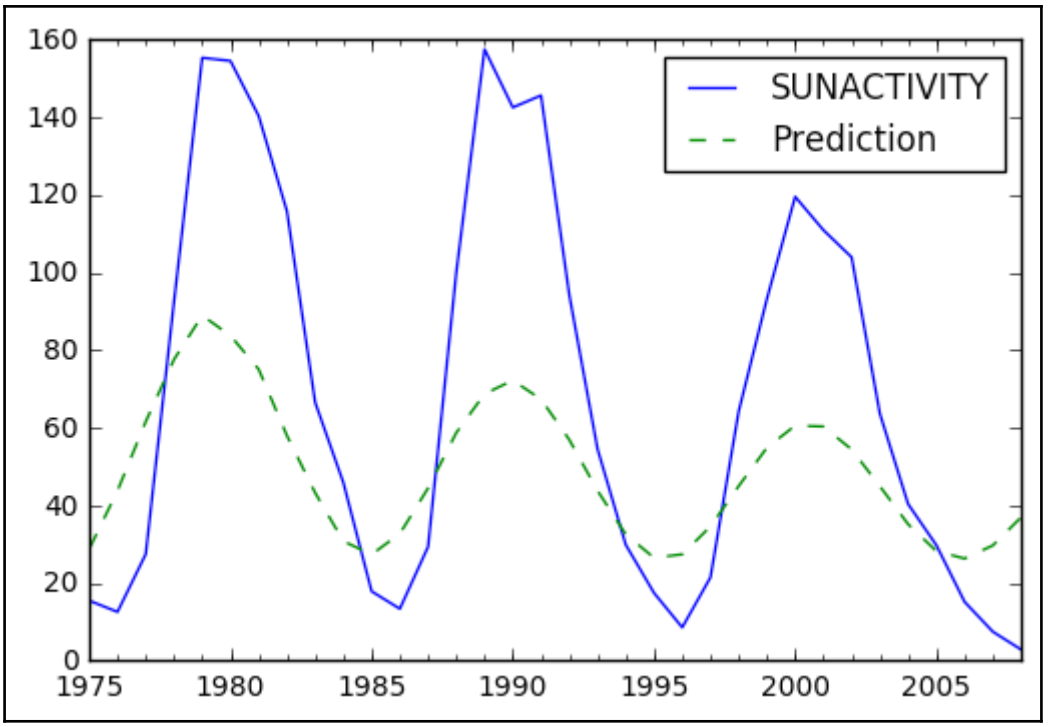
$$\frac{E\left[\left(x_t - \mu_t\right)\left(x_s - \mu_s\right)\right]}{\sigma_t \sigma_s}$$

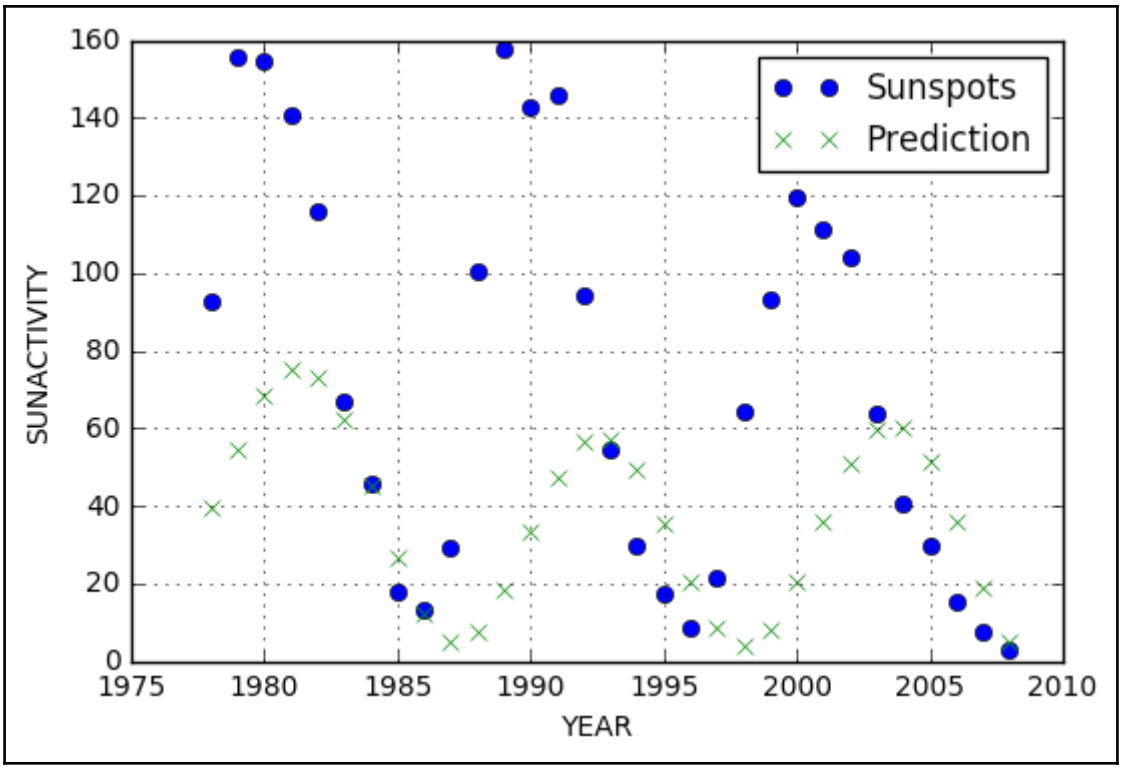




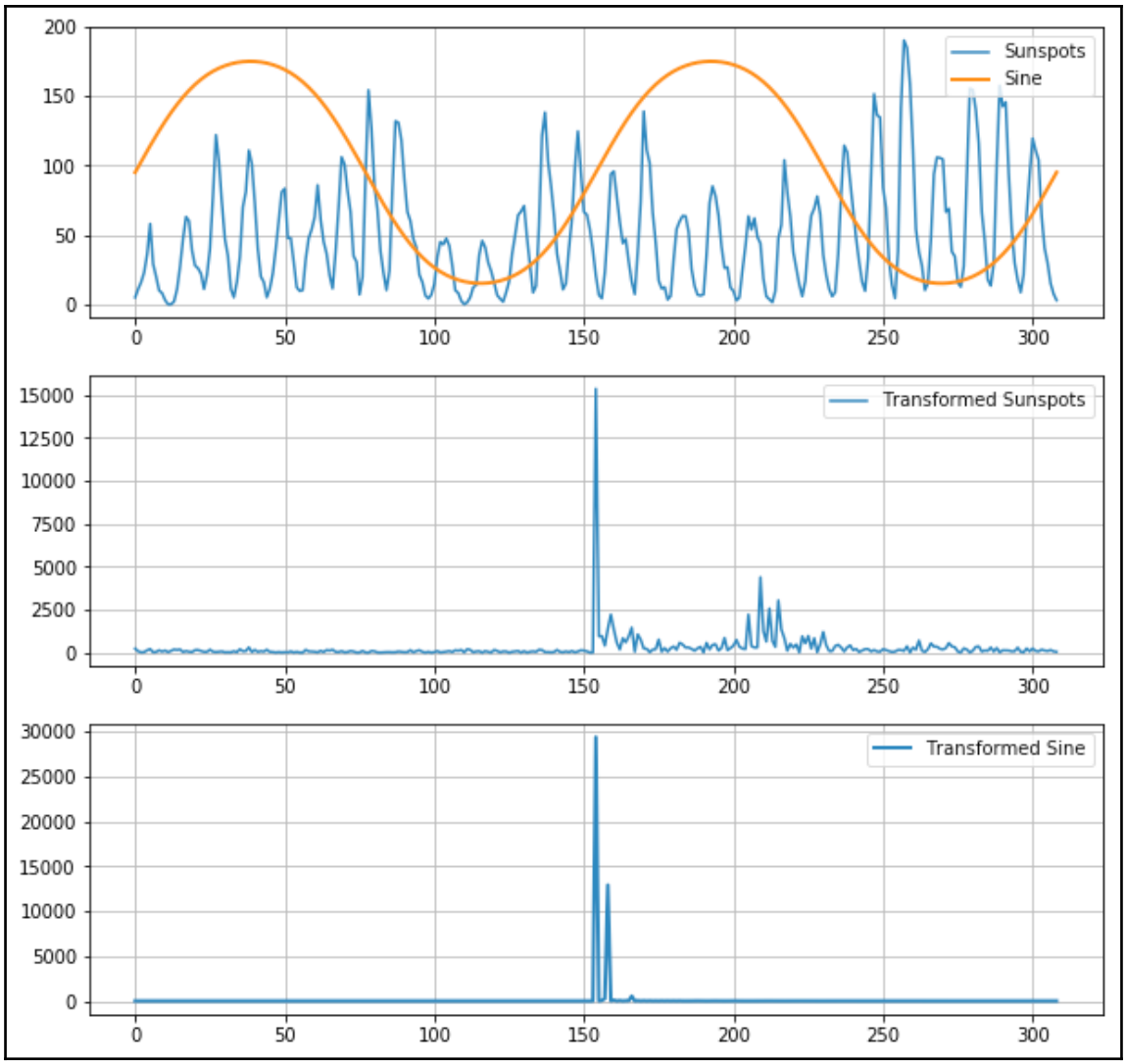
$$x_t = c + \sum_{i=1}^{p} a_i x_{t-i} + \in_t$$

