magic | %lsmagic dir(np) dir(pd)

%ls -lh working dir with bytes
%rm blabla.text removes blabla.text
Variabele met '?' erachter voor info.
%prun, %time check resources gebruik.
%paste gekopieerde code met >>> ... runnen.
%run myscript.py use script from somewhere.
acces shell with !
pwd = print working directory
ls = list working directory
cd = change directory of file (geen aanhalingstekens)
mv '.../...' = move de file hierheen.
mkdir = make new directory
tmp = temporary file place
cp = copy file..

classification: regression: RMSE
Recall = TP/(TP+FN)
Precision = TP/(TP+FP)
accuracy how many of all predictions where correct!

- plot(kind='bar') df.dropna(subset=['col'])

Datascience = Based on patterns in data predicting a value on new or unseen data.

paginas = BeautifulSoup(source_file,"html.parser")
- find_all('page')

newpagina1 = page.get_text()
" " 2 = re.sub("]]", "", newpagina1)
df['col'] = df['col'].str.replace('is', '')
id = digits \D alles behalve
\w = letters \W alles behalve
df[~df.col.str.contains("vragen", na=False)] {\w alle woorden bijven}

Join combine

df1.append(df2) add rows in df1 to end of df2. (rows should be identical)
pd.concat([df1, df2], axis=1) add cols in df1 to end of df2 (rows should be identical)
df1.join(df2, on=col1, how='inner') joins the cols in df1 with cols in df2 where rows have identical values on col1

Statistics

df.describe() summary stats for num cols
df.mean() mean of all columns
df.corr() correlation between columns of df.
df.count() number of non-null values per column
df.max() highest value of each column
df.min() lowest value of each column
df.median() median each column
df.std() standard deviation each column

Viewing / Inspecting Data

df.head() first n rows
df.tail() last n rows
df.shape rows, columns
df.info() Index, datatype & memory info
df.describe() Summary stats for num columns
s.value_counts(dropna=False) unique values met counts
df.apply(pd.Series.value_counts) met counts for all columns

Selection

df['col'] return given column as series
df[['col1', col2]] these columns as new df
s.iloc[0] selection by position
s.loc['index_one'] selection by index
df.iloc[0,:] First row
df.iloc[0,0] First element of first column

Data cleaning

df.columns = ['a','b','c'] rename columns
pd.isnull() checks for null return boolean array
pd.isnotnull() checks for not null " "
df.dropna() Drop all rows with null values
df.dropna(axis=1) Drop all columns with null
df.fillna(x) replace nulls with x
s.astype(float) datatype to float.
s.replace(1,'one') Replace all 1 with 'one'
s.replace([1,3],['one','three']) " "
df.rename(columns=lambda x: x+1) mass renaming
df.rename(columns={'old':'new'}) naming
df.set_index('column_one') change index

Filter, Sort, Groupby

df[df[col] > 0,5] rows where col > 0,5
df[(df[col] > 0,5) & (df[col] < 0,7)] " "
df.sort_values(col1) sort values ascending
df.sort_values([col1, col2], ascending=[True, False])
df.groupby(col) returns a groupby object for values from one column
df.groupby([col1, col2]) object for multiple column
df.groupby(col1)[col2] returns the mean of values in col2 grouped by col1 values. mean can be replaced by any statistic function
df.pivot_table(index=col1, values=[col2, col3], aggfunc=mean) creates pivot table that groups col1 and calculates mean of col2 and col3
df.groupby(col1).agg(np.mean) averages across all columns for every unique col1 group.
df.apply(np.mean) mean for each column
df.apply(np.max, axis=1) max for each row.

Magic

%timeit
%time
%magic
%prun
%lprun
%memit
%mprun
%rm
%ls_lh

Numpy

1D
2D axis0 axis1
3D

sorting
a.sort()
b.sort(axis=0)
np.sort(a)
np.argsort(a)

inspecting array
a.shape
len(a)
b.ndim
e.size
b.type
b.type.name
b.astype(mt)

creating arrays

x = np.array([1,2,3])
x2 = np.array([(1,2,3),(4,5,6)], dtype=float)

Initial placeholders

np.zeros((3,4))
np.ones((2,3,4), dtype=np.int16)
np.ones(10)
d = np.arange(10,25,5)
np.arange(9).reshape((3,3))
np.linspace(0,2,9)
np.arange(6)[:, np.newaxis]
e = np.full((2,2),7)
f = np.eye(2)
np.empty((3,2))
np.random.random((2,2))
x = np.random.randint(10, size=6)
np.random.randint(10, size=(3,4))
np.random.randint(10, size=(3,4,5))

