I python
% time
% timeit
% ls
% prun
?
% rm
% cd
% mkdir

df. argmax
df. index
df. groupby( )
df. value_counts
df. sort_values (ascending = , [ ]
df. plot (kind = 'h bar'
df. drop
df. dropna
df. read_csv( )?
pd. crosstab (rows, columns)
pd. pivot

Series. values( ) = np.array

df. index. weekday / month / day
df. merge / df. join?

.str. contains
.str. len
.str. replace

np. reshape ([rows, colums])
np. mean
np. max , np. min
np. std.
np. sum
np. size , np. shape
np [rows, colums]
np [first : last : step]
np. indices
np. transpose (array)
np random. shuffle
np invert
np zeros, np. ones
np count_nonzero (value)
np random . range

100 mensen

10 ziek   10 juist   9 juist
90 niet ziek

echt
ziek   niet ziek
Test ziek   9   9
niet ziek   1   81

1000

echt
ziek   niet ziek
Test ziek   9
niet ziek   99

990 - 99   891

% ls magic

Shellmet ! (pwd, ls, cd)

? access info

- * - wildcard options

---

dot $\sum_i (s_i \cdot S_i)$

- evaluation

Classification → accuracy

recall $TP/(TP+FN)$

Precision $TP/(FP+FP)$

RMSE : regression

NP → dir(np)

np. empty/zeros → array

np. shape/ndim/size → [i]

↓ [cols] #

np.info(np.ndarray.dtype)

math	aggregate	
. exp → ^{2}	. sum	. min
. sqrt → √	. std	. max

np. arange (start, stop, step)

[: : -1] → backwards

[: : 2] → from 0 with step 2

a[2] → el. at 2nd index

a[1,2] → 2nd row, 3rd col

a[0:2] → items at 0 and 1

- boolean indexing

a[a<2] → alles in a kleiner dan 1

- fancy

b[[1,0,1,2],[0, 1,0,1]]

↳ (1,0), (0,1), (1,0), (2,1)

- transpose

np. T

np.transpose() [1] / [2] → add rows/cols

np. concatenate()

arr[1]=4 (maak 4 van index 1)

np. add(arr1, arr2) → elementwise

(arr, 4) → 4 bij elk element

PD → dir(pd) → help(pd.Series.loc)

pd.read_csv('file.csv', header, rows)

df['a'] → get one element

df[1:] → get subset

Position → df.iloc[[0],[0]] → select

by label → df.loc[[0],['Country']]

by index → df.ix[2] → single row/sub

→ df.ix[:,['Country']]

bool select

s[~(s>1)] → s where s is not >

s[(s<-1)|(s>2)] → values under -1 & above 2

sort_index()

sort_values(by...)

- Statistics

df.describe → summary

df.mean → mean of all cols

df.corr → corr between cols

df.count → value count col

df.min/max/mean/median/std

- cleaning

df.colums['a','b','c'] → rename cols

pd.isnull() check null, bool array

df.dropna() → drop row nan values

df.fillna(x) → fill nan with x

(s.mean()) → fillna with mean

s.replace([1,3],[one, three])

↳ replace all 1 → one 3 → three

s['col'].str.lower()

str.replace('vb', '')

delete all met ...

s[~s['col'].str.contains('...')]

- filter, sort, group

df[df[col] > 1] → rows w/ col val > 1

df[(df[col] > 1)&(df[col]<2)]

df. sort_values(col, ascending)

df. groupby(col)

([col1], [col2])

df.pivot_table(index=col, values=col2,3, aggr)

→ group by col, aggfunc on 2,3

- Join / combine

df1.append(df2) → add row to end df1

. concat([df1, df2], axis=1)

→ add cols to end df1

df.join(df2, on=col1, how=inner)

- View, inspect

df.head

df.tail

df.shape

df.info

df.value_counts

len(.unique()) → unieke len

vtg → kur['Party'] == 'pvdd'