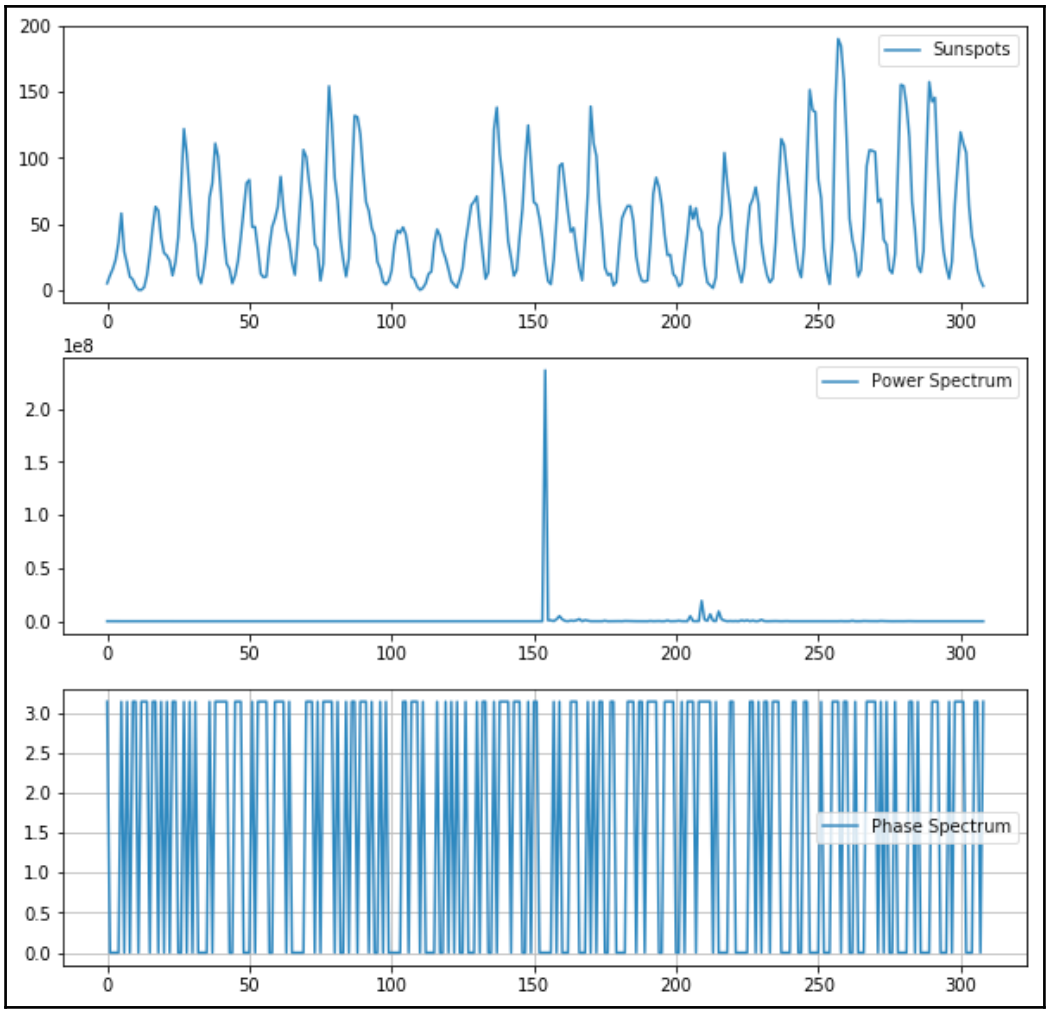

Sunspots
Prediction
SUNACTIVITY
YEAR
0
20
40
60
80
100
120
140
160
1975
1980
1985
1990
1995
2000
2005
2010

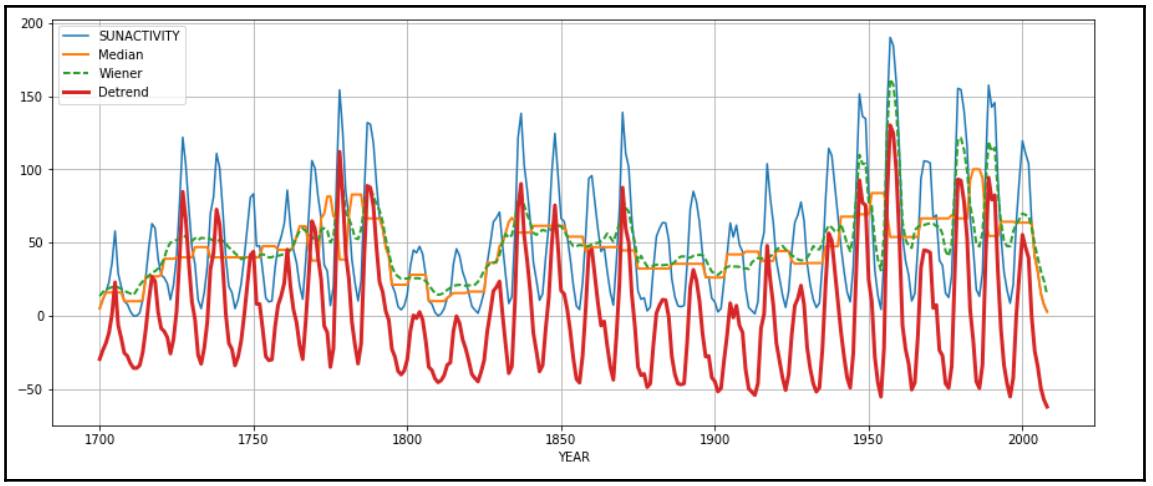

160
140
120
100
80
60
40
20
0
1975
1980
1985
1990
1995
2000
2005
SUNACTIVITY
Prediction



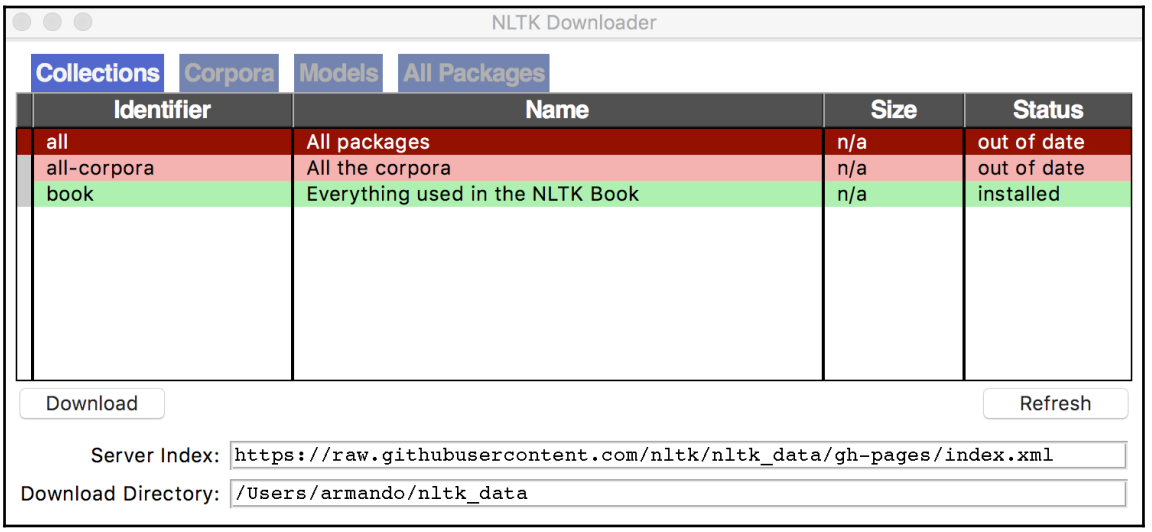
$$\sum_{t=-\infty}^{\infty} \chi\left[t\right] e^{-i\omega t}$$