Array Mathematics

np.subtract(a,b)
np.add(b,a)
np.divide(a,b)
np.multiply(a,b)
np.multiply.outer(x,x) → create table
np.add.accumulate(b,a) → store
np.add.reduce(b,a) → sum
np.power()
np.absolute([-1,4])
np.sqrt(b)
np.random.random(100)
L = np.random.random(100)
sum(L)
np.sum(L)
a.sum()
np.sum(x<6)
np.count_nonzero(x<6) aantal
a.min()
a.max(axis=0) per column
a.cumsum(axis=1) per row
a.mean()
b.median()
a.corrcoef()
np.std(b)
np.any(x<6)
np.all(x<6)

broadcast

1 they are equal
or
2 one of them =1

subsetting, slicing, indexing

a[2] subsetting
b[1,2]

a[0:2]
b[0:2,1]
b[:1]

slicing
x[::,&]
x[::-1]
x[1::2]

a[a<2] bool

b[[1,0,1,0],[0,1,2,3]]
array([4,2,6,1])
b[[1,0,1,0],:,[0,1,2,3]] fancy
array([[4,5,6,4],
[1,2,3,1],
[4,5,6,4],
[1,2,3,1]])

manipulating
i = np.transpose(b)
i.T
g.reshape(3,-2)
h.resize((2,6))
np.append(h,g)
np.insert(a,1,5)
np.delete(a,[1])
np.concatenate((a,d),axis=0)
np.vstack((a,b))
np.hstack((e,d))

np.split

Pandas

data = pd.read_csv('text.csv')
data.values
data.index
data = pd.Series([1,2,3,4], index=['a','b','c','d'])
pd.Series(data, index=index)
df_Naam = pd.read_csv('bestand.csv')
dfNaam = df_iets[df_iets.kolom == "string"]
print(len(df_Naam.kolom2 == "string"
kijken → df.shape()
df.head()
df.tail()
df.describe
s.value_counts(dropna=False)
df.apply(pd.Series.value_counts)
df.values
df.size

df.drop('xx', axis=0) → delete row
df.drop('xx', axis=1) → delete column

selection

s.iloc[0] selection by position
s.loc[0] selection by index
df.loc[0,:] → first row
df.loc[0,0] first element of first column.

cleaning

df.columns = ['a','b','c'] rename
df.dropna()
s.replace(1,'one')
s.replace([1,3],['one','three'])

df*5
df['B':'C'] * df['B':'C']
df.plot(kind='bar')
df[1:3]

a 1
b 2
c 3
a 4

alle bomen na 2000
bomen[df['plantjaar']>=2000]
bomen[df['plantjaar']>=2000]
['plantjaar','boomsoort']

Join combine
sort group by
df.groupby(['col1','col2'])
df1.append(df2)

```
np.arange(1, 10, 2).reshape(3,3)
          begin end step
                =5
np.concatenate([x,y]) -> join  - pd.concat()
np.vstack([x,y]) -> vertical
np.linspace(0,6,4)  -> 0,2,4,6
            start end elements
np.add.reduce(x) -> sum of all elements
np.count_nonzero(x<6)
np.sum(x<6)
np.add.at(x, i, 1)
          array index plus
np.partition(x, 3) -> 3 smallest numbers
data = np.zeros(4, dtype={'names':('name','age'),
                          'formats':('U10','i4')})
np.sqrt(((x1-x1.mean)**2).mean()) = np.std(x1)
li = soup.findAll('li', class_=re.compile('^interlanguage-link'))
  {l.a.attrs['lang']:l.a.attrs['href'] for l in li}
html_doc = requests.get(url)
soup = BeautifulSoup(html_doc.content)
comments = soup.findAll('li',class='comment')
[soup.title.text for c in comments if c.p]
re.findAll(b'beeeey', line.lower())
line.split(b',')[1]
np.random.rand(3,2)
```

Index

- intersection &
- union |
- difference ^

```
iloc -> index
loc
ix -> both
df.query("sepal_length>5 & .... @a")
df.groupby('species')['sepal_length'].max()
df.sort_values(["  ","  "],ascending=[False,True])
df.drop('species')
pd.MultiIndex
pd.merge(df1,df2) -> how='outer'
 axis=0 -> columns
 axis=1 -> rows
df.set_index("  ")
df.join()
 %magic
```

```
pd.read_excel("  ", index_col="  ")
len(set(  .index)) == len(...index)
df.value_counts()
df.sort_index()
str.lower().str.strip().str.replace('\w',"").str.replace
('*\w','').sort_values()
~.str.contains('school')
m = ci.to.mean()
ci.to[m,index] - m).tail()
df.groupby(0)[1].count().sort_values().tail()
 .lstrip()
```

?→ help()
help(len), re?

% quickref

& = and
| = or

np.array([...]), dtype='float'
np.array([range(i, i+3) for i in [2,4,6]])
np.zeros | np.full((3,5), '2', dtype=str)
np.ones

np.arange(0,20,2)
np.linspace(0,1,5)

. ndim . copy()
. shape np.sort()
. size
. dtype

3D Array:
np.random.randint(10, size=(3,4,5))

2D Array:

2d [x,y]
2d [:x, :2] every other column
2d [::-1, ::-1] Reversed array

1D Array:

1d [N], 1d [:N], 1d [N:]
1d [::2] every other element
1d [::-1] all Reversed

Van 1D naar 2D:

np.arange(1,10).Reshape((3,3))

Samenvoegen:

np.concatenate([x,y,z], axis=...)
np.vstack([x, grid])
np.hstack([grid, y])

Splitten:

x1, x2, x3 = np.split(x, [3,5])
upper, lower = np.vsplit(grid, [2])
left, right = np.hsplit(grid, [2])

Sum/mult. of all elements:

np.add.reduce(x)
np.multiply.reduce(x)

Mult. table:

np.multiply.outer(x, x)

Hoeveel minder dan 6?

np.count_nonzero(x<6)

List(data.items())

data.values
data.index
data.columns
data['columns/rows']