- np slices and indices are views not copies so changing them changes original array, use .copy() for a copy.
- Broadcasting rules:
1. If the two arrays differ in their number of dimensions the shape of the one with fewer dimensions is padded with on it's leading (left) side.
2. If the shape of the two arrays does not match in any dimension, the array with shape equal to one in that dimen is stretched to match the other shape.
3. If in any dimension the sizes disagree and ne is equal to 1, an error is raised.
- x[::-1] all elements, reversed
- Count amount of True entries in a Boolean array w np.count_nonzero (True = 1, False = 0)
- Quickly check any/all values are True with np.any / np.a
- Data science is: Based on patterns in data predicting a value or new unseen instances of data (classification/regression). Steps: interactin with outside world, preperation, transformation, modeling and computat presentation. Data Science is the study of the generalizable extraction of Knowledge from data. Precision $= \frac{TP}{TP+FP}$ Recall $= \frac{TP}{TP+FN}$
$F_1 = 2 * \frac{Precision \cdot Recall}{Precision + Recall}$
- np.nditer to iterate through numpy arrays
.value_counts() .index.valeres .str. ('[^\w\s]
x[~x.Y.str.contains(" ", na = False)]
.value_counts().sort_index() x.loc[x['x'] == '']
normalize = True ·100 voor percentages

P N
P
N

raw column

x[1,2]

x[:2, :3] #tworows, three colums

x[:3, ::2] #all raws, every other column

x[::1, ::-1] reverse

x[:, 0] first colum

ints divisible by 3
x[x % 3 == 0]

x[0] first row

normalize
((x-x.mean())/x.std().head
.., ((x - x.mean()) /
x.std()).std()

masking
df[(df >0,3)&(df<0,8)]

df.loc = explicit indexing

df.iloc = pythonstyle indexing

#sort iris by heigth then width

iris.sort_values(['height', 'width', ascending=[False, True])

# max length species
iris.groupby('species')['length'].max()

x.sort() of x.argsort
sort indices

k nearest 2:

```
k = 2    z = np.argpartition(y, k+1, axis=1)
for i in range(x.shape[0]):
    for j in z[1,:k+1]:
        plt.plot(zip(x[j], x[i]), color='black')
```

```
def shape (n = len(L), 1):
    return L.reshape(np.ma.count(L),1)
```

alltokens which occupy 2u char
CS = pd.Series (C).sort_values(ascending=False)
C=Counter(text.split())
(S[(S*(S.index.str.len()== 2u].index

help(len) of len?
↑
dubbel voor source code

*Warning? every object that en with warning

str.*find*? string method the contains find

%past %cpaste %magic %lsmagic %run

de ide rij pakken df.iloc[i]/[-1]

df.groupby(0)[1].count().sort_values().tail()

df.crosstab(df[column], df[column], margins = True)

import matplotlib.pyplot as plt

df.index.str.lower().str.strip().str.replace(" ", "").sort_values()

df['RMSE'] = np.sqrt((df. colom - df. colom ** 2)

df[df.kolom > x].kolom2.value_counts().sort_values().plot(kind=barh)

df.describe()

aantal vrouwen in party =
df['party'].loc[(df['geslacht'] ==
'vrouw')].unique()
#alle voornamen met mo
df[['voornaam', 'party']].loc
[(df['voornaam'].str.startswith('mo'))

df.columns[~df.columns.str.contains('low') | df.columns.str.contains('high')]

Data Science cheat sheet

NP

% lsmagic = all magic commands

- x[:2, -3:] = last 3 columns first 2 rows | x + np.arange(len(x))
- x.reshape(12,) = make one dimensional | x[[0,1,2,3],[0,1,2,3]] = fancy index diagonal (4,)
- (x - x.mean())**2 sqr dif. each elem from mean of x
- np.sqrt(((x - x.mean())**2).mean()) == x.std() = ~~variance~~ hand wr. std vs function
- x[x % 3 == 0] = apply bool. mask
- x2[[0,1,2],[0,1,2]] diagonal (fancy index)
- big_arr = np.random.randint(10**3, size=10**7)

% timeit sum([x^2 for x in big_arr])

% timeit sum(big_arr**2)

x[:, -1] = last column 2D array | x[x % n == 0] | x.mean(axis=0) ← 0 = column, 1 = row

PD

dubbel = df.index.value_counts()

dubbel[dubbel>1]

df.sort_index()

df.index.str.lower.str.strip().str.replace("", '').str.replace(" ", '').sort_values

df[~df.index.str.lower().str.contains('string')].head(10)

df.describe

df.inkomen.sort_values.plot();

df[df.inkomen>50000].gemeente.value_counts().sort_values().plot(kind='barh');

= gemeentes met scholen met inkomen > 50.000, gesorteerd op hoeveelheid in hor. bar plot

top_minis = kur[kur['partij'] == 'PVDA'].Ministerie.value_counts()

top_minis[top_minis >= 5].plot(kind='barh');

df.dropna(subset=['partij'])

deelvragen = kur.Vraag.str.count('\?')