Sunspots
Sine
Transformed Sunspots
Transformed Sine


Sunspots
1e8
Power Spectrum
Phase Spectrum

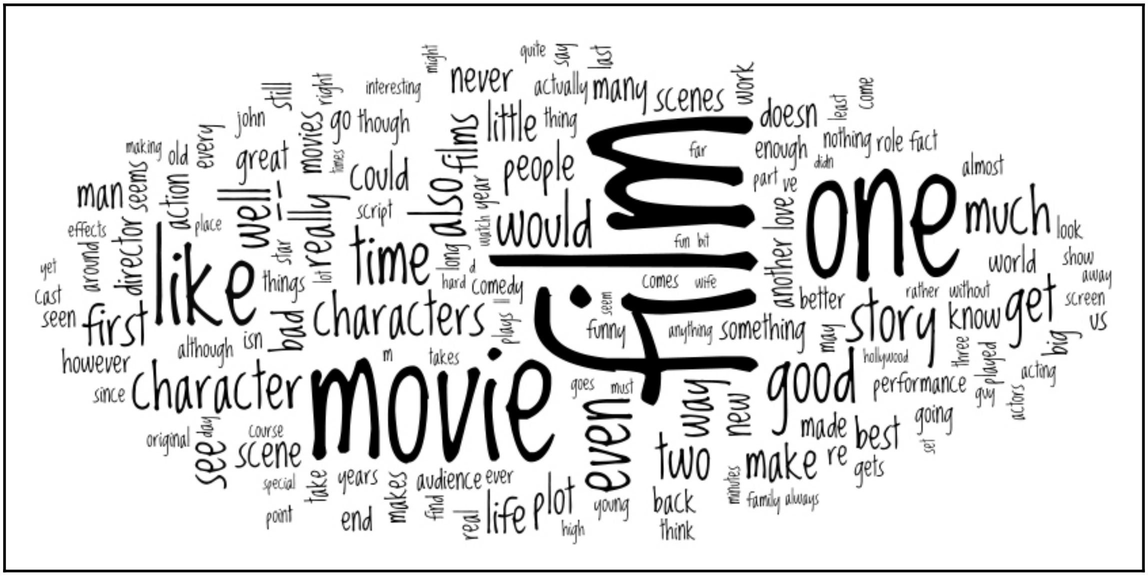

SUNACTIVITY
Median
Wiener
Detrend
200
150
100
50
0
−50
1700
1750
1800
1850
1900
1950
2000
YEAR

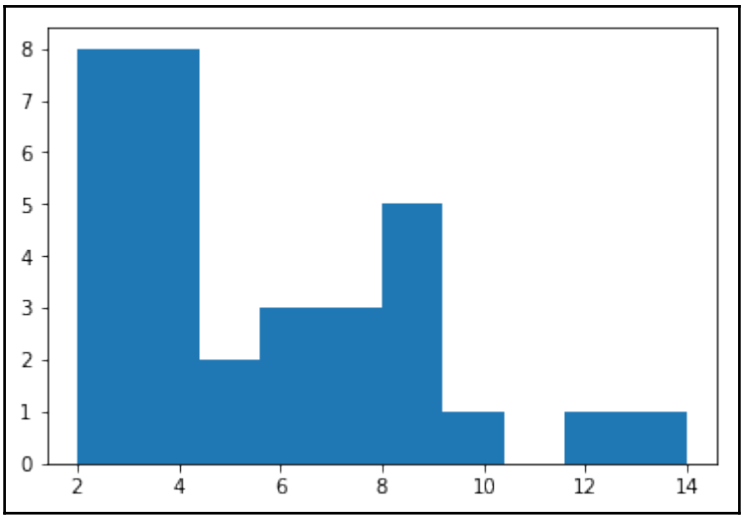
Chapter 9: Analyzing Textual Data and Social Media

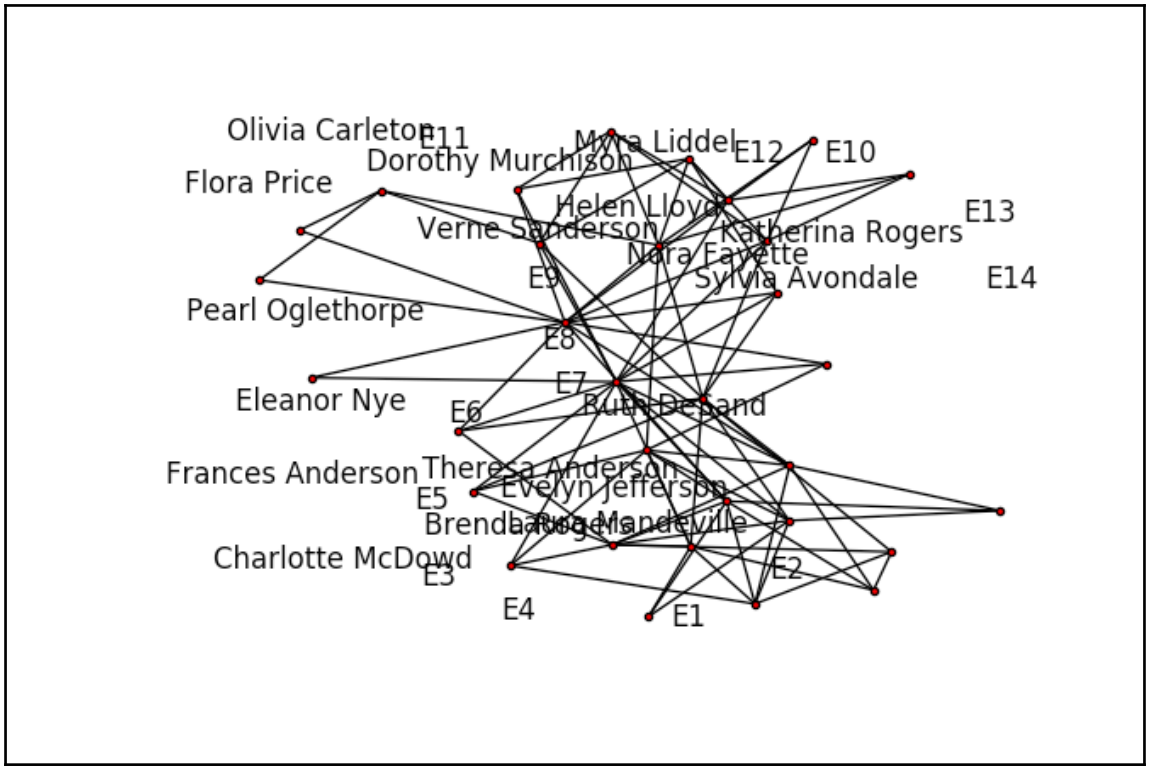



```
len = 7              False : True   =     62.7 : 1.0
len = 6              False : True   =     49.1 : 1.0
len = 1               True : False  =     12.0 : 1.0
len = 2               True : False  =     10.7 : 1.0
len = 5              False : True   =     10.4 : 1.0
```