2D array → Df

pd.DataFrame(np.random.rand(3,2),
columns=['foo', 'bar'],
index=['a', 'b', 'c'])

Indexing & slicing

data.loc[1]
data.iloc[1]

Ufunc

column.index

NaN → 0

fill_value=0
data.fillna(0)

data.isnull()
data[data.notnull()]

Drop all if column value=0

df.dropna(axis='columns')
(axis='row', how='all')

Mult. indexed → DF

data.unstack()

Samenvoegen:

pd.concat([ser1, ser2])

Merge into one DF:

df3 = pd.merge(df1, df2)

Van 2D naar 1D Array:

L.flatten()

(n,1) → 1D met n aantal waarde in L:

np.reshape(L, (len(L.flatten())), 1)

Laatste kolom 2D Array:

Return (L[:,-1:])

Array tafel van (n,x):

np.arange(0, n*x, x)

Alle elem. uit array L deelbaar door n:

output=
[L[x] in range(len(L)) if L[x] % n == 0]
Return np.asarray(output)

Crosstable:

pd.crosstab(df.col1, df.col2)

Bepaalde value in column:

df[df['col1'] == 'value'][col2]

Bepaalde value tellen:

df['col1'].str.count('\?')

---

sqRt = int(np.sqrt(len(L)))
np.reshape(L, (sqRt, sqRt))

Mean_per_column:

np.mean(L, axis=0)

omdraaien 2D array:

L.transpose()

---

df['col'].value_counts()
df.str.replace('[^\w\s]', '')
df[~df.str.contains("-")]

Pivot table:

df.pivot_table()
(index='_', columns='col1', 'col2', values='col',
aggfunc=np.sum, margins=True, margins_name='all',

Sorteren op groot→klein:

df.nlargest(len(df), 'column')

Reading from the web:

import requests
f = requests.get(url)
lines = f.text.split('\n') → [line.split(';') for line in lines] → f.close() → len(lines)
lines[:2]

Pd.merge(df1, df2, left_on='column1', right_on='col2').drop('col2', axis=1)
Pd.merge(df1, df2, how='inner')
↓
'outer', 'left', 'Right'

```
df.index = df.index.replace(r'[a-zA-Z]*', '', regex=True)
df = df.drop(df[df.index.str.match('.*word.*', na=False)].index)
df.index.value_counts()[df.index.value_counts() > 1]
df.index = df.index.str.lower()
df['RMSE'] = np.sqrt((df.A - df.B)**2)
M = df.mean(); (df[m.index] - M)
df.A.sort_values().plot(kind='barh')   (== .plot.barh())
df[df.A > 100].B.value_counts().sort_values.plot(kind='barh')

np.random.randint('int')
          .choice('list')

for i, x in df.iterrows()
pd.read_csv(file, sep='\t', dtype=str, header=None, names=['...',...], index_col='...

np.arange()
np.array.reshape(x,y)
np.array.shape
np.mean()
np.median()
df.A.mode()
df.A.corr(df.B, method='pearson')
pd.pivot_table(df, index='...', columns='...', values='...', aggfunc=np.sum, fill_
               value=0, margins=True).sort_values(by=('...'), ascending=False,
               kind='mergesort')

pivot.loc[(pivot.A < .1) & (pivot.A > -.1)].index.values

list.extend(list2)

plt.plot(x, y)
plt.legend(['...',...])
plt.show()

page = re.sub('<[^<]+?>', '', page)
```

Import re
Import Collections - Counter.most_common()
Import math - math.sqrt()

PLT

plt.hist
plt.barh
john.pivot_table(values='births', index='year', columns='sex', aggfunc=sum).plot().

Basics
random.randint()
random.choice()
└ [-1:] laatste rij
└ [:-1] laatste kolom
$g \log(A) = b \quad g^b = a$
$2 \log 16 = 4 \quad 2^4 = 16$

Numpy
.info()
.shape()
.size()
.inspace()
.full()
.eye()
.subtract
.transpose()
.mean()
.reshape
.arrange
.ones
grid.shape = [-1, "%"]
.mode()[0]
.groupby

Pandas

```
kvr = pd.read_csv(File, sep='\t', skipinitialspace=True,
        na_values="?")
kvr.columns = ["naam", "naam2"]
kvr_obj = kvr.select_dtypes(['object'])
kvr[kvr_obj.columns] = kvr_obj.apply(lambda x: x.str.lstrip())
kvr["partij"].nunique()

maxerror = 1/big x 100
real error = sum(error) / sum["all"] x 100
.sort_values(ascending=False)
subset = names[(names.name=='john') |
         (names.name=='mary')]
```

| = or
& = and
~ = not

names.query("name=='john' or name=='mary'")

rel_error = boogste_eror
most. error = meeste x fout

Accuracy = $\frac{tp+tn}{tp+tn+fp+fn}$

recall = $\frac{tp}{tp+fn}$

precision = $\frac{tp}{tp+rp}$

re.sub(r, " ", text)
re.Findall()
\w = word
\d = digit
.*? = 0 of meer . 1 of meer
↳ any char