no_outliers = deelvragen[deelvragen <= 56]

no_outliers.plot(kind='hist');

np.mean(no_outl)

median = statistics.median(no_outl)

modus = Counter(no_outliers).most_common(1)[0][0]

corr1 = no_outl.corr(kur.Vraag.str.len())

corr2 = no_outl.corr(kur.Antwoord.str.len())

df.sort_values(['sepal_len', 'sepal_wid'], asc=[False, True])

df.groupby('species')['sepal_len'].max()

df.query('sepal_len > 5 and sepal_wid < np.sqrt(5)')

C = Counter(text.split())

CS = pd.series(C).sort_values(asc=False)

CS[CS[CS.index.str.len() == 21].index

import numpy
pandas
matplotlib.pyplot as plt
Seaborn as sn
Vandean
array
OS
math
%matplotlib inline

np.random.seed(42)
randint (10, size=6) 1d
randint (10, size=(3,4)) 2d
(10, size=(4,4,6)) 3d

np.concatenate ([X, Y])
np.vstack
Hstack
np.split [3,5] = tot 3 / 4 5 / van 6
np.vsplit [X, [3]] vanaf rij 3
Hsplit

indexing:
1d = X[0]
2d = X[0,0] / X[::-1] / X[:: -2] / X[:, 2]
reverse — van 5 reverse — alle rijen tot kolom

reshaping X [np.newaxis, :] = row vector
X [:, np.newaxis] = column vector

magic: %ls - lh = directory
%rm = remove
%% = double line

Computation = 2**x - X%2 - X//2 -
np.log - np.log2 - np.log10(x)

Broadcasting: rule 1 = padded to left
2 = 1 dimension array stretched
3 = if dimension disagree = error
m+A[:, np.newaxis].shape = right padding

x = np.arange(1,5)
np.add.reduce (X) = returns it on all elements
np.multiply.outer(X)
np.linspace (start, eind, spreiding)
np.arange (start, einde, stap)

np.sum (L)
L.min (axis=0/1)
L.max

a = np.random.randomState (42)
a.randint (10, size=(3,4)) → a < 6 - a>6 -

acces simple = X[0]
slicing = X[:5]
masking = X[X>5]
Fancy indexing = [X[3], X[5], X[6]]
Fancy + simple = [X[2, [2,0,1]]
rij kolom
Fancy + slicing = [X[:2, [1,3,2]]
tot rij + kolommen
Fancy + masking = mask = np.array ([0,1,0])
X[row[:, np.newaxis], mask]

np.sort (X, axis=0/1)
np.argsort (X) = values of position
np.partition (X,3) = 3 smallest values + remaining

Theorie!
Precision = # of predictions for class C correct
Recall = # instances of C predicted to be C } - TP/(TP+FN)
accuracy = # of all predictions being correct } - TP/(TP+FP)

accuracy = 95% n = 10000

	heeft diis	heeft niet	
pos +	95	495	590
neg +	5	9405	9410
	100	9900	10000

Precision
95/(95+495) = 0,16

Pandas = good for data analysis and modeling without use of R
df = pd.read_csv ('...'), names = ['kolom', 'namen', 'invullen'])
df = pd.read_html ('...')[1]
%ls -l ../Data/*.csv
% cat ../Data/foo.csv → df = pd.read_csv ('../Data/foo.csv')

1 plot samples in dataset = iris.species.value_counts().plot(kind='barh');
2 create dataframe + numerical data = x and y = X, y = iris.drop ('species', axis=1), iris.species
3 z-normalize by substracting mean by std = ((X-X.mean()) / X.std()).head(), '', ((X-X.mean()) / X.std()).std ()
4 Sort iris by column and row + inverse = iris.sort_values (['...', '...'], ascending=[False, True])
5 maximal lengths for each species = iris.groupby ('species')['length'].max()
6 larger than 5 and smaller than sqr root 5 = a = np.sqrt(5) - iris.query ('x > 5 & y <@a')
7 tokenize the text by split, count token, show head = C = counter(text.split())
8 Find all tokens which have ≥ 24 = CS[CS*CS.index.str.len() == 24].index
CS = pd.Series(C).sort_values(ascending = false)
CS.head (10)
9 what percentage of all unique words occur once = CS[CS == 1].sum()/CS.count(), ((CS==1).mean, CS[CS==1].sum / CS.sum,
(CS==1).sum / CS.sum()
10 make histogram which indicates for each number i occur how many tokens in text occur i times
pd.Series (Counter (CS.values)).plot (kind='barh'); CS.value_counts().sort_index().plot (kind='bar')

extra vraag: geef land met meeste dagleden:
bd = pd.read_html ('...')[1], bd.groupby [0][1].count().sort_values().tail(1)

data = [{'a':i, 'b':2*i}
for i in range(4)]
pd.Dataframe (data)
pd.Dataframe (np.random.rand(3,
columns=['1', '2', '3'],
index = ['jo', 'ma', 'bo'])