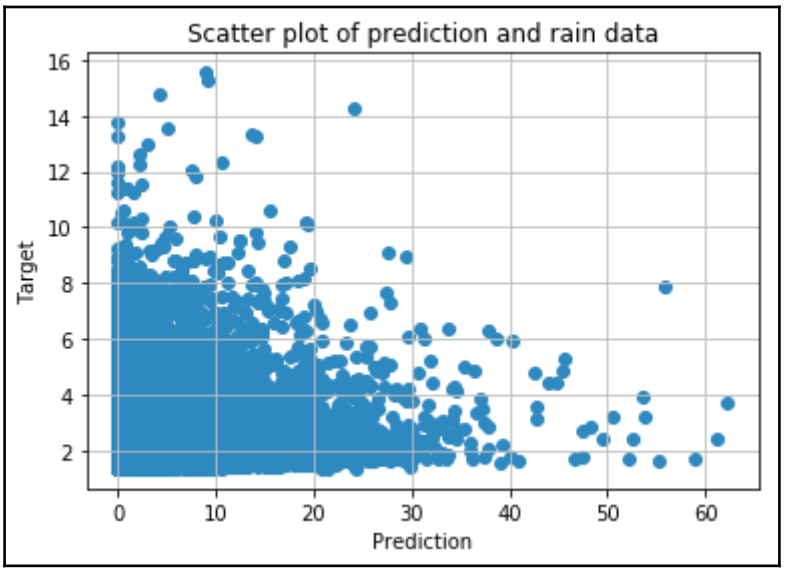
```
    count (wonderful) = 2                pos : neg    =     14.7 : 1.0
   count (outstanding) = 1               pos : neg    =     11.2 : 1.0
         count (bad) = 5                 neg : pos    =     10.8 : 1.0
      count (stupid) = 2                 neg : pos    =     10.8 : 1.0
      count (boring) = 2                 neg : pos    =     10.4 : 1.0
      count (nature) = 2                 pos : neg    =      8.5 : 1.0
   count (different) = 2                 pos : neg    =      8.3 : 1.0
         count (bad) = 6                 neg : pos    =      8.2 : 1.0
   count (apparently) = 2                neg : pos    =      8.0 : 1.0
        count (life) = 5                 pos : neg    =      7.6 : 1.0
```

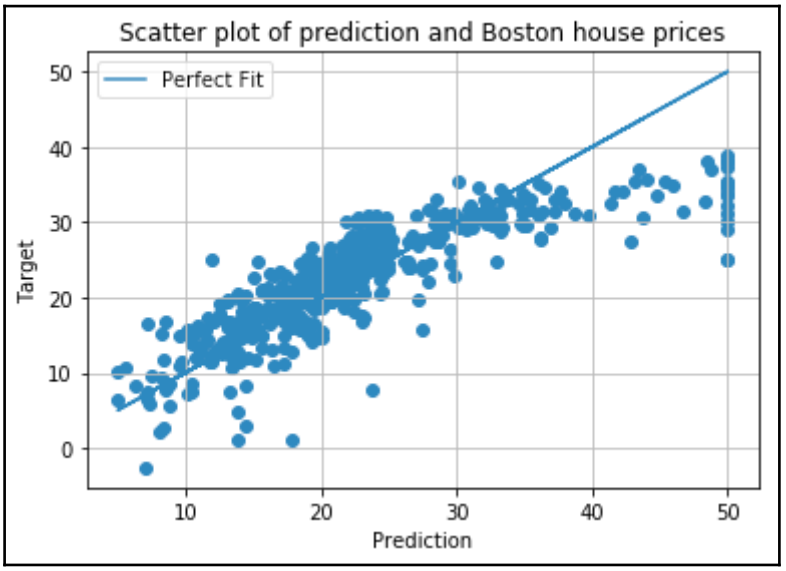

film
movie
one
like
time
characters
character
much
good
story
get
would
even
two
make
also
films
people
could
well
really
great
first
director
scene
see
plot
life
best
way
new
know
never
many
scenes
doesn
little
man
action
movies
something
another
love
better
world
bad
made
back
end
years
audience
think
young
comedy
funny

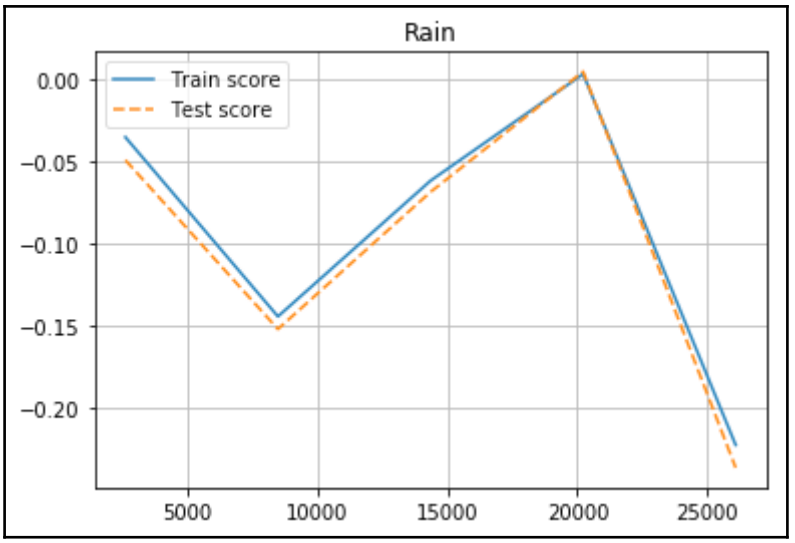

8
7
6
5
4
3
2
1
0
2
4
6
8
10
12
14

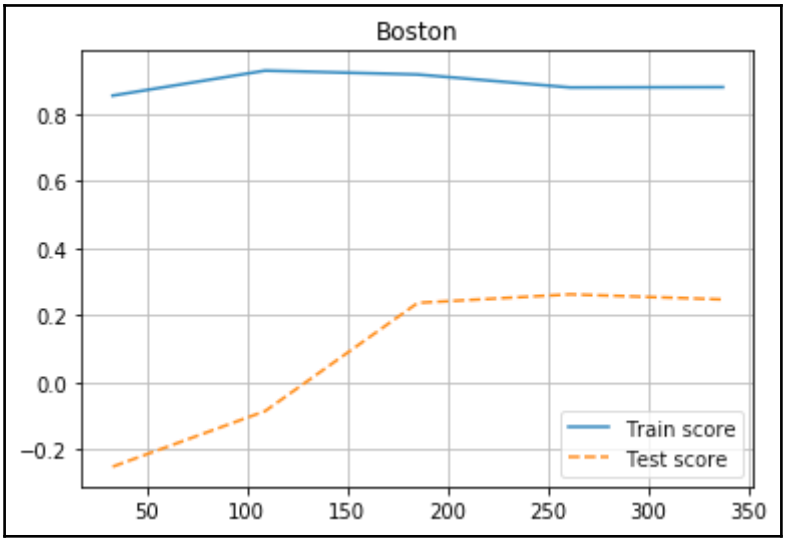

Olivia Carleton
E11
Myra Liddel
E12
E10
Dorothy Murchison
Flora Price
Helen Lloyd
E13
Verne Sanderson
Katherina Rogers
Nora Fayette
E9
Sylvia Avondale
E14
Pearl Oglethorpe
E8
E7
Eleanor Nye
E6
Ruth DeSand
Frances Anderson
Theresa Anderson
E5
Evelyn Jefferson
Brenda Rogers
Mandeville
Charlotte McDowd
E3
E2
E4
E1

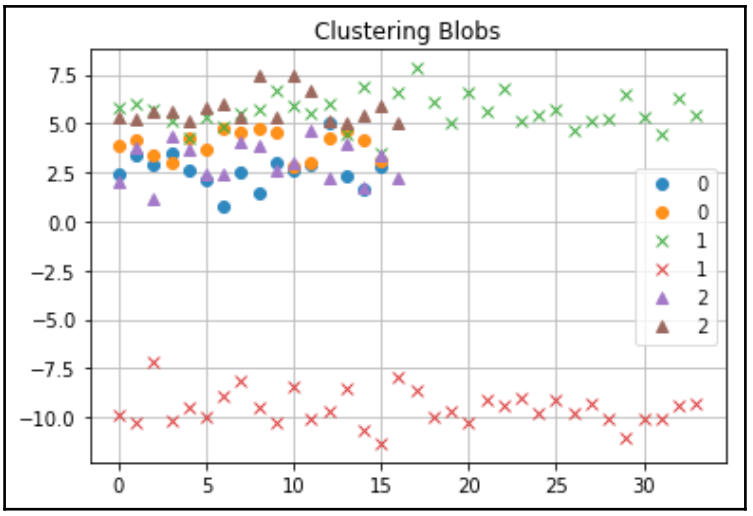
Chapter 10: Predictive Analytics and Machine Learning