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

np.random.randint(low, high, size)
np.random.choice(a, size)

win_percentage(prizedoors, guesses):
    100 * (np.array(prizedoors) == np.array(guesses)).mean()
    → == op numpy array kijkt per positie of ze gelijk zijn en geeft True/False terug (etc. >, <)

np.absolute()
np.arange(start, stop, step) → make 1D array
np.reshape: [array-naam].reshape(shape) (tuple of int) → shortcut for np.reshape()
→ [1,2,3] = a, a.reshape(a.size, 1), a=[[1],[2],[3]] (column vector)
np.shape(array) → ook array.shape
np.size(array) → ook array.size
np.extract(condition, array) → return elements of array that satisf. condition in array >0.
np.mean(array, axis)
np.transpose(array)

Numpy functies

np.Log()

Plotje Pandas

df = pd.DataFrame(data)
Plotje = df.plot(kind=.., title='...', ...);
Plotje.set_xlabel('....');
Plotje.set_ylabel('....');
→ if you want to plot your own date, e.g. switched & not switched, consider using a dict comprehension to make the data for the dataframe

n	stay	switch
3	perc.	perc.
4	perc.	perc.

→ data={'switch': {n: percentage}, 'stay': {n: percentage}}
stay&switch → lijnen, 3,4 x as, perc

hist = kolom.hist(bins=...)
hor.bar = kolom.value_counts().plot(kind="bar")

Pandas functi

pd.read_csv()
pd.crosstab(col1, col2)
dataframe['column'].value_counts() → counts how many time elements oc in the colu
df[df.column=='...'].column.value_counts()
df.df[df.column == '...'] → waar rij in deze column gelijk is aan string hoe
df[[column1, column2]].corr()
df.column.mean()
df.column.mode()
df.column.median()
df.pivot_table()
df.sort_values(
df.drop() → liever subset maken
df[['column1', 'column2',
df.column.min/max/sum
df.copy()

Numpy slicing

fixed type arrays

X1 = [1,2,3,4,5,6]
X1[0] ⇒ 1
X1[-1] ⇒ 6
X1[:3] ⇒ [1,2,3]
X1[3:] ⇒ [4,5,6]
X1[::2] ⇒ [1,3,5]
X1[1::2] ⇒ [2,4,6]
→ every other element starting at index 1
X1[::-1] ⇒ [6,5,4,3,2,1]
X1[3::-2] ⇒ [4,2]
→ every other element reversed, starting at index 3
y1 = [98,99]
np.concatenate([X1,y1]) ⇒ [1,2,3,4,5,6,98,99]
→ also possible with more than 2, or with grids (2D) default: add 2nd grid as rows, with axis=1 add as columns

X2 = [[1,2,3],
      [4,5,6],
      [7,8,9]]
X2[2,0] ⇒ 7 (X2[row,column])
X2[0,0] = 7 (assigning value 7 to row0 column0)
X2[:2,:3] ⇒ [[1,2,3], two rows, three columns
             [4,5,6]]
X2[:3,::2] ⇒ [[1,3], all rows, every other column
              [4,6],
              [7,9]]
X2[::-1,::-1] ⇒ [[9,8,7],
                 [6,5,4],
                 [3,2,1]]
X2[:,0] ⇒ [1,4,7] first column of X2
X2[0,:] ⇒ [1,2,3] first row of X2
↳ zelfde: X2[0]
X2[:2,:2] ⇒ [[1,2], two rows
             [4,5]] two columns

Pandas subsets

columns: df[['col1','col2']]
single column as series: df['col']
row: df.loc['index_of_row']
rows: df.loc[['index1','index2']]
range of rows: df.loc[idx1:idx4] (t/m)
range of rows w/ steps: df.loc['idx1':idx4':2]
rows & columns: df.loc[[rows]][cols]]
row & column: df.loc['row','col']
all rows & column: df.loc[:,[col1,col2]]
index selecting: df.iloc[slicing as in numpy slicing]

Cleaning

from b34 import Beach
soup = BeautifulSoup('string', 'html.pars
soup.findAll('html-tag', ...
re.sub('regex', 'subst', 'str
list_of_patterns = ['1s', '\w', 'word', ...]
compiled_re = re.compi
"|".join(list_of_pattern
df.column = df.column.s lower().str.replac (compiled_re, '')
df = df.dropna(subset=
df.column = df.column.app (str)
df = df[~df.colu str.contains(

File

with b22.open(file, encoding='utf8', mode='rt') as file:
    for line in tqdm_notebook(file):
        ...
progress bar

Pd series

value_counts()
min(), max(), sum(), mean(), mode()
median(), abs()

Splitting of Arrays

grid = [[0,1,2,3],
        [4,5,6,7],
        [8,9,10,11],
        [12,13,14,15]]
X = [1,2,3,99,99,3,2,1
X1,X2,X3 = np.split(X,[3
⇒ [1,2,3] [99,99] [3,2,1]
upper, lower = np.vsplit(grid,[2]) ⇒ upper: [[0 1 2 3], [4 5 6 7]]
lower: [[8 9 10 11], [12 13 14 15]]
left, right = np.hsplit(grid,[2]) ⇒

left	right
[[0 1]	[[2 3]
[4 5]	[6 7]
[8 9]	[10 11]
[12 13]]	[14 15]]

Precision / Recall

Precision = TP/(TP+FP)
Recall = T/(TP+FN)