Series

data = pd.Series ([1,2,3], index=['a','b','c','d'], dict = {'Ams' : 100
'UTC' : 50} } pop = pd.Series (Dict)
pd.Series (5, index=[3,6,9])
pd.Series ({2:'a', 3:'b'}) → index = [3,1]

Pandas

states = pd.Dataframe ({'pop': pop, 'area' : area})
states

X.size — x_1 & x_2 = similarity check
X.shape — x_1 | x_2 = add without same/union
X.ndim — x_1 ^ x_2 = symmetric difference

indexing data ['b'], 'a' in data, list (data.items())
slicing data ['a':'b'], data[0:3], data [(data >0) & (data <3)], data [['d', 'b']] data.loc[1]
masking — Fancy

data.values = alle waardes
data.T = transpose

data.loc [data.x > 100, ['pop', 'area']] = masking + fancy
A.add (B, fill_value = 0)

Slicing in 2d = pop [('ams', 2010) : ('UTC', 2000)]
hard = select all values from 2010 → pop [[i for i in pop.index if i[1] == 2010]] → pd.MultiIndex.from_tuples(index

concatenate
pd.concat ([ser1, ser2], axis = 0/1)

categories of join = 1-to-1
many-to-one
many-to-many
df3 = pd.merge (df1, df2)
→ need key column
→ choose by pd.merge(x, y, on='...')
(x, y, left_on='...',
right_on='...')

pd.merge(x, y, how = 'outer'
'inner'
'left'
'right'

Lees de vragen goed! Kijk of je iets moet plotten!   iris = ... hoeft niet   verzameling: set()   a.tab → tel

NUMPY

1D array: x2.reshape((12,7))

```
X1.std() == np.sqrt(((x1 - x1.mean())**2).mean())
```

(x1 - x1.mean())**2 = squared difference; ((x1 - x1.mean())**2).mean() = variance

diagonaal: x2[[0,1,2], [0,1,2]]

```
%timeit sum([x**2 for x in big_array])
%timeit sum(big_array**2)
```

– sum of squares in an array –

homogeen vs heterogeen. size = elementen   ndim

PANDAS

index_col = 'kolomnaam' die je als index wil

- unique() of set()

str.strip() haalt whitespace aan begin/eind weg
str.contains('str')
df[~df....]
df.describe() → statistiek data

→ van elke waarde het gem. v die waarde in die kolom aftrekken (normaliseren)
mean = cito.mean()
cito[mean.index] - mean

cito.gemeente[cito.inkomen > 50000].value_counts() /
- sort_values().plot(kind='barh')

→ in welke gem. staan scholen waar inkomen meer dan 50000
plot aantal scholen per gemeente

df.drop(colums='naam') of ('naam', axis=1)

→ Z-normalize data by subtracting mean of column from each cell and divide by standard dev of each column
((X - X.mean())/X.std()).head()

→ also show std of each col of the Z-normalized data
- std() → moet gelijk zijn aan 1

→ Sort iris first by S-L & then by S-W. Sort in reverse order (but index opgepend)
iris.sort_values(['S_L', 'S_W'], ascending=[False, True])

→ max S_L for each species
iris.groupby('species').sepal_length.max()

→ Find all tokens in text that occupy 24 char in total
printe
CS[CS.index.str.len() * CS.~~values~~ == 24].index
(CS = pd.series(Counter(text.split())))

→ What % of all unique words in text occurs once
CS[CS == 1].sum() / CS.count()
of len(CS.index.unique()[CS.values == 1]) / len(CS.index.unique())

→ What % of all words in text occurs once
CS[CS == 1].sum() / CS.sum()
of len(CS.index.unique()[CS.values == 1]) / len(text.split())

→ Make a histogram (bar plot) which indicates for each number i of occurrances how many tokens occur i times
CS.value_counts().plot(kind='bar')

→ Land met meeste bondsdagleden + aantal leden
bd[0].value_counts().head(1)
of bd.groupby(0)[1].count().sort_values().tail(1)
of winnaar = bd.groupby(0)[1].count()
winnaar.argmax(), winnaar.max()