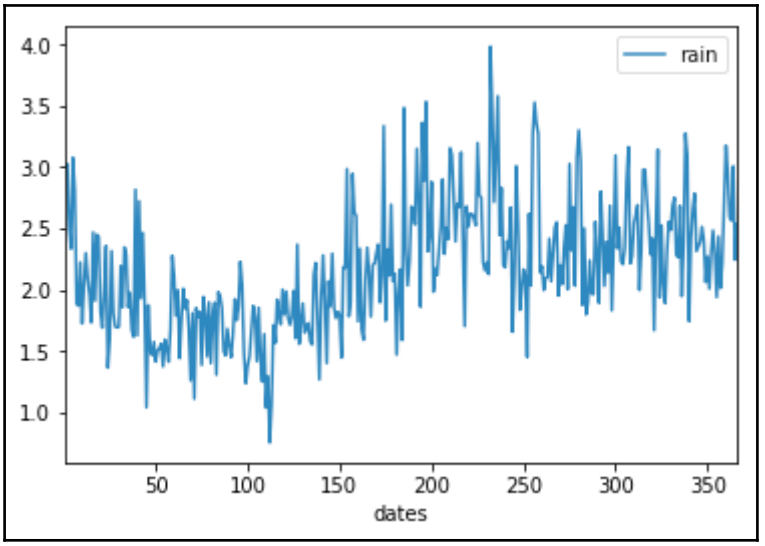


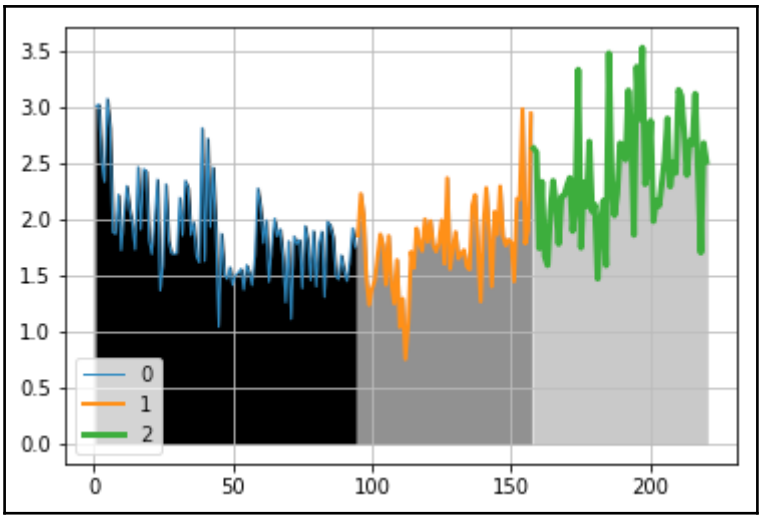


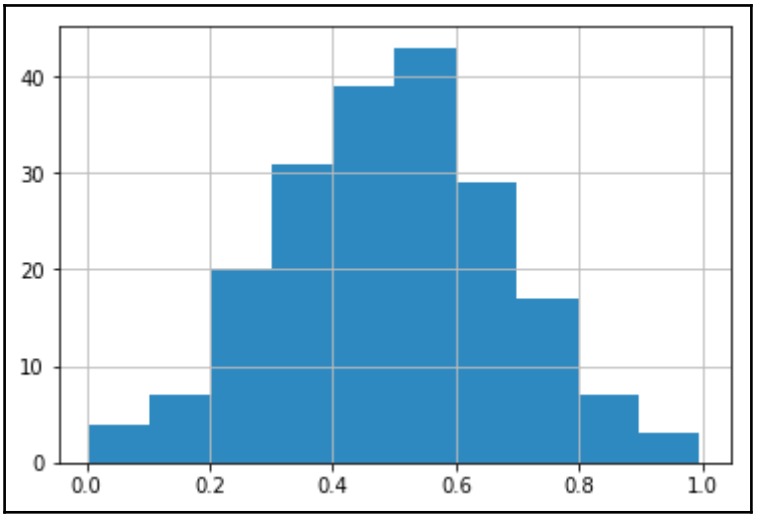

Rain
Train score
Test score
0.00
−0.05
−0.10
−0.15
−0.20
5000
10000
15000
20000
25000

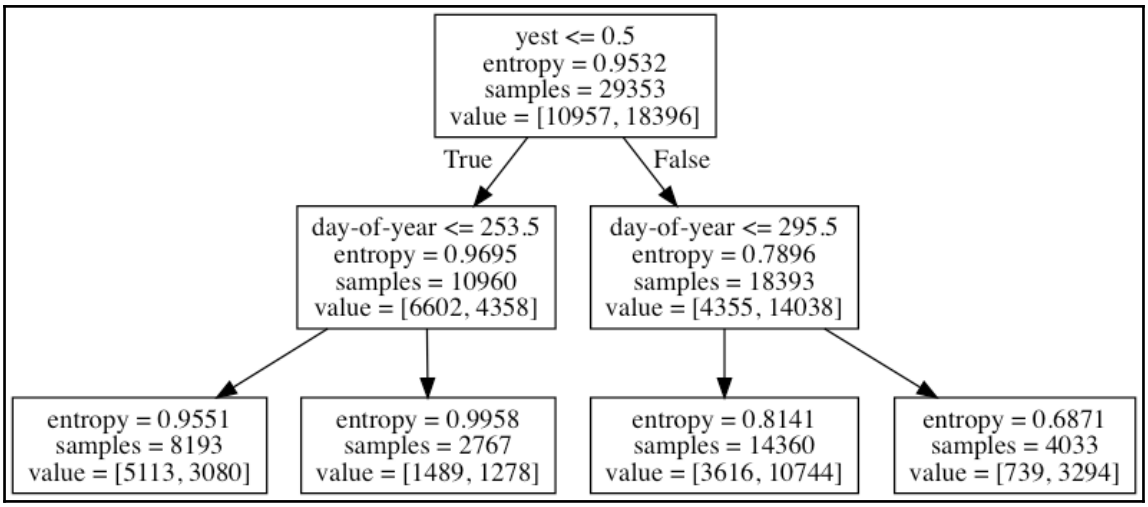

Boston
0.8
0.6
0.4
0.2
0.0
−0.2
50
100
150
200
250
300
350
Train score
Test score


Clustering Blobs
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
0
5
10
15
20
25
30
0
0
1
1
2
2