Numpy Broadcasting

0	0	0
10	10	10
20	20	20
30	30	30

+

0	1	2
0	1	2
0	1	2
0	1	2

=

0	1	2
10	11	12
20	21	22
30	31	32

Boolean indexing

arr = [1,2,3,4]
arr[True, False, True, False] = [1,3]
arr[arr < 3] = [1,2]
arr[(arr > 1) & (arr < 4)] = [2,3]

grid = [[1,2,3],
        [4,5,6]]
x = [7,8,9]
y = [[99],[99]]

np.concatenate([grid,grid]) ⇒
[[1,2,3],
 [4,5,6],
 [1,2,3],
 [4,5,6]]

np.concatenate([grid,grid],axis=1) ⇒
[[1,2,3,1,2,3],
 [4,5,6,4,5,6]]

np.vstack([x,grid]) ⇒
[[7,8,9],
 [1,2,3],
 [4,5,6]]

np.hstack([grid
[[1,2,3,99]
 [4,5,6,99]

%magic %timeit %cp
%time %ls

Concatenation of Arrays

pandas

Pd.Read_csv()/xml

Select: df[df.kolom == 'value'], kolom[o]
df.loc[(df.kolom == 'value')&()]

Toevoegen: df['naam'] = df.b/df.a

sorteren: df.sort_values('kolom', ascending=[false, false])

value_counts() = waarden tellen van kolom. Hoe vaak komt iets voor?

boolean mask: x[x%3==0]

shape .size .describe() .ndim

samenvoegen: pd.merge(x,y)
1-1 many-1 many-many
many-1 = 1/2 vd kolommen heeft duplicaten

pd.concat([x,y]) of x.append(y)

nans: df.fillna(0) df.dropna
np.nansum(array) df[df.notnull()]

fancy indexing
c = array([[3,5,2,4], [7,6,8,8], [1,6,7,7]])

3 rijen, 4 kolommen
axis=0 kolom
axis=1 Rij

get diagonal (3,6,7)
x[[0,1,2],[0,1,2]]

plotten: .plot(kind='barh')

aanpassen: df.kolom = df.kolom.str.lower()

crosstab: pd.crosstab(df.kolom, df.kolom)[lijst]

pivot: df = pd.Read_csv(file)
pd.pivot_table(df, index='name', columns='sex', values='count', aggfunc=np.sum, margins=True)
spreadsheet style as Dataframe

index .index .values

groupby: df.groupby('species')['length'].max
geeft grootste waarde van kolom length met specie ervoor
setosa 5,8
virginia 6,1
(what are max lengths for each specie?)

df.ff.unique() data[1] -> transpose (kolom = rij)

numpy

Create: np.arange(start, stop, step)
np.random.randint(5, size=10)

slicing: x[::2] elk even getal
x[::-1] alle elementen, omgekeerd
x[:2, :2] geeft 2x2 array

Reshape: x.reshape((3,4))

samenvoegen: np.concatenate([x,y])
axis=0 verticaal toevoegen (vstack)
axis=1 horizontaal toevoegen (hstack)

Rekenen: ~~or~~.min(axis=0/1)
x.argmax() vindt index max value
np.count_nonzero(x<6)
hoeveel values zijn kleiner dan 6?
np.any(x>8) zijn er values groter dan 8
np.all(x<9) np.amax(L)

Sorteren np.sort(x, axis=0/1)

Toevoegen np.insert(L, index, object)

literature		is sick	is not sick	
Accuracy = 95	positive test	95	495	590
	negative test	5	9405	9410
		100	9900	

Recall = TP/(TP+FN) = 95/95+5

Precision = TP/(TP+FP) 95/95+5

Regression = RMSE, rest = classification

Variance = $E(x-\mu)^2/N$ df.std

STD = √variance

2,5,6,9 = 6,93

data science steps:
- interacting outside world
- preparation
- transformatation
- modeling/computation
- presentation

6-2-2015 - free
29-02-09 - fixed

Broadcasting
1. if 2 arrays differ in ndim, shape of one with fewer dimension is padded with ones on its leading left side
2. if 2 arrays not match, array with shape equal to 1 is stretched to match
3. if still not same and not 1 = ERROR

```
a = array([[0,0,0],
           [1,1,1],
           [2,2,2]])
b = array([0,1,2])
1. a(3,1) b(3,) →
2. a(3,1) b(1,3)
3. a(3,3) b(3,3)
a+b = array([[0,1,2],
             [1,2,3],
             [2,3,4]])
```

%timeit
%magic
%lsmagic
%ls -lh = fliks

functies

dataframe van lijst met dicts
df = pd.DataFrame.from_dict(list)
df['id'] = df['id'].str.get(0)

meest pop woonplekken
df = df.woonplaats.value_counts()[:20]
df.plot(kind='barh')

Partijen met vrouwen
df.Partij.loc[[df.geslacht == 'vrouw']].unique()

tekst tokenize, most occuring tokens
C = counter(text.split())
CS = pd.Series(C).sort_values(ascending=False)
CS.head(10)

find tokens in text which occupy in total 24 characters
CS[CS*CS.index.str.len() == 24].index

Check voor dubbeler in text
len(set(cito.index)) = len(cito.index)

welke namen komen dubbel voor?
dubbel = cito.index.value_counts()
dubbel[dubbel > 1]

verwijder spaties, gekke tekens & lower
cito.index.str.lower().str.strip().str.replace('"', '').sort_values()