→ welke partij heeft grootste aantal leden 1 deelstaat
bd.groupby(0)[2].value_counts().sort_values().tail(1)

IPYTHON

%ls files in directory
%cp file1 file2
%rm no-good.txt
} %magic, %lsmagic

pwd print work-directory
cd naam / change dir
mkdir naam make dir
mv ../.txt ./ move one dir up

inspect resources: %time / %timeit
- verschil → %time ? / %timeit ?

datatype of var. X → x? inspect 1st line of output
get documentation: help(..)
print(...)
suppress output ;
%debug interactief
%mode hoe exceptions zien

THEORY

precision: $\frac{TP}{TP+FP}$ accuracy: $\frac{TP+TN}{totaal}$ recall: $\frac{TP}{TP+FN}$

	T	F	
T	TP 95	FP 495	590
F	FN 5	TN 9405	9410
	100	9900	10 000

accuracy 95%
ziekte 1/100

precision = (0.01 × 0.95) / ((0.01 × 0.95) + (0.99 × 0.05)) = 0,16

xml / json: cel + lijst (als er niet evenveel kolommen zijn → spreadsheet lenen)

Teken spreadsheet:
- insert / delete / update
- andere view (sterug / draaing)
- combineren v. sheets
- aggregate / correlate

dot prod: A.dot(B)
sum(a[i] * b[i] for i in range(len(a)))

no births p/year p/gender
names.groupby(['year', 'sex'])['births'].sum().unstack()

numpy hw wk2

- create array(n, 0):
  return np.arange(0, n*0, 0)   eerste n getallen deelbaar door 0
- add_per_column(L)
  for i in range(L.shape[1]):   tel i op voor item in ide kolom
  L[:, i] = L[:, i] + i
  return L
- add_per_row(L)
  tp = np.transpose(L)
  return add_per_column(tp).transpose()
- mean_per_column(L)
  return np.mean(L, axis=0)

alles reversed: x[::-1, ::-1]
1D → 2D   x.reshape((1,3)) of x[np.newaxis, :]
x1, x2, x3 = np.split(x, [3,5]) → [:3] [3:5] [5:]
np.sum of L.sum()
broadcasting – de regels verbieden het om x1 naar de shape van x2 te krijgen.
how many vals less than zero: np.sum(x<6)
'' in each row : np.sum(x<6, axis=1)
np.any() / np.all()
combined indexing: x[2, [2, 0, 1]] = x[2,2], x[2,0], x[2,1]
get name from last row   data[-1]['name']
data.loc expliciet → = indices
data.iloc impliciet
data.ix hybride vorm
data.T transpose

test zegt dat ik t heb
heb ik het ook

Yulia Valencia

Numpy

1 Last 3 columns of first 2 rows: X[:2, -3:]

2 Change each element in x1 by its squared difference from the mean of x1.
(x1 - x1.mean())**2 = variance

3 standard deviation = sqrt of variance (np.sqrt(variance) == x1.std()

4 1D en 2D arrays vanwege broadcast regels niet samen te voegen, 1D kan niet naar 2D

5 boolean mask to x1 and reduce x to ints divisible by 3: x1[x1 %3 ==0]

6 Fancy indexing for diagonal of x2
x2[[0,1,2], [0,1,2]]

TP / TP + FP
TN / TN + FN

IPython
%ls -lh a= working directory b= size
%rm = remove
%time, timeit, prun
~~% cp = copy previous, new~~

Pandas

pd.read_excel(" ", index_col=' ')
dimensies = .shape
test voor dubbelen = len(set(")) = len(")
cito.index.value_counts() telt hoevaak elk elem in cito.index voorkomt
>1 betekent dubbel (dubbel = cito.index.value_counts() | dubbel[dubbel>1]
sort_index()
str.contains
~ = negation
(cito[M.index] - M.tail()
M = cito.mean()
Cito[cito.inkomen>50000].gemeente.value_counts() 7,2
(condition) (series)

.sort_values() ascending=[6,8
.groupby .max() /.min()
a = np.sqrt(5)
iris.query('sepal_length > 5 and sepal_width < @a')
C = Counter(text.split()) = text splitten en tellen 6,2 + 8,7c
pd.Series(c)

14,9

10.10: 10.60.10.10 60 0,6 60 7,9 7,45
10 70

5,5 10 10/2= 0,4

import re

① import collections → Counter.most_common()

import math → math.sqrt()

BASICS!

random.randint() 3rd

random.choice()   set()

[start : stop : step]

L[-1:] → laatste rij

L[:-1] → laatste kolom

* glog(a) = b → $g^b = a$

2log(16) = 4 → $2^4 = 16$

PLT

plt.hist

plt.bar

* plt.plot([ ], [ ]), df.plot.scat

Regex

re.sub(r' ', " ", text)

re.findall()

\w → word → [A-Z a-z]

\d → digit → [0-9]

\ → Escape spec. char.