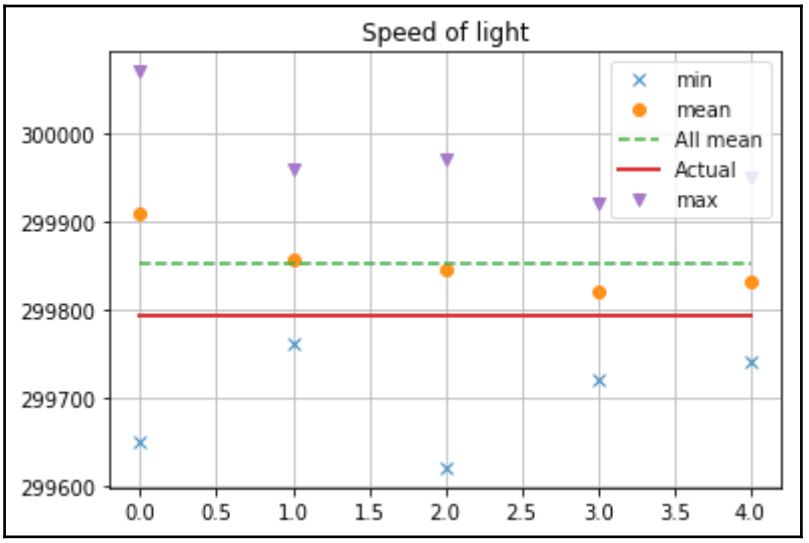

rain
4.0
3.5
3.0
2.5
2.0
1.5
1.0
50
100
150
200
250
300
350
dates

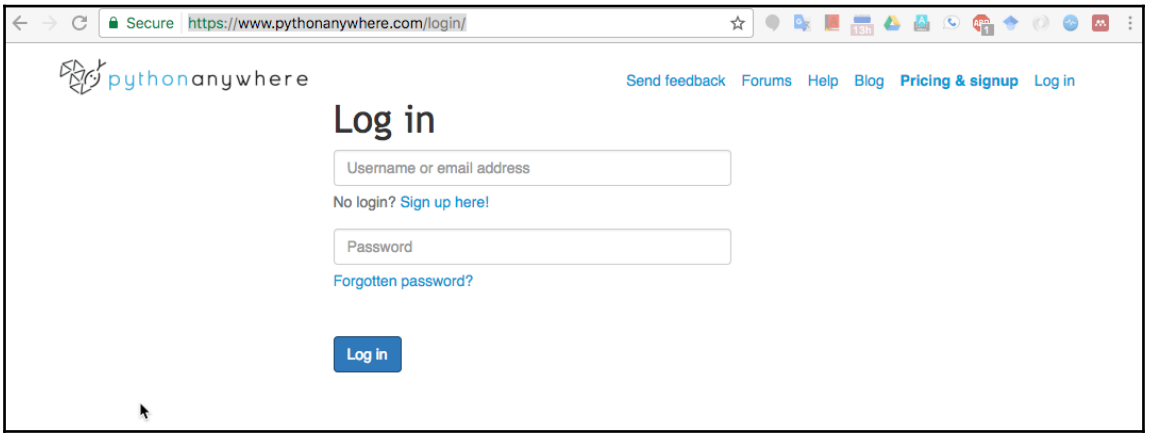

3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
50
100
150
200
0
1
2

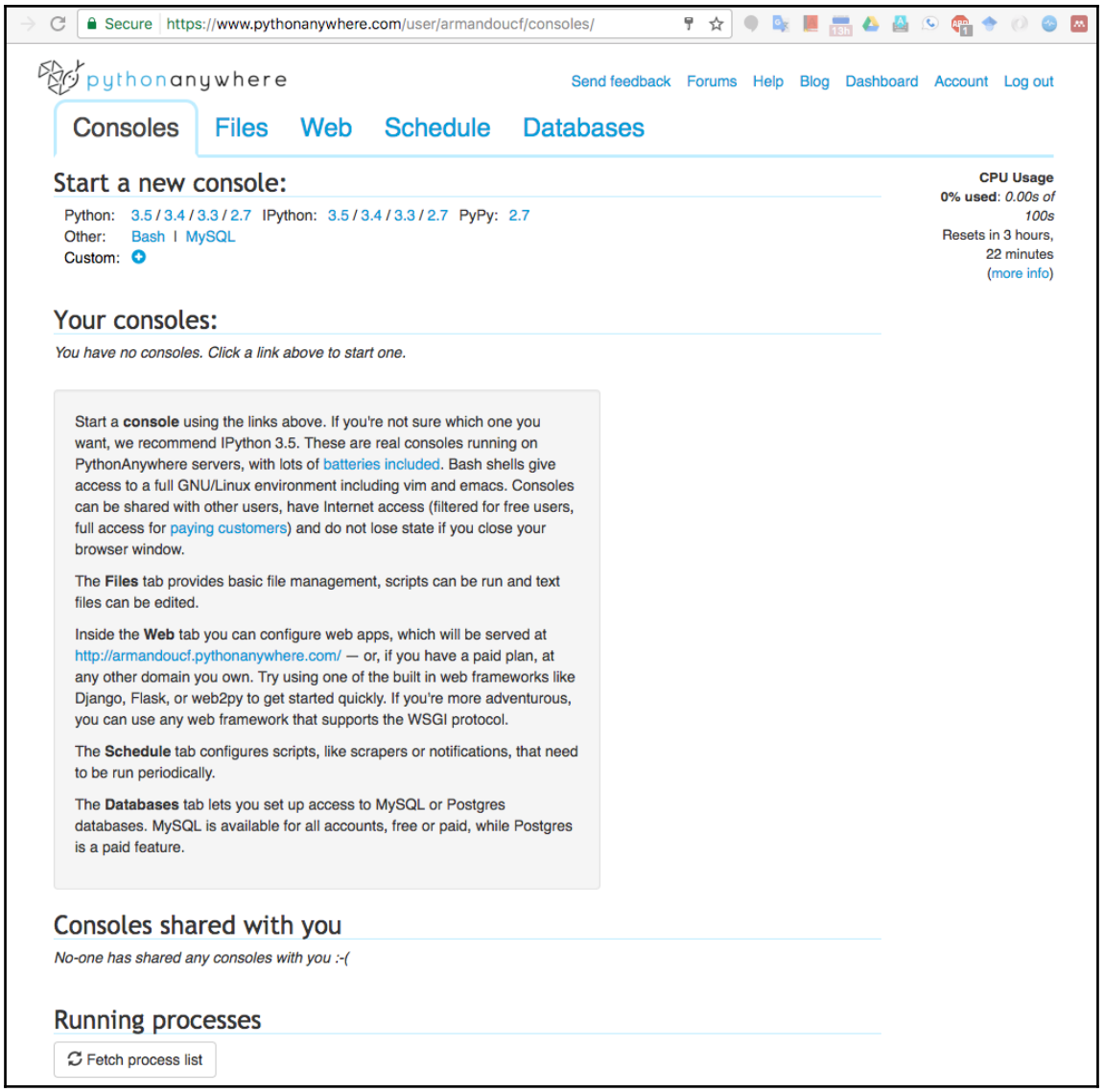

40
30
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0

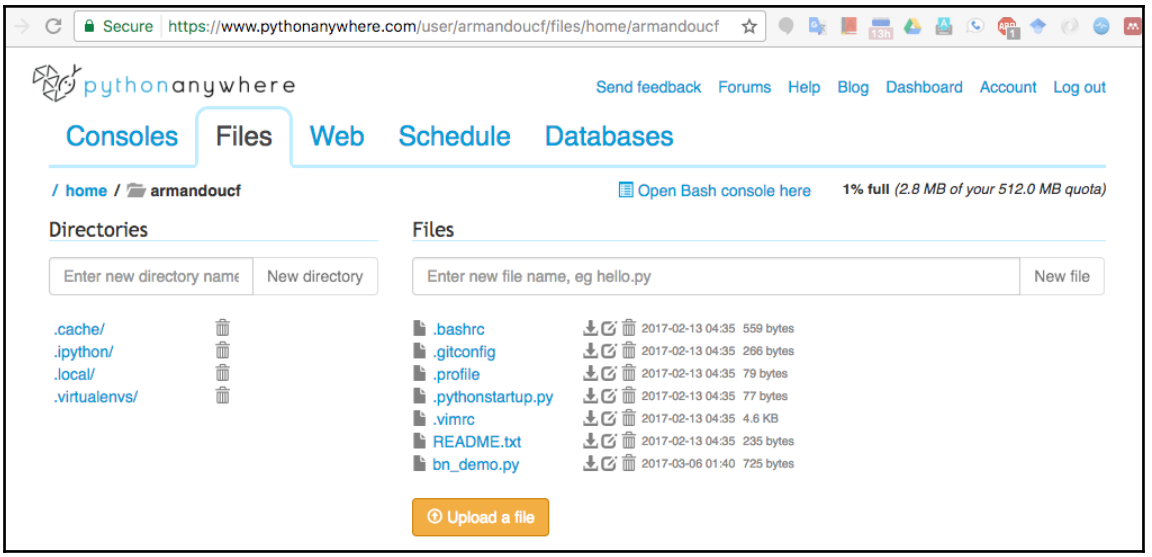

yest <= 0.5
entropy = 0.9532
samples = 29353
value = [10957, 18396]
True
False
day-of-year <= 253.5
entropy = 0.9695
samples = 10960
value = [6602, 4358]
day-of-year <= 295.5
entropy = 0.7896
samples = 18393
value = [4355, 14038]
entropy = 0.9551
samples = 8193
value = [5113, 3080]
entropy = 0.9958
samples = 2767
value = [1489, 1278]
entropy = 0.8141
samples = 14360
value = [3616, 10744]
entropy = 0.6871
samples = 4033
value = [739, 3294]