Bepeek cito tot scholen zonder sub 'school'
citoT = cito.index.str.lower().str.contains('school')].head(10)

Bereken RMSE
cito['RMSE'] = np.sqrt((cito.bla - cito.40)**2)

Plot inkomens per school
Cito.inkomen.sort_values().plot()

in welke gemeente is inkomen >50000
cito[cito.inkomen > 50000].gemeente.value.counts().sort_values().plot(kind='barh')

maak lijst met tuples van items
list(data.items())

meerdere kolomen pakken
df[['kolom1', 'kolom2']]

de ie rij pakken
df.iloc[i]

Kolom als index
df.set_index('Kolom')

Regex
p = re.compile('\w+')
P.findall('string')
re.split('[\w]+', string)
re.sub('[\w]+', '', string)
\w = (a-z, A-Z, 0-9)
\W = not ↑
\d = (0-9)
\s = (space, tab, newline)

Beautifulsoup
from bs4 import Beautifulsoup
soup = Beautifulsoup(html-doc.content)
soup.findall()

laatste rij = L[-1:]
laatste kolom L[:,-1]

kind met meeste bordstag leden
bd[0].value_counts().head(1)

partij grootste aantal leden uit 1 deelstaat
bd.groupby(0)[1].count()

cefeutentamen

fancy indexing

arr[[rows],[cols]]

plot samples

- value_counts().plot(kind='bar(h)')

df X, y

X, y = dataset.drop('col', axis=1), dataset.col

- Z-normalize (column based)

((X - X.mean())/X.std()).head(), '',

(X - X.mean())/X.std()).std()

double sort

iris.sort_values([col1, col2], ascending=[t/f, t/f])

- find tokens in text 24 chars & print

CS[cs * (CS.index.str.len() == 24].index

- hand met meeste bondsdag leden & #

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

- vs pandas

cito[cito.inkomen > 50000].gemeente.value_counts()

.sort_values().plot(kind='bar(h)');

NumPy

- 1D → 2D row or col

X[np.newaxis, :] or ↔

- np.concatenate([grid, grid], axis=0/1)

0 → #cols similar 1 → #rows

- mixed dims

vstack / hstack

- split

np.split(x, [pos1, pos2, n])

n+1 variables

v/hsplit (x, [2])

dsplit → 3rd axis

- multiplication table

np.multiply.outer(x, x)

- reduce array for binary - .any/all

np.add.reduce(x)

- store intermediate with accumulate

or ex. cumsum

- np.linspace

- np.eye (3) identity matrix 3x3

- itemsize, nbytes

- slicing

x[begin:end(:step), begin:end(:step)]

x[0] == x[0, :]

- min/max axis 0 for cols

- nansum

- argmin (index)

- broadcasting

M.shape = (2,3) (2,3) → (2,3)

a.shape = (3,) → (1,3) → (2,3)

- values less than 6?

np.count_nonzero(x<6)

also in agg. functions

- Boolean operators & array mask

- fancy ind → shape of result reflects shape of index arrays

- np.sort(a, axis=-1, kind='', order)

sorted copy

- np.add, subtract, ... (a, 5)

Theorie

Precision → how many positive actually positive TP / TP + FP

Recall → how many positive samples are captured by the pos. prediction: TP / TP + FP

Data Science → based on patterns in data predictions are made on unseen data

Steps: interacting with outside world → preparation → transformation → modeling & computing → presentation

Study of the generalizable extraction of knowledge

make implicit knowledge explicit & in format that is computationally fast for computer and still corresponds to human's view of that information

Pandas

x fx in df.iterrows

df.iloc[[0],[0]] → single value by row & col

df.iat([0],[0])

df.loc[[0], ['Country']] → row & col label

df.ix[2] → single row of subset of rows

df.ix[:, 'Capital'] → single col of subset of cols

df.ix[1, 'Capital'] → select rows & cols

df.sort_values(by=...)

df.rank() → assing ranks

df.idxmax() → index

- pivot table

pd.pivot_table

- get name where age < 30

data[data['age'] < 30]['name']

L[start:eind:stap] → even: L[::2]
Dotproduct: $\sum x_i y_i$ → s.dot(s)
DS: Predicting value on unseen instances based on patterns in data → regr./class
evaluate: class: Accuracy, Precision, recall, F1
regr: RSME

Precision: how many pred. were corr? $\frac{TP}{TP+FP}$
Recall: How many C's predicted as C? $\frac{TP}{TP+FN}$
Accuracy: how many of all pred. were corr?

	Has D	No D	
Pos T	TP	FP	= alles P voorspeld
Neg T	FN	TN	= alles N voorspeld
	= Alle zieken	= alle gezonden	

Impliciete info expliciet maken
. lees baar voor computer in een formaat dat computationeel snel is, en aansluit op het beeld vd mens mbt die informatie
2015-02-09: alfabetisch ordenen.
Spreadsheets: tabellen:
Delete, insert, update: rijen/kols weghalen/toev.. = nieuw element / nieuw eigenschap
Andere view: sorteren, slicing
Aggregatie: nieuw veld = functie
Combineren van sheets
excell/open off/Google: lastig mergen
Google fusion = goed in grote sheets
excell: schaalt niet naar grote sets
- steeds dezelfde handmatige GUI commando's met voor zware berekeningen
- Functionaliteit beperkt: opschonen moeilijk
- Geen integratie met andere tools/libery's.