*? → 0 of meer, 1 of meer

. └→ any char

() → capture {} → aantal

Numpy

.log2() .info()

.shape() .linspace()

.size() .full()

.eye()

.substract()

.multiply()

.sin, cos, sqrt()

.corrcoef()

.transpose()

.mean()

.reshape()

.arrange()

.ones()

df[[ ]]. grid.shape()

min(axis=1) └→ [i -1 % "],...

midden → L[int(len(L/2)-2 : " +2, " -2 : " +2]

Pandas

pd.read_csv()

└ compression, skipinitialspace, sep, names=[ ]

df.loc[" "].isin() of df[df[" "] == " "]

.crosstab([ ], [ ])   .unique()

.dropna(subset = [ ]) / .notnull()

df[~df[" "].str.contains(" "), str.lower()

df.pivot_table(values, index, columns,

aggfunc, margins = true)

pivot.sort_values([" "], ascending)

df.column.sum()

df.column.index.values()

pivot[(pivot[" "] >= 10.000) & ((pivot["ratio"] >= 4

max_error = 1/17 * 100

acc = sum(error ...) / sum([“All”] * 100

rel_error = hoogste error

most_errors = meeste x fout

Accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$

f_error : F.sum() + m_error : pivot.m.sum() / pivot.all.su

→ = acc_error

Recall = $\frac{TP}{TP + FN}$ precision = $\frac{TP}{TP + FP}$

pd.head() .groupby()

.tail()

.plot()

voorspellen ipv uitleggen

np.linspace(0,1,5) - 0, 1, 0.4
L.mean(axis=0)
np.concatenate([a,b], axis=1)
np.full((3,5), 3.14)

np.all(x==6)
x[[2,1,8,4]] == 10
np.argmin / x.argmin index of min value

Rule 3: if in any dimension the sizes disagree and neither is equal to 1, an error is raised
data.loc[data.density > 100, ['pop', 'density']]
np.nansum() , pd.merge(df1, df2, on='employee')
df.dropna() , pd.merge(df1, df2, left_on='employee', right_on='name') , zelfde mee aber bedoeld woord
.drop('name', axis=1) erachter om maar een te gebruiken
pd.join is zelfde als merge, alleen custom op index
pivot_table = df.pivot_table(values='births', columns='name', index='year',
aggfunc = sum
Nadelen excel: schaalt niet, beperkte functionaliteit, integratie met andere tools
cross = crosstab(df.Jaar, df.Party)[partijen[:10].index.values]
1 Panda helps filling in the gap with Python concerning data analysis and modeling, without having to switch to a more domain specific language like R
2 Python data analyze excels in performance, productivity and the ability to collaborate

np.arange(0,144,12) , create_array(144,7) , np.reshape(L,(y,x))
laatste kolom = L[:(alles), -1] , L[begin:eind][begin:eind]
df.columns voor namen, len(df.index) geeft hoeveelheid instanties
index.value_counts
cito.index.str.lower().str.strip().str.replace(' ', '').sort_values()
cito[~index.str.contains('school')] , cito.describe()
cito['RMSE'] np.sqrt((cito - cito.verwacht)**2) , arr[::-1]
np.random(0,10,(3,3))
x[1::2] every other elem x[5::-2]
np.split(x,[3,5]) , splitsen op 3 en 5
x<3 geeft bool array data.(i)loc[ ]
data.nansum , data.fillna(0), data.notnull()
df.groupby('key').sum() pd.read_csv/html
data.plot(x='', y='', kind='bar')

Kom met hypothese
verzamel data
schoon op
herstructureer
analyseer
rapporteer

ziek ingeschat en klopt
ziek in en fout
niet ziek en klopt
niet ziek en fout

$\frac{9}{90} \big| \frac{81}{1}$

$acc = \frac{TP + TN}{TP + FP + TN + FN}$

	TP	FN	90
10 / 90	FP	FN	10

$99 \quad \frac{9}{} \big| \frac{89}{1}$