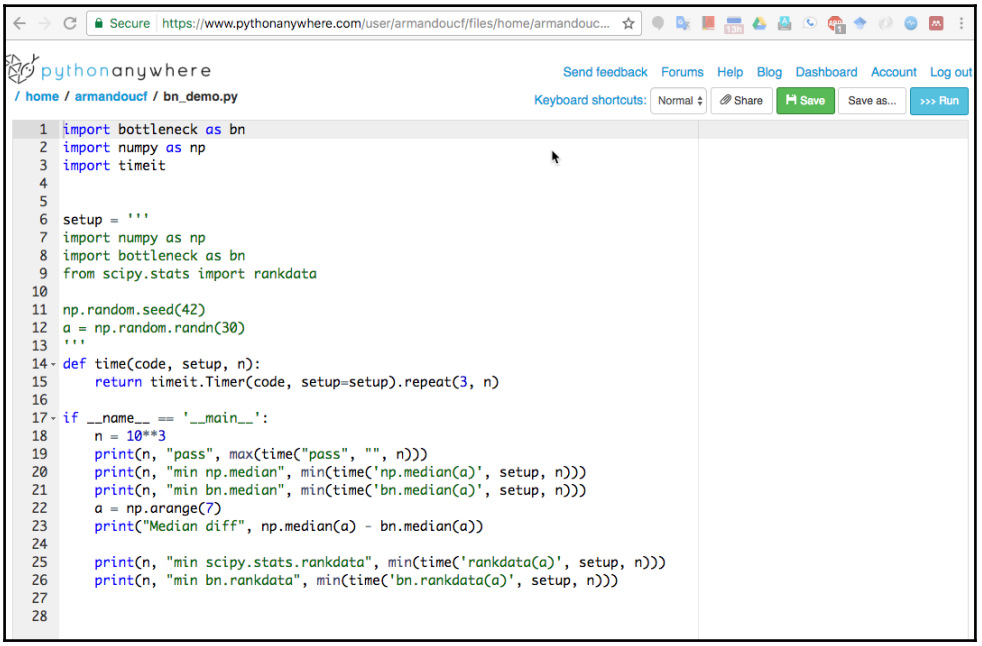
Chapter 11: Environments Outside the Python Ecosystem and Cloud Computing





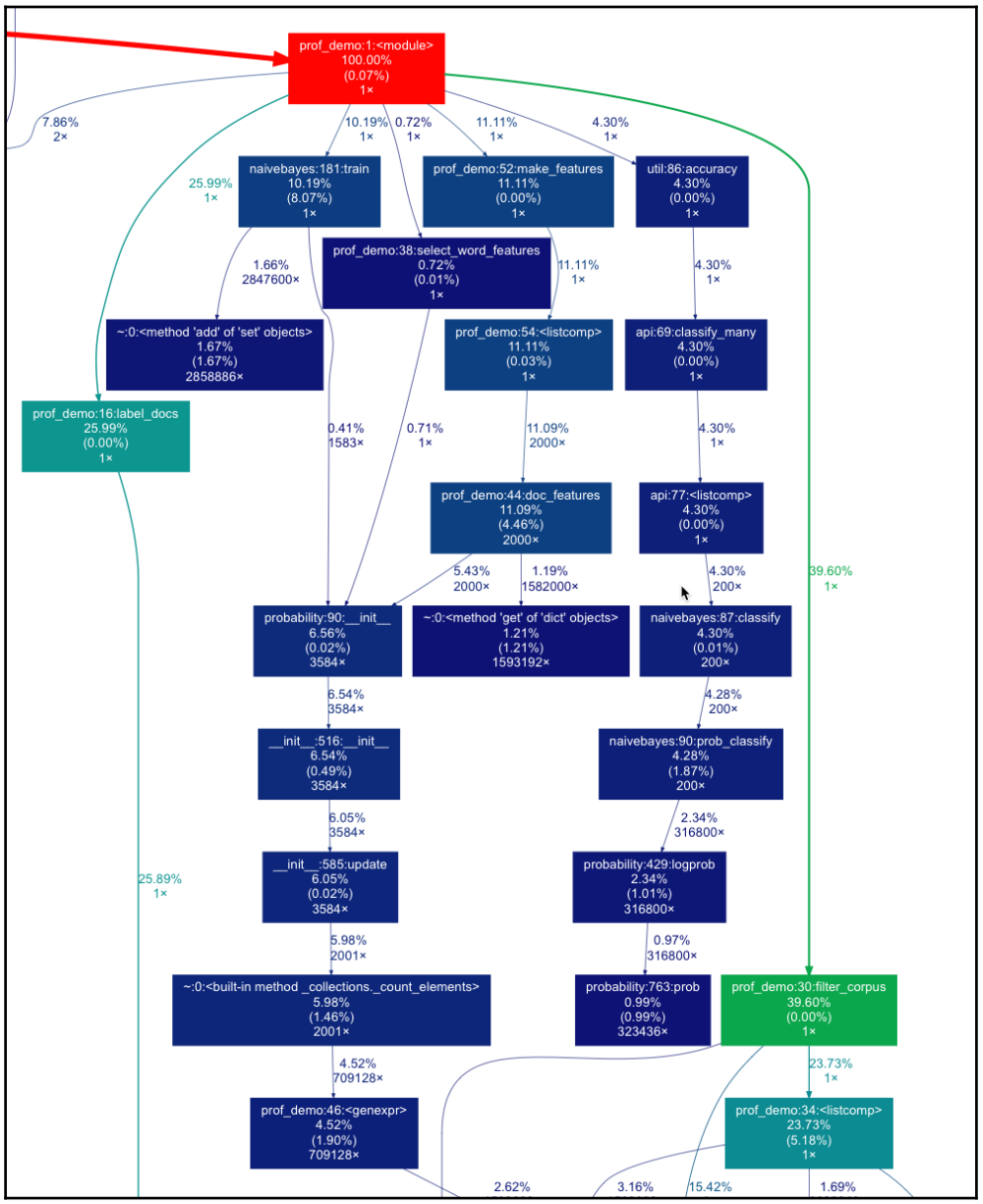

Secure https://www.pythonanywhere.com/user/armandoucf/consoles/
pythonanywhere
Send feedback Forums Help Blog Dashboard Account Log out
Consoles Files Web Schedule Databases
Start a new console:
Python: 3.5 / 3.4 / 3.3 / 2.7 IPython: 3.5 / 3.4 / 3.3 / 2.7 PyPy: 2.7
Other: Bash | MySQL
Custom:
CPU Usage
0% used: 0.00s of 100s
Resets in 3 hours, 22 minutes
(more info)
Your consoles:
You have no consoles. Click a link above to start one.
Start a console using the links above. If you're not sure which one you want, we recommend IPython 3.5. These are real consoles running on PythonAnywhere servers, with lots of batteries included. Bash shells give access to a full GNU/Linux environment including vim and emacs. Consoles can be shared with other users, have Internet access (filtered for free users, full access for paying customers) and do not lose state if you close your browser window.
The Files tab provides basic file management, scripts can be run and text files can be edited.
Inside the Web tab you can configure web apps, which will be served at http://armandoucf.pythonanywhere.com/ — or, if you have a paid plan, at any other domain you own. Try using one of the built in web frameworks like Django, Flask, or web2py to get started quickly. If you're more adventurous, you can use any web framework that supports the WSGI protocol.
The Schedule tab configures scripts, like scrapers or notifications, that need to be run periodically.
The Databases tab lets you set up access to MySQL or Postgres databases. MySQL is available for all accounts, free or paid, while Postgres is a paid feature.
Consoles shared with you
No-one has shared any consoles with you :-(
Running processes
Fetch process list


Secure https://www.pythonanywhere.com/user/armandoucf/files/home/armandoucf
pythonanywhere
Send feedback Forums Help Blog Dashboard Account Log out
Consoles Files Web Schedule Databases
/ home / armandoucf
Open Bash console here
1% full (2.8 MB of your 512.0 MB quota)
Directories
Enter new directory name
New directory
.cache/
.ipython/
.local/
.virtualenvs/
Files
Enter new file name, eg hello.py
New file
.bashrc 2017-02-13 04:35 559 bytes
.gitconfig 2017-02-13 04:35 266 bytes
.profile 2017-02-13 04:35 79 bytes
.pythonstartup.py 2017-02-13 04:35 77 bytes
.vimrc 2017-02-13 04:35 4.6 KB
README.txt 2017-02-13 04:35 235 bytes
bn_demo.py 2017-03-06 01:40 725 bytes
Upload a file