Stappen DS: 1. Interacting outside world: reading/wtr. data formats
2. Prep.: cleaning, combining, normaliz. slicing, etc.
3. transformation: applying maths/stats
4. Moddel./computation: connect data to ML alg,
5. Presentation: create visualizations

MVM: verzamel, clean, herstructureer, analyze, rapporteer.

vragen_PvdD.Ministerie.value_counts().idxmax()
barchar = 4, barchar[barchar >= 5].plot.barh()
Pivot(values = waarde in cel, index, cols, aggf: operatie → voor values, margins)
Pivot(CF, M]).min(axis=1).sum()
No_antL = kvr.vraag[kvr.vraag.str.count(r'\?') <= 50].str.count
app_per_row = add_per_column(L.T).T

np: (rij, kolom): 2D, columnnames/values,): 1
Axis 0: within each column
1: within each row
np.concatenate voor joinen 2 arr's in NP
↳ .vstack/hstack
np.abs = absolute
.linspace = start, end, aantal elementen
Broadc. 1. Shape with fewer dimen +1 left side
2. arr. with 1 stretch to match
PD: .iloc = rijen pakken op indexen (getallen)
.loc = rijen pakken op label
.loc[rij, col, st]
voorbeeld TT:
Boolean mask: x1[x1 % 3 == 0]
Fancy index: x2[[0,1,2]]
.plot(kind='barh')
iris.drop('species', axis=1)
iris.sort_values(['col', 'col'], ascending: [T,F])
iris.groupby('species')['sepallength'].max()
Counter(List) → pd.Series(counter(list))
cs[cs.cs.index.str.len() = 24].index
bd.groupby([party, land]).count().sortvalues ...
magic: % : line magic %% cell magic
Pasting code blocks: % Paste cpaste
Run external code % run
timing: %(%) timeit, %time, %prun
magic functions: % magic → list: % lsmagic
Print(_): Print Previous output
Out[2] <+ 2 .... % history
working dirr: % pwd, content+size % ls
change dirr: % cd, make dirr % mkdir
move file: % mv file loc, remove: % rm
all data/info from var: var?
memory usage: % memit % mprun

PD.multiIndex.from_tuples([(c,c,c)])
or. df(data, index=[c1],[c2].cols...)
np.concatenate([x,y,z]) ... axis.
PD.concat([df1, df2], axis) inner
Pandas preserves indices
PD.merge: intelligent → zoekt naar zelfde kolomnaam
PD.merge(on='keyword') outer
on indices df1.join(df2).
.isnull

A.species.value_counts().plot(kind='barh')
dataframe X numerical data, series Y species inf.
X, Y = A.drop('species', axis=1), A.species
Z-normalize data in each X.kolom by substracting
mean of the column from each cel and divide
by the standard deviation of each column.
Also show each colum of the normalized data
((X-X.mean())/X.std()).head(), (X-X.mean())/X.std()).std()
sort by Name and ALL in reverse
pivot.sort_values(by=['ALL','NAME'], ascending=[False, TRUE])
maximal length each species
A.groupby('species')['length'].max()
length larger than 5 and width smaller than $\sqrt{5}$
R = np.sqrt(5)
A.query('length > 5 and width < @R')
tokenize with split and count each token.
create serie with this dict. and sort it
C = Counter(text.split())
cs = pd.series(C).sort_values(ascending=False)
tokens that occupy in total 24 characters
eg: if ida occurs 5 times it occ. 15 charact.
cs[cs*(cs.index.str.len()) == 24].index
what % of all unique words occurs just once?
and what % of all words in text?
cs[cs==1].sum()/cs.count(), ((cs==1).mean(),
cs[cs==1].sum()/cs.sum(), (cs==1).sum()/cs.sum()
Hist for each number i of occurances how many
tokens in text occur i many times (so if there are
8 words which occur each 4 times, we have an
8-high bar at x-value 4)
pd.series(Counter(cs.values)).plot(kind='bar');
cs.value_counts().sort_index().plot(kind='bar')
meeste leden met aantal leden van dat land
pd.groupby(0)[1].count().sort_values().tail(1)
grootste aantal leden uit 1 deelstaat
pd.groupby(0)[2].value_counts.sort_values.tail(1)

x1[-3:], x1[3:]
last 3 columns of first two rows
x2[:2,-3:], x2[:2,1:]
reshape into 1-dimens. array
x2.reshape((12))
change each element to its
squared difference from the mean
(x1 - x1.mean())**2
np.sqrt((x1-x1.mean())**2.mean()) == x1.std()
boolean mask on x1 and reduce x1 to int
divisible by 3
x1[x1 % 3 == 0]
fancy indexing to get the diagonal of x2
x2[[0,1,2],[0,1,2]]
use comprehension to compute sum of squares
sum([x**2 for x in big_array])
universal functions
sum(big_array**2)

files in directory, how large
!ls -lh
remove
!rm no_good.txt
what resources does the algoritm use?
%time, %timeit, %prun
find out datatype using magic command
%?
1d array deelbaar door 0
def ... bla
    np.arange(n) % 0
maak float van de array,
np.array([1,2,3], dtype='float32')
np.array([range(i, i+8) for i in [2,4,6]])
np.linspace(0,1,5)
np.random.randint(0,10,(3,3)) #array of 3x3

allelements reversed
x[::-1] #1d array
every other from
index 5
X[5::-2]
two rows, 3 columns
x2[:3, ::2]
all rows, every other c
x2[::, ::2]
completely reversed
x2[::-1, ::-1]
first column of x2
x2[:,0]
3x3 grid
.reshape(3,3)
x[:, np.newaxis]
np.multiply(x, 10, out=y)
np.add.reduce(x)
sum of all elements
min value of column
M.min(axis=0)
in each row how many?
np.sum(x<6, axis=1)
sort each row
np.sort(x, axis=1)
np.sum(~((inch<0.5)|(inch>1)))
all non zero in python
are True, 'and' is over
an entire object.
[1 tm 4]
ind = np.array([[3,2],[4,5]])
x[ind] #
[[a,e],
 [e,f]]
fancy indexing

the shape of the one with fewer dimensions is padded with ones on its leading (left) side
M.shape → (2,3)
a.shape → (1,3)
M.shape = (2,3)
a.shape = (3,)
if the shape of the arrays does not match in any dimension, the array with shape=1 in that dimension is stretched to match the other shape
M.shape (2,3)
a.shape (2,3)
error is raised if in any dimension the sizes disagree

pd.merge(a, b, how='inner')
pd.merge(a,b, left_on='read', right_on='help').drop('help', axis=1)
Nietdeelbaar = np.array(candidates)[d, dtype=bool)
↳ Erasthostenes: lijst 2 tot max → kleinste getal → streep alle veelvouden door (incl getal zelf)
Grootste gemene deler (Euclides)
def ...
  while A!=0 and B!=0:
    if A>B:
      A = A%B
    else:
      B = B%A
  return(max(A,B))
↳ toevoegen aan priemgetallen
600 = 4·136 + 56 (600, 136)
136 = 2·56 + 24
56 = 2·24 + 8
24 = 3·8 + 0

pivot['ratio'] = np.log2(pivot['M']/pivot['F'])
subset = pivot[(pivot.ratio > -0.1)&(pivot.ratio<0.1)].index
ambi_names = set(subset)
act_error = 100*(pivot[['F','M']].min(axis=1).sum()).pivot['ALL'] sum()

kvr = pd.read_csv(linktofile() compression='gzip'
    sep='\t', index_col=[0] #eerste colom = index
    header=None, names=['joepie','ja']
    skipinitialspace=True)

dot product:
sum(a[i]*b[i] for i in range(len(a)))
sum(i*i for i in baby)
df[df.A=='b'][['A','B']]#kolom A en B
projection:
kvr[['jaar','partij']]
s.argmax()# naam hoogste cijfer
s.idmax()# index of first occarance of max
pivot['colfromindex'] = pivot.index
pivot.drop('colfromird', axis=1, inplace=True)
↳ axis=1 dan drop je de kolom

versnel je programma door:
- vervangen for-loops door vector berekeningen
- np.arrays ipv lists en sets
- sleep geen onnodige ballast mee.

breadt-first search: afstand tussen node n in netwerk G, alsin de lagen
bipartiet = netwerk met twee sets.
9 feb != ambigu
impliciete info: wilders zijn uitspraken

lemma = onhandig om definities in spreadsheet, beter in XML of JSON

data science = classification (spam), regression (predicting, e.g. grade) (RMSE) → niet als normale statistiek, je ligt niks uit.
interacting with outside world, preparation, transformation, modeling and computing, presentation
→ verschillende file formats
↳ schoon maken, ↳ berekeningen om nieuwe sets te krijgen
↳ data mining
↳ statistieke modellen
↳ data visualisatie
data science = generalizable extraction of knowledge from data
slicen

Precision = hoeveel in voorspelde klas zijn correct (hoeveel voorspeld als spam, echt spam)
Recall = hoeveel instances van klasse (voorspeld als (hoeveel van de spammail zei ik dat spam was)
Accuracy = hoeveel voorspellingen zijn correct?
test op ziekte = 95% correct, 1 op 100 heeft het:

	ziek (T)	niet (F)	
postest(T)	95 TP	495 FP	590
neg.test(F)	5 FN	9405 TN	9410
	100	9900	10.000

precision = $\frac{TP}{TP+FP} = \frac{95}{95+495} = .16$
↳ om de hoge pakkans te krijgen betaal je de prijs van veel valse beschuldigingen, accuracy is niet de juiste maat als je regt dat iedereen zwanger is, je precision en recall zijn dan 0
pakkans

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

Logaritme:
product = $\log_b mn = \log_b m + \log_b n$
ratio = $\log_b \frac{m}{n} = \log_b m - \log_b n$
powers = $\log_b n^p = p \log_b n$
Roots = $\log_b \sqrt[q]{n} = \frac{1}{q} \log_b n$
change of bases = $\log_b n = \log_a \log_b a$

if $y = b^x$ then $x = \log_b y$ $b \neq 1$
$y = 2^x \Leftrightarrow x = \log_2 y$
$2^x = 16 \Leftrightarrow \log_2 16 = x = 4$
$3^4 = 81 \Leftrightarrow \log_3 81$

Re.findall(b'brewery', line.lower())
line.split(b',')[1]
.lstrip()
index_col = ''