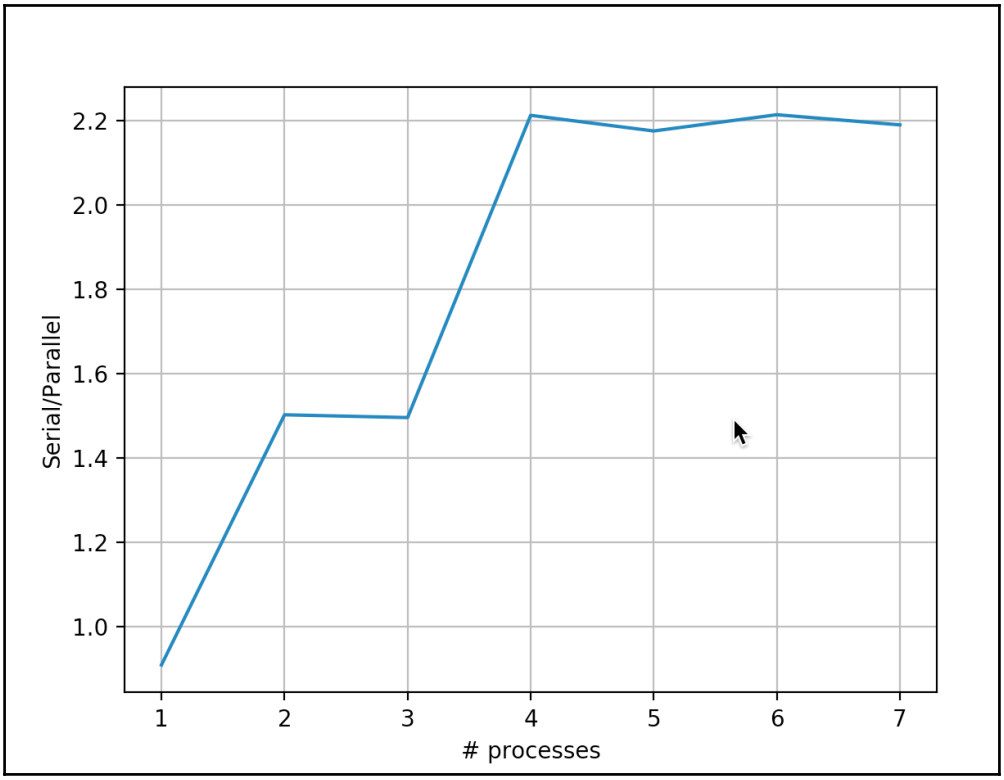

Secure https://www.pythonanywhere.com/user/armandouc...
pythonanywhere
Send feedback Forums Help Blog Dashboard Account Log out
/ home / armandoucf / bn_demo.py
Keyboard shortcuts: Normal Share Save Save as... >>> Run
1 import bottleneck as bn
2 import numpy as np
3 import timeit
4
5
6 setup = '''
7 import numpy as np
8 import bottleneck as bn
9 from scipy.stats import rankdata
10
11 np.random.seed(42)
12 a = np.random.randn(30)
13 '''
14 def time(code, setup, n):
15 return timeit.Timer(code, setup=setup).repeat(3, n)
16
17 if __name__ == '__main__':
18 n = 10**3
19 print(n, "pass", max(time("pass", "", n)))
20 print(n, "min np.median", min(time('np.median(a)', setup, n)))
21 print(n, "min bn.median", min(time('bn.median(a)', setup, n)))
22 a = np.arange(7)
23 print("Median diff", np.median(a) - bn.median(a))
24
25 print(n, "min scipy.stats.rankdata", min(time('rankdata(a)', setup, n)))
26 print(n, "min bn.rankdata", min(time('bn.rankdata(a)', setup, n)))
27
28

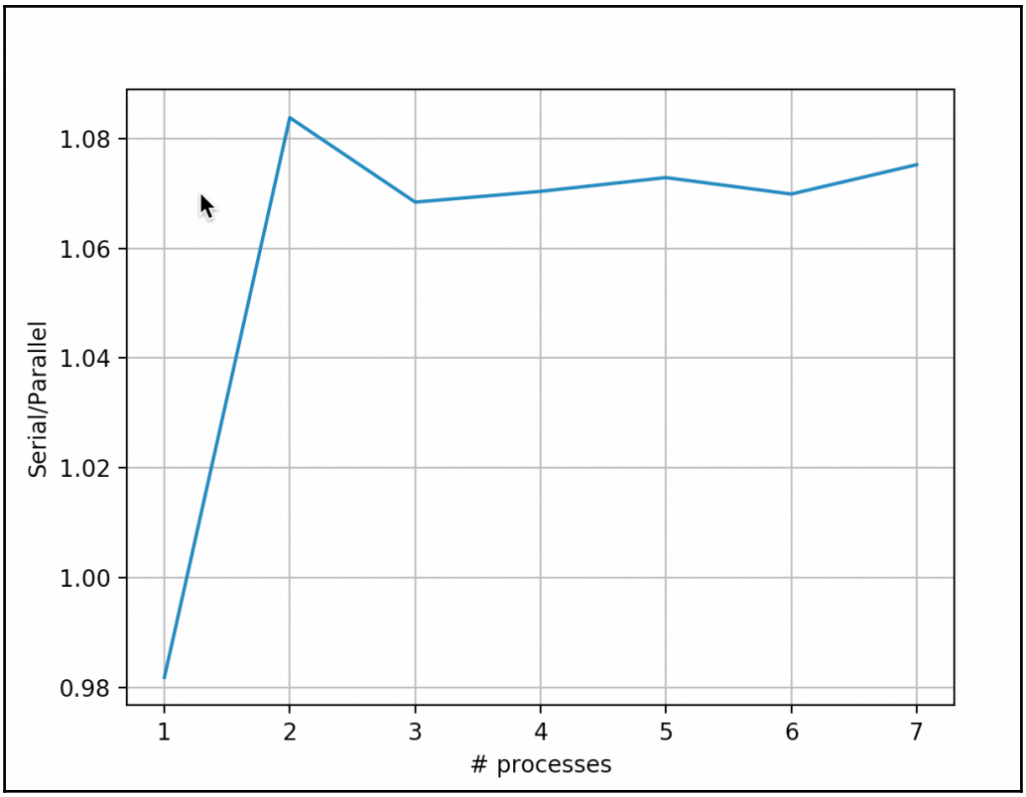

1000 pass 9.993906132876873e-06
1000 min np.median 0.057233922998420894
1000 min bn.median 0.0007603260455653071
Median diff 0.0
1000 min scipy.stats.rankdata 0.11264373897574842
1000 min bn.rankdata 0.001692580059170723
>>>

Chapter 12: Performance Tuning, Profiling, and Concurrency



```
fluid:ch-12 armando$ python3 -m pstats /tmp/stat.prof
Welcome to the profile statistics browser.
/tmp/stat.prof% strip
/tmp/stat.prof% sort time
/tmp/stat.prof% stats 10
Sun Feb  5 18:24:49 2017    /tmp/stat.prof

         30643998 function calls (30123080 primitive calls) in 15.502 seconds

   Ordered by: internal time
   List reduced from 3823 to 10 due to restriction <10>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   319962    2.397    0.000    2.397    0.000 {method 'findall' of '_sre.SRE_Pattern' objects}
        1    1.251    1.251    1.580    1.580 naivebayes.py:181(train)
   319960    1.056    0.000    2.748    0.000 data.py:1114(readline)
  6343280    0.847    0.000    7.338    0.000 util.py:261(iterate_from)
        1    0.803    0.803    3.678    3.678 prof_demo.py:34(<listcomp>)
  3167640    0.741    0.000    0.896    0.000 prof_demo.py:26(isStopWord)
     2000    0.692    0.000    1.719    0.001 prof_demo.py:44(doc_features)
  3167642    0.628    0.000    4.361    0.000 util.py:388(iterate_from)
   371223    0.393    0.000    0.898    0.000 data.py:1353(_read)
3885294/3376152    0.359    0.000    2.696    0.000 {built-in method builtins.len}


/tmp/stat.prof%
```




1.08
1.06
1.04
1.02
1.00
0.98
Serial/Parallel
1
2
3
4
5
6
7
